RANK-ONE APPROXIMATION TO HIGH ORDER TENSORS

T. ZHANG* AND G.H. GOLUB†

Abstract. The singular value decomposition (SVD) has been extensively used in engineering and statistical applications. This method was originally discovered by Eckart and Young in [11], where they considered the problem of low-rank approximation to a matrix. A natural generalization of the SVD is the problem of low-rank approximation to high-order tensors, which we call the multi-dimensional SVD. In this paper, we investigate certain properties of this decomposition as well as numerical algorithms.

Keywords. **Key words.** singular value decomposition, low rank approximation, tensor decomposition

AMS subject classifications. **AMS subject classifications.** 15A69, 65F30

1. Introduction of the problem. The problem of high order tensor decomposition has been studied by mathematicians who are interested in algebraic properties of tensors, by psychologists who need to analyze multi-way data, as well as by engineers and statisticians who are interested in high-order (tensor) statistics and independent component analysis (ICA). This decomposition is a generalization of the SVD that gives a low-rank approximation to a matrix (i.e. a second order tensor). However, a direct generalization of the SVD is non-trivial, since the definition of a rank that preserves all of the good properties of the SVD does not exist. At the current stage, little detail is known concerning general rank decompositions of a high order tensor, even though there have been a number of works in this direction [21, 24, 25, 27, 28]. As a consequence of the lack of a good tensor rank definition, there is no "best" way to define low-rank approximation for tensors of order higher than two, as pointed out in [27].

Computationally, the most popular method is based on alternating least squares minimization. However, the convergence behavior of this method has not been sufficiently analyzed. A rigorous analysis of the method is given in Section 4. We also propose a new method to compute the optimal rank-one approximation. This algorithm is a generalization of the Rayleigh quotient iteration for eigenvalue problems. If we consider a matrix as a high order tensor, then an interesting application of this procedure leads to a novel method for computing a singular value/vector pair for the matrix.

An important application of multi-dimensional SVD is multi-way analysis. Two models of decomposition have been frequently used: one is the Tucker3 model proposed in [30]; the other is the PARAFAC-CANDECOMP model proposed in [7, 16]. For third order tensors, the Tucker3 model is given by $\sum_{i,j,k} x_i \otimes y_j \otimes z_k \; g_{ijk}$, where g is an order 3 tensor called the *core array*. The PARAFAC-CANDECOMP model approximates a third order tensor by the sum of a few rank one tensors — this is equivalent to the Tucker3 model with a diagonal core: $\sum_i x_i \otimes y_i \otimes z_i$. Both models can be easily extended to the higher order case. For more detailed descriptions of these models and existing computational algorithms, see [15, 17, 18, 22, 27] and references therein.

Another application of multi-dimensional SVD is independent component analysis (or blind source separation). In this case, we attempt to find a matrix A from vector observations $x_1, \ldots, x_n$ that are taken from an unknown distribution D, such that the components of Ax are statistically independent when x is drawn from D. Many solutions have been proposed in the literature based on different formulations of this problem; see [3, 4, 5, 8, 9, 10, 26, 31] and references therein. One solution, which is based on fourth order cumulants, solves the ICA problem by decomposing a symmetric fourth order tensor into the sum of symmetric orthogonal rank-one tensors [4, 8, 10] (see Definition 3.1). Note that a fourth order tensor $[a_{ijkl}]$ is symmetric, if we have $a_{ijkl} = a_{i'j'k'l'}$ for any permutation $(i'j'k'l')$ of $(ijkl)$. From the tensor decomposition point of view, this approach to the ICA problem leads to an orthogonal PARAFAC-CANDECOMP model.

There is an interesting relationship between rank-one and rank-F approximation in the PARAFAC-CANDECOMP model. In the second order (matrix) case, from the optimal approximation property of the

*IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (tzhang@watson.ibm.com)

†Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA 94305 (golub@sccm.stanford.edu)

SVD, the optimal rank-F approximation of a tensor is equivalent to the following incremental rank-one approximation approach: we first fit the original tensor by a rank-one tensor, then subtract the rank-one approximation from the original tensor and fit the residue with another rank-one tensor. This procedure is repeated until F rank-one tensors are found. Therefore for second order tensors, the rank-F approximation problem can be reduced to the rank-one approximation problem. The simplicity of this incremental rank-one fitting procedure is very attractive; although for higher order tensors it is not necessarily equivalent to the PARAFAC-CANDECOMP approximation. However, we will show in Section 3 that for the special case of *orthogonally decomposable tensors* defined later (this special case includes the fourth order cumulants approach to the ICA problem), the incremental rank-one approximation procedure yields the solution to the optimal rank-F approximation.

Therefore computationally, we only focus on the following rank one approximation problem: find vectors x, y, and z to minimize

$$\sum_{i,j,k}(x_i y_j z_k - a_{ijk})^2, \tag{1.1}$$

where $[a_{ijk}]$ denotes a third order tensor. For notational simplicity, we illustrate our results by using third order tensors whenever generalizations to higher order cases are straightforward. Subtle differences will be mentioned when they exist.

2. Equivalent rank-one formulations. Note that in (1.1), each vector x, y, or z is only determined up to a scaling factor. Therefore we can impose the constraints $\|x\|_2 = \|y\|_2 = \|z\|_2 = 1$ and write (1.1) as

$$\min \sum_{i,j,k}(\lambda x_i y_j z_k - a_{ijk})^2. \tag{2.1}$$

DEFINITION 2.1. *Given nonzero vectors x,y, and z, the generalized Rayleigh quotient (GRQ) is defined as*

$$\mathrm{GRQ}(x,y,z) = \frac{\sum_{i,j,k} a_{ijk} x_i y_j z_k}{\|x\|_2 \, \|y\|_2 \, \|z\|_2}.$$

Similar to the standard Rayleigh quotient, we define the generalized Rayleigh quotient $\mathrm{GRQ}(x,y,z)$ in a way that is invariant under a scaling of x, y, or z. It is easy to verify that if $\|x\|_2 = \|y\|_2 = \|z\|_2 = 1$, then $\lambda = \mathrm{GRQ}(x,y,z)$ minimizes (2.1), and the minimum value is

$$\sum_{i,j,k} a_{ijk}^2 - \mathrm{GRQ}(x,y,z)^2. \tag{2.2}$$

It thus follows that (1.1) is equivalent to the dual problem of maximizing GRQ:

$$\max \sum_{i,j,k} a_{ijk} x_i y_j z_k, \tag{2.3}$$

under the constraints that

$$\sum_i x_i^2 = \sum_j y_j^2 = \sum_k z_k^2 = 1. \tag{2.4}$$

We can write down the Lagrangian for the dual problem as

$$\sum_{i,j,k} a_{ijk} x_i y_j z_k - \frac{\mu_1}{2}\sum_i (x_i^2 - 1) - \frac{\mu_2}{2}\sum_j (y_j^2 - 1) - \frac{\mu_3}{2}\sum_k (z_k^2 - 1). \tag{2.5}$$

By differentiating (2.5), we obtain the following system at a critical point for each component x_i of x, y_j of y, and z_k of z:

$$\begin{cases} \sum_{j,k} a_{ijk} y_j z_k = \mu_1 x_i, \\ \sum_{i,k} a_{ijk} x_i z_k = \mu_2 y_j, \\ \sum_{i,j} a_{ijk} x_i y_j = \mu_3 z_k. \end{cases} \tag{2.6}$$

We now multiply x_i, y_j, and z_k to the first, the second and the third equations, and sum over i, j, and k respectively. This gives $\mu_1 = \mu_2 = \mu_3 = \sum_{i,j,k} a_{ijk}x_iy_jz_k = \mathrm{GRQ}(x,y,z)$. Let $\lambda = \mathrm{GRQ}(x,y,z)$, then we can rewrite the above system as

$$\begin{cases} \sum_{j,k} a_{ijk}y_jz_k = \lambda x_i, \\ \sum_{i,k} a_{ijk}x_iz_k = \lambda y_j, \\ \sum_{i,j} a_{ijk}x_iy_j = \lambda z_k, \\ \sum_{i,j,k} a_{ijk}x_iy_jz_k = \lambda. \end{cases} \tag{2.7}$$

Note that a nonzero solution to (2.7) automatically guarantees that $\|x\|_2 = \|y\|_2 = \|z\|_2 = 1$.

3. A special tensor decomposition. In this section, we study the following orthogonal tensor decomposition:

DEFINITION 3.1. *We say that a tensor $[a_{ijk}]$ is orthogonally decomposable if it can be written as the sum of F rank-one tensors $x_p \otimes y_p \otimes z_p$ $(p = 1, \ldots, F)$, such that $x_p \perp x_q$, $y_p \perp y_q$, and $z_p \perp z_q$ for $p \neq q$.*

It is not difficult to extend this definition to include higher order tensors. In general, orthogonal decompositions do not necessarily exist. However for the ICA problem, the fourth order cumulant tensors are orthogonally decomposable [4]. In the ICA literature, a Jacobi-type scheme for approximate diagonalization of multiple symmetric matrices has been proposed to compute this decomposition [4, 6]. Some numerical aspects of simultaneous matrix diagonalization can be found in [2, 12, 13]. In the following, we show that if a tensor is of order higher than two and is orthogonally decomposable, then the decomposition can be correctly computed by the incremental rank-one approximation algorithm[1]. Therefore, the orthogonal tensor decomposition problem can be reduced to the rank-one approximation problem. A consequence of this result is the uniqueness of orthogonal decomposition for tensors of order higher than two. This uniqueness of decomposition (in the special case of third order tensors) has been previously discovered in [28] using a different analysis.

Now consider the orthogonal decomposition of tensor $[a_{ijk}]$:

$$a_{ijk} = \sum_{p=1}^{F} x_{ip}y_{jp}z_{kp},$$

where we assume that $\sum_i x_{ip}x_{iq} = 0$, $\sum_j y_{jp}y_{jq} = 0$, and $\sum_k z_{kp}z_{kq} = 0$ for $p \neq q$. Note that for convenience, we use the notation v_{ip} to denote the i-th component of a vector v_p.

Consider the least squares rank-one approximation (1.1), for which we can always do an orthogonal transformation separately for each index i, j, and k without changing the least-squares error. Therefore, without loss of generality, we can assume that $x_{ip} = \alpha_p\delta_{ip}$, $y_{ip} = \beta_p\delta_{ip}$, and $z_{ip} = \gamma_p\delta_{ip}$, where

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

is the Kronecker delta symbol.

Let $\lambda_p = \alpha_p\beta_p\gamma_p$, then $a_{ijk} = \sum_p \lambda_p\delta_{ip}\delta_{jp}\delta_{kp}$. Let (x^*, y^*, z^*) be a nonzero solution of (1.1), and let $x' = x^*/\|x^*\|_2$, $y' = y^*/\|y^*\|_2$, and $z' = z^*/\|z^*\|_2$. Without loss of generality, we can assume that $|x'_1|$ achieves $\max(\|x'\|_\infty, \|y'\|_\infty, \|z'\|_\infty)$. By (2.7), we have $\lambda x'_1 = \lambda_1 y'_1 z'_1$, where $|\lambda|$ is given by (2.3).

Since $|\lambda|$ achieves the maximum in (2.3), $|\lambda| \geq |\lambda_1|$. Assume that $[a_{ijk}]$ is nonzero, then $\lambda \neq 0$ and $x'_1 \neq 0$. We thus obtain the inequality: $|x'_1| \leq |y'_1 z'_1|$. Note that by assumption, $1 \geq |x'_1| \geq \max(|y'_1|, |z'_1|)$. Therefore the inequality can only be achieved at $|x'_1| = |y'_1| = |z'_1| = 1$, which shows that $|\lambda| = |\lambda_1| = \max_i |\lambda_i|$ and $x' = y' = z' = e_1$, where e_1 is the vector with 1 in the first element and 0 elsewhere. Therefore an optimal rank-one approximation is given by $x^* \otimes y^* \otimes z^* = \lambda_1 e_1 \otimes e_1 \otimes e_1$.

Since $[a_{ijk}] - x^* \otimes y^* \otimes z^*$ is still orthogonally decomposable with rank $F-1$ by definition, it follows that by repeating the rank-one approximation algorithm F times, we obtain the decomposition $\sum_{p=1}^{F} x_{ip}y_{jp}z_{kp}$.

[1] In [27], the authors introduced a more general concept of orthogonal PCA-k decomposition. They argued that the decomposition can be computed using the incremental rank-one approximation procedure. However, their proof was faulty. See [21] for a detailed study.

Observe also that the uniqueness of the computational procedure implies that the orthogonal decomposition of $[a_{ijk}]$ is unique, and the same analysis is valid for tensors with order greater than 3. We can summarize the above results in the following theorem:

THEOREM 3.2. *If a tensor of order at least 3 is orthogonally decomposable, then this decomposition is unique, and the incremental rank-one approximation algorithm correctly computes it.*

System (2.7) is stable under a small perturbation if the Jacobi matrix J in (4.5) is not singular. Since our analysis is based on the equality $\lambda x_1' = \lambda_1 y_1' z_1'$ that comes from (2.7), the decomposition computed by the incremental rank-one approximation method is also stable under a small perturbation of $[a_{ijk}]$. This is not true for matrices since J becomes singular (see Section 4). Consequently, the singular vectors can be non-unique and unstable under perturbation when some of the singular values are not distinct.

4. Algorithms.

4.1. Generalized Rayleigh-Newton iteration. Newton's method can be applied to solve (2.7) for critical points. In order to derive the algorithm, we shall state an important property of the generalized Rayleigh quotient:

THEOREM 4.1. *Assume that (λ, x, y, z) is a nonzero solution to (2.7), then for small perturbations δx, δy, and δz, we have*

$$\mathrm{GRQ}(x+\delta x, y+\delta y, z+\delta z) = \lambda + O(\|\delta x\|_2^2 + \|\delta y\|_2^2 + \|\delta z\|_2^2).$$

Proof. We can assume without loss of generality that $x^T\delta x = y^T\delta y = z^T\delta z = 0$. This is because we can write δx as the sum of a component orthogonal to x and a component parallel to x (similarly for δy and δz). The component parallel to x does not modify GRQ; thus only the component orthogonal to x, which is at most as large as δx, contributes to the change of GRQ.

Now we have

$$\sum_{i,j,k} a_{ijk}\delta x_i y_j z_k = \lambda \sum_i x_i \delta x_i = 0.$$

Similarly $\sum_{i,j,k} a_{ijk} x_i \delta y_j z_k = \sum_{i,j,k} a_{ijk} x_i y_j \delta z_k = 0$. Therefore

$$\begin{aligned}
&\sum_{i,j,k} a_{ijk}(x_i+\delta x_i)(y_j+\delta y_j)(z_k+\delta z_k)\\
=&\sum_{i,j,k} a_{ijk}x_i y_j z_k + (\sum_{i,j,k} a_{ijk}\delta x_i y_j z_k + \sum_{i,j,k} a_{ijk} x_i \delta y_j z_k + \sum_{i,j,k} a_{ijk} x_i y_j \delta z_k)\\
&\quad + \sum_{i,j,k}(a_{ijk}\delta x_i \delta y_j z_k + a_{ijk}\delta x_i y_j \delta z_k + a_{ijk} x_i \delta y_j \delta z_k + a_{ijk}\delta x_i \delta y_j \delta z_k)\\
=&\lambda + 0 + O(\sum_{i,j}|\delta x_i \delta y_j| + \sum_{j,k}|\delta y_j \delta z_k| + \sum_{i,k}|\delta x_i \delta z_k| + \sum_{i,j,k}|\delta x_i \delta y_j \delta z_k|)\\
=&\lambda + O(\|\delta x\|_2\|\delta y\|_2 + \|\delta x\|_2\|\delta z\|_2 + \|\delta y\|_2\|\delta z\|_2). \qquad (4.1)
\end{aligned}$$

The last equality follows from the Schwartz inequality. We also note that

$$\|x+\delta x\|_2 = \sqrt{\|x\|_2^2 + 2x^T\delta x + \|\delta x\|_2^2} = \sqrt{1+0+\|\delta x\|_2^2} = 1 + O(\|\delta x\|_2^2).$$

Similarly, $\|y+\delta y\|_2 = 1 + O(\|\delta y\|_2^2)$, and $\|z+\delta z\|_2 = 1 + O(\|\delta z\|_2^2)$. Therefore

$$\|x+\delta x\|_2\ \|y+\delta y\|_2\ \|z+\delta z\|_2 = 1 + O(\|\delta x\|_2^2 + \|\delta y\|_2^2 + \|\delta z\|_2^2). \qquad (4.2)$$

Observe that (4.1) and (4.2) are respectively the numerator and the denominator in the definition of $\mathrm{GRQ}(x+\delta x, y+\delta y, z+\delta z)$, therefore

$$\begin{aligned}
\mathrm{GRQ}(x+\delta x, y+\delta y, z+\delta z) =& \frac{\lambda + O(\|\delta x\|_2\|\delta y\|_2 + \|\delta x\|_2\|\delta z\|_2 + \|\delta y\|_2\|\delta z\|_2)}{1 + O(\|\delta x\|_2^2 + \|\delta y\|_2^2 + \|\delta z\|_2^2)}\\
=&\lambda + O(\|\delta x\|_2\|\delta y\|_2 + \|\delta x\|_2\|\delta z\|_2 + \|\delta y\|_2\|\delta z\|_2) + O(\|\delta x\|_2^2 + \|\delta y\|_2^2 + \|\delta z\|_2^2)\\
=&\lambda + O(\|\delta x\|_2^2 + \|\delta y\|_2^2 + \|\delta z\|_2^2). \quad \square
\end{aligned}$$

This theorem is a generalization of the following well-known fact: for eigenvalue problems, the Rayleigh quotient is quadratically as accurate as the approximate eigenvector. Note that the theorem is also quite intuitive from the following non-rigorous argument: since a critical point of (2.3) optimizes $\mathrm{GRQ}(x,y,z)$, $\mathrm{GRQ}(x,y,z)$ has a zero gradient at a critical point.

Based on Theorem 4.1, a procedure similar to the Rayleigh quotient iteration (c.f. [14]) can be obtained. By Taylor expansion, we know that given λ, a linearization of (2.7) at (x,y,z) gives

$$\begin{aligned}
\sum_{i,j,k} a_{ijk}(y_j\delta z_k + \delta y_j z_k) - \lambda\delta x_i &= \lambda x_i - \sum_{i,j,k} a_{ijk} y_j z_k, \\
\sum_{i,j,k} a_{ijk}(x_i\delta z_k + \delta x_i z_k) - \lambda\delta y_j &= \lambda y_j - \sum_{i,j,k} a_{ijk} x_i z_k, \\
\sum_{i,j,k} a_{ijk}(x_i\delta y_j + \delta x_i y_j) - \lambda\delta z_k &= \lambda z_k - \sum_{i,j,k} a_{ijk} x_i y_j.
\end{aligned} \tag{4.3}$$

Now, let the (approximate) true solution be: $x^* = x + \delta x$, $y^* = y + \delta y$, and $z^* = z + \delta z$, we obtain the following linearizations:

$$\begin{aligned}
-\lambda x_i^* + \sum_{j,k} a_{ijk}(y_j z_k^* + z_k y_j^*) &= \sum_{j,k} a_{ijk} y_j z_k \\
-\lambda y_j^* + \sum_{i,k} a_{ijk}(x_i z_k^* + z_k x_i^*) &= \sum_{i,k} a_{ijk} x_i z_k \\
-\lambda z_k^* + \sum_{i,j} a_{ijk}(x_i y_j^* + y_j x_i^*) &= \sum_{i,j} a_{ijk} x_i y_j.
\end{aligned} \tag{4.4}$$

For r-th order tensors, the right hand side should be multiplied by $r-2$ (which is 1 in our case of $r=3$). The reason is that in the general case, 2-way product terms such as $y_j z_k$ in (2.7) are replaced by $(r-1)$-way product terms. The linearization of each of the $(r-1)$-way product terms contributes $r-1$ additive terms to the left hand side of the equation, which needs to be compensated by a multiple of $r-2$ on the right hand side for compatibility with (2.7). Note that for matrices (second order tensors), the right hand side is zero, and this is related to the singularity of the left hand side when $r=2$ (this point will be discussed later). In general, the above linear equation can be written in the matrix form as

$$J(\lambda, w) w^* = b(w), \tag{4.5}$$

where w denotes the vector concatenation of x, y, and z, and $b(w)$ is the right hand side of (4.4). Here

$$J(\lambda, w) = \begin{bmatrix} -\lambda I_{d_1\times d_1} & A_3 & A_2 \\ A_3^T & -\lambda I_{d_2\times d_2} & A_1 \\ A_2^T & A_1^T & -\lambda I_{d_3\times d_3} \end{bmatrix},$$

where $I_{d_1\times d_1}$, $I_{d_2\times d_2}$, and $I_{d_3\times d_3}$ are identity matrices corresponding to the x, y, and z directions respectively. The (i,j)-th element of A_3 is $\sum_k a_{ijk} z_k$ $(i=1,\ldots,d_1;\ j=1,\ldots,d_2)$; the (i,k)-th element of A_2 is $\sum_j a_{ijk} y_j$ $(i=1,\ldots,d_1;\ k=1,\ldots,d_3)$; and the (j,k)-th element of A_1 is $\sum_i a_{ijk} x_i$ $(j=1,\ldots,d_2;\ k=1,\ldots,d_3)$. To some extent, $J(\lambda,w)$ can be regarded as the Jacobian of (2.7) or the Hessian of (2.3).

Note that given x, y and z, λ can be taken as the generalized Rayleigh quotient $\lambda = \mathrm{GRQ}(x,y,z)$; and given λ, w can be updated by (4.5). We can alternate between these two steps, and that leads to the following algorithm:

ALGORITHM 4.1. (GRQ-Newton iteration)
Given initial estimate $w^0 = [x^0, y^0, z^0]^T$
for $p = 0, \ldots,$
  normalize w^p so that $\|x^p\|_2 = \|y^p\|_2 = \|z^p\|_2 = 1$
  let $\lambda^p = \mathrm{GRQ}(x^p, y^p, z^p)$
  solve $J(\lambda^p, w^p) w^{p+1} = b(w^p)$ for w^{p+1}
endfor

Note that (4.5) is used in Algorithm 4.1 since this particular formulation is directly comparable to the standard RQI (Rayleigh quotient iteration). However, our later convergence analysis will mostly rely on (4.3) which can be written as:

$$J(\lambda, w)\delta w = c(\lambda, w), \tag{4.6}$$

where $c(\lambda, w) = \lambda w - \frac{1}{r-2}b(w)$ and δw corresponds to the difference $w^{p+1} - w^p$ in Algorithm 4.1. Equation (4.6) can also be more suitable for iterative algorithms since problems introduced by the non-definiteness of J are alleviated (this point is discussed shortly).

As we have mentioned, there is a factor $r-2$ in $b(w)$ for the order r tensor formulation. Consequently, an important observation is that Algorithm 4.1 fails at $r = 2$ since $b(w) \equiv 0$. This case corresponds to the standard matrix singular value decomposition. A standard RQI replaces the definition of $b(w)$ by $b(w) = w$. The inconsistency of the algorithm at $r = 2$ is due to the singularity of $J(\lambda^*, w^*)$ at the critical point (λ^*, w^*).

For the order r tensor formulation of (4.5), let W_r be the $r \times r$ matrix consisted of all 1's except for -1's on the diagonal. Let $(\mu, [\alpha_1, \ldots, \alpha_r]^T)$ be an eigen-pair of W_r, and consider the vector $\tilde{w}^* = [\alpha_1 w_1^{*T}, \ldots, \alpha_r w_r^{*T}]^T$, where (λ^*, w^*) is a solution to (2.7), and $w^* = [w_1^{*T}, \ldots, w_r^{*T}]^T$. Because for all i: $\sum_{j \neq i} \alpha_j = (\mu + 1)\alpha_i$, by using the critical point equation (2.7) we obtain

$$J(\lambda^*, w^*)\tilde{w}^* = -\lambda^* \tilde{w}^* + (\mu + 1)\lambda^* \tilde{w}^*.$$

This implies that $\tilde{w}^*$ is an eigenvector of $J(\lambda^*, w^*)$ with an eigenvalue $\lambda^* \mu$. Since

$$W_r = v_r v_r^T - 2I_{r \times r},$$

where v_r is the column vector of dimension r that is consisted of all 1's, W_r has one eigenvalue of $r-2$ and the rest are -2. It follows that when $r = 2$, $J(\lambda^*, w^*)$ is always singular. However, when $r > 2$, such a conclusion cannot be drawn from this analysis. In fact, $J(\lambda^*, w^*)$ becomes singular only in degenerate cases, which rarely happens in practice. This is the fundamental difference between the case $r = 2$ and the case $r > 2$. Therefore unlike the ill-conditioned standard Rayleigh quotient iteration, matrix J is usually well-conditioned when $r > 2$. In this case, Algorithm 4.1 is consistent, and it locally achieves quadratic convergence as shown in Theorem 4.2.

From the above discussion, we can see that vectors $[\alpha_1 w_1^{*T}, \ldots, \alpha_r w_r^{*T}]^T$ span an invariant subspace of $J(\lambda^*, w^*)$. Although the matrix J is indefinite at (λ^*, w^*), the only positive eigenvalue of $J(\lambda^*, w^*)$ is $(r-2)\lambda^*$ with the eigenvector w^*. Because $c(\lambda^*, w^*)$ in (4.6) is orthogonal to w^*, J behaves like a definite matrix in a neighborhood of w^* for (4.3) if iterative methods are employed. Computationally, we do not have to factorize the matrix at each iteration if the direct factorization of a size $\sum_j m_j$ matrix ($O((\sum_j m_j)^3)$ operations) is costly. A (preconditioned) Krylov subspace method [14] may be employed in practice. With a fixed number of inner iterations, the computation only requires $O((\sum_j m_j)^2)$ operations. However, this method reduces the quadratic convergence shown in the following theorem to linear convergence.

THEOREM 4.2. *Let (λ^*, w^*) be a nonzero solution to (2.7). If $J(\lambda^*, w^*)$ is non-singular, then Algorithm 4.1 converges to (λ^*, w^*) quadratically in a neighborhood of (λ^*, w^*).*

Proof. Since $J(\lambda^*, w^*)w^* = b(w^*)$, it follows from the linearization formulations (4.3) and (4.5) that

$$\begin{aligned} 0 &= J(\lambda^*, w^*)w^* - b(w^*) \\ &= J(\lambda^*, w)w^* - b(w) + O(\|w^* - w\|_2^2). \end{aligned}$$

Since $\lambda^* = \mathrm{GRQ}(w^*)$ and, by Theorem 4.1, $\mathrm{GRQ}(w) = \lambda^* + O(\|w^* - w\|_2^2)$, we have $J(\lambda^*, w) = J(\mathrm{GRQ}(w), w) + O(\|w^* - w\|_2^2)$. Note that b does not depend on λ. Therefore

$$J(\mathrm{GRQ}(w), w)w^* - b(w) = O(\|w^* - w\|_2^2).$$

We have assumed that J is nonsingular at $J(\lambda^*, w^*)$, therefore

$$\|w^{p+1} - w^*\|_2 = \|J(\mathrm{GRQ}(w^p), w^p)^{-1} b(w^p) - w^*\|_2 = O(\|w^p - w^*\|_2^2).$$

Also note that if $w = [x^T, y^T, z^T]^T$ is in a neighborhood of $w^* = [x^{*T}, y^{*T}, z^{*T}]^T$, then

$$\|[\frac{x^T}{\|x\|_2}, \frac{y^T}{\|y\|_2}, \frac{z^T}{\|z\|_2}]^T - w^*\|_2 = O(\|w - w^*\|_2).$$

This implies that the normalization step in Algorithm 4.1 preserves the rate of convergence. □

Algorithm 4.1 can also be used to find a singular value and vector for a matrix (or an eigenvalue and vector for a symmetric matrix). Let B be a matrix of size $m_1 \times m_2$. We can regard it as a third order tensor of size $m_1 \times m_2 \times 1$. Since the third dimension is always unit 1 after normalization, the iteration only depends on vectors x and y corresponding to the first two dimensions of B. Matrix J can be written as

$$J(\lambda, [x, y]) = \begin{bmatrix} -\lambda I_{m_1 \times m_1} & B & By \\ B^T & -\lambda I_{m2 \times m_2} & B^T x \\ y^T B^T & x^T B & -\lambda \end{bmatrix}, \tag{4.7}$$

and b becomes

$$b([x, y]) = \begin{bmatrix} By \\ B^T x \\ x^T By \end{bmatrix}, \tag{4.8}$$

where both x and y have to be normalized: $\|x\|_2 = \|y\|_2 = 1$. If J is non-singular, which is extremely likely, then Algorithm 4.1 is consistent with good convergence properties. This algorithm can be compared to the standard Rayleigh quotient iteration, where the last row and the last column of J are omitted in the definition, and b is replaced by $w = [x^T, y^T]^T$. We note that in the Rayleigh quotient iteration, two choices can be made in the normalization step. The first one is to normalize x and y separately after each iteration. This case corresponds to an application of Algorithm 4.1 in that the generalized Rayleigh quotient is equivalent to the traditional Rayleigh quotient. The second choice is only to normalize $w = [x^T, y^T]^T$ as a whole ($\|x\|_2$ and $\|y\|_2$ can be different). This choice is more standard since it is obtained from the direct application of RQI to the equivalent eigenvalue problem of a SVD problem. However, in this case, the generalized Rayleigh quotient is not equivalent to the traditional Rayleigh quotient. The difference between these two normalization procedures is very small, as indicated by Example 5.2. Finally, it should be noticed that we can regard B as even higher order tensors, and that leads to different procedures.

4.2. Alternating least squares method. In practice, the most commonly used method for solving (1.1) is the alternating least squares algorithm (ALS), which was studied in [1, 20, 23]. An interesting property of this procedure is that it generalizes the power method for eigenvalue problems. Other generalizations are possible, such as the Jacobi procedure described in the next section. Although the convergence of this method was studied, the rate of convergence has not yet been analyzed in the literature. We show that by using the formulation developed in the previous sections, we can prove linear convergence of this method in a neighborhood of the optimal solution.

ALGORITHM 4.2. (ALS)
Given initial position $w^0 = [x^0, y^0, z^0]^T$
for $p = 0, \ldots,$
  for $i = 1, \ldots,$
    $x_i^{p+1} = \sum_{i,j,k} a_{ijk} y_j^p z_k^p$
  endfor
  for $j = 1, \ldots,$
    $y_j^{p+1} = \sum_{i,j,k} a_{ijk} x_i^{p+1} z_k^p$
  endfor
  for $k = 1, \ldots,$
    $z_k^{p+1} = \sum_{i,j,k} a_{ijk} x_i^{p+1} y_j^{p+1}$
  endfor
  normalize so that $\|x^{p+1}\|_2 = \|y^{p+1}\|_2 = \|z^{p+1}\|_2 = 1$
endfor

Algorithm 4.2 is derived by individually varying x (or y or z) while fixing the other two vectors in (1.1). It can be easily checked from (2.3) that the optimal x is proportional to $\sum_{i,j,k} a_{ijk} y_j z_k$. Due to the normalization step, λ does not need to appear in the algorithm.

Another way to look at this method is to regard it as a nonlinear version of the Gauss-Seidel iteration (c.f. [14]) applied to (2.7). Locally, after a linearization of the original problem, this algorithm can be regarded as an approximation of the block Gauss-Seidel iteration for solving the linear system (4.5). Although the Jacobian matrix $J(\lambda, w)$ can be indefinite, as we have discussed in Section 4.1, the right hand side of (4.3) lies approximately in the subspace where J is definite. Moreover, at each Gauss-Seidel iteration, δx (δy or δz) is approximately orthogonal to x (y or z), therefore each direction generated by the Gauss-Seidel iteration lies in the subspace where J is definite. This indicates that J can essentially be regarded as a definite operator. From this reasoning, we can obtain the following theorem:

THEOREM 4.3. *Assume that $(\lambda^*, x^*, y^*, z^*)$ maximizes (2.3) and $J(\lambda^*, w^*)$ is non-singular, where $w^* = [x^*, y^*, z^*]^T$, then Algorithm 4.2 converges to (λ^*, w^*) linearly in a neighborhood of (λ^*, w^*).*

Proof. When w^p is close to w^*, consider w^{p+1} obtained by Algorithm 4.2, u^{p+1} obtained by Algorithm 4.1, and v^{p+1} obtained by approximately solving (4.5) with the block Gauss Seidel iteration. That is, for each $k = 1, \ldots, 3$, solve the following equation for v_k^{p+1}:

$$\sum_{\ell \le k} J_{k,\ell}(\lambda^p, w^p) v_\ell^{p+1} + \sum_{\ell > k} J_{k,\ell}(\lambda^p, w^p) w_\ell^p = b_k(w^p), \tag{4.9}$$

where the subscript k (or ℓ) indicates one of the block component corresponding to x, y or z. As mentioned earlier, w^p is obtained from the nonlinear block Gauss-Seidel version of (4.9). Now notice that (4.5) is a linearization of (2.7) and $\lambda^p = \mathrm{GRQ}(w^p)$ is second order in $w^p - w^*$ from λ^*. Therefore at each step k, $v_k^{p+1} - w_k^{p+1}$ is second order in $w^p - w^*$ and $w^p - w^{p+1}$. That is, $\|v^{p+1} - w^{p+1}\| = O(\|w^p - w^*\|^2 + \|w^p - w^{p+1}\|^2)$. Also by Theorem 4.2 we have $\|u^{p+1} - w^*\| = O(\|w^p - w^*\|^2)$. Therefore we only need to show that $\|v^{p+1} - u^{p+1}\| \le \alpha \|w^p - u^{p+1}\|$ for some $\alpha < 1$ where α is independent of w^p.

By (4.6), we know that $J(\lambda^p, w^p)(u^{p+1} - w^p) = c(\lambda^p, w^p)$. Equation (4.9) implies that $v^{p+1} - w^p$ can be regarded as the approximation of the solution $\delta w = u^{p+1} - w^p$ to (4.6) after one block Gauss-Seidel iteration. Let $J^* = J(\lambda^*, w^*)$, then (4.6) can be replaced by $J^* \delta w = c(\lambda^p, w^p)$ with second order accuracy both for the exact solution $u^{p+1} - w^p$ and for the Gauss-Seidel approximation $v^{p+1} - w^p$. We thus only need to show that the Gauss-Seidel iteration for solving $J^* \delta w = c(\lambda^p, w^p)$ converges linearly with the starting point $\delta w = 0$. Furthermore, each component $c_k(\lambda^p, w^p)$ of the right hand side is orthogonal to w_k^p by the definition of λ^p. Therefore, if we let V denote the subspace spanned by $[x^{*T}, 0, 0]^T$, $[0, y^{*T}, 0]^T$ and $[0, 0, z^{*T}]^T$, then we can decompose $c(\lambda^p, w^p)$ as $\bar{c} + \Delta c$ such that $\bar{c} \in V^\perp$ and $\Delta c = O(\|\bar{c}\| \cdot \|w^p - w^*\|)$. Since Δc is a small perturbation which does not affect the linear convergence rate, we only need to show that the Gauss-Seidel iteration for solving $J^* \delta u = \bar{c}$ converges linearly with the starting point $\delta u = 0$.

It is easy to check that if $\bar{c} \in V^\perp$, then each new component δu_k generated from the block Gauss-Seidel iteration also lies in $V^\perp$. Therefore, the convergence only relies on the properties of J^* in the subspace $V^\perp$. Since (λ^*, w^*) maximizes (2.3) and J^* is non-singular, J^* has to be negative definite on $V^\perp$. The theorem follows from the well-known fact that the Gauss-Seidel iteration converges linearly for definite matrices [14]. □

There are many possible variants of Algorithm 4.2. One is to replace the Gauss-Seidel iteration by an iterative algorithm with a better convergence behavior (such as the non-linear version of successive over-relaxation or a Krylov subspace method [14]). Another variant is to vary two or more components (x or y or z) at the same time, instead of varying only one component. Note that the optimization of varying two-components can be obtained from an SVD algorithm. Varying more than two components at the same time leads to a divide-and-conquer approach. However, all of these algorithms have linear convergence rates. Therefore locally the GRQ-Newton iteration is more efficient computationally.

4.3. Jacobi Gauss-Newton procedure. For problems that need to be solved on a parallel computer, it is often desirable to use the Jacobi-version of Algorithm 4.2. There is also an interesting relationship between this Jacobi algorithm and the Gauss-Newton procedure for non-linear least squares problems. The Gauss-Newton method is an approximation to Newton's method with the property that the resulted linear system is always semidefinite. As we show in Section 5, this method (Algorithm 4.3) could be much slower

than Newton's method. However, it still has many useful applications, such as for certain engineering problems where the enhanced stability of the Gauss-Newton procedure may be desirable. In addition, the method does not require the Hessian matrix which may be expensive to compute.

ALGORITHM 4.3. (Jacobi Gauss-Newton iteration)
Given initial position $w^0 = [x^0, y^0, z^0]^T$
for $p = 0, \ldots,$
$x_i^{p+1} = \sum_{i,j,k} a_{ijk} y_j^p z_k^p$
$y_j^{p+1} = \sum_{i,j,k} a_{ijk} x_i^p z_k^p$
$z_k^{p+1} = \sum_{i,j,k} a_{ijk} x_i^p y_j^p$
normalize so that $\|x^{p+1}\|_2 = \|y^{p+1}\|_2 = \|z^{p+1}\|_2 = 1$
endfor

To derive Algorithm 4.3 as a Gauss-Newton procedure, we consider the formulation (2.1) where λ is still estimated as GRQ(w). Given this parameter λ, we linearize each term $\lambda(x_i + \delta x_i)(y_j + \delta y_j)(z_k + \delta z_k) - a_{ijk}$ as

$$\lambda x_i y_j z_k - a_{ijk} + \lambda(\delta x_i y_j z_k + x_i \delta y_j z_k + x_i y_j \delta z_k).$$

For the Gauss-Newton procedure, we work with the least squares formulation of this linearization as follows:

$$\min_{\delta x, \delta y, \delta z} \sum_{i,j,k} [\lambda x_i y_j z_k - a_{ijk} + \lambda(\delta x_i y_j z_k + x_i \delta y_j z_k + x_i y_j \delta z_k)]^2. \tag{4.10}$$

The above system is singular; therefore, we need to impose the following normalization constraints: $x^T \delta x = y^T \delta y = z^T \delta z = 0$. After some algebraic manipulations, we obtain the following solution to (4.10):

$$\begin{aligned}
\delta x_i &= \sum_{j,k} a_{ijk} y_j z_k / \lambda - x_i, \\
\delta y_j &= \sum_{i,k} a_{ijk} x_i z_k / \lambda - y_j, \\
\delta z_k &= \sum_{i,j} a_{ijk} x_i y_j / \lambda - z_k.
\end{aligned}$$

By normalizing $x_i + \delta x_i$, $y_j + \delta y_j$, and $z_k + \delta z_k$, we obtain an update rule that is equivalent to Algorithm 4.3. Similar to Algorithm 4.2, the Gauss-Newton iteration approximately solves (4.5), but it does not guarantee the convergence even locally. However in our experiments, the algorithm usually converges, although the observed rate of convergence is slightly slower than that of Algorithm 4.2. The main advantage of this method is its parallelizability.

4.4. Computational costs. In order to compare algorithms described in the previous sections, it is necessary to analyze their computational costs. To be more general, we analyze these algorithms in the case of r-th order tensor approximation. Assume that the dimension m_j for each side of the tensor satisfies $m_1 \geq m_2 \geq \cdots \geq m_r$. We also assume that $[a_{i_1,\ldots,i_r}]$ is not sparse and does not have any special structure that we can take advantage of. For simplicity, we only keep operation counts accurate up to the leading term in our analysis.

Given any vector w_j, we define the inner product of a r-th order tensor $A = [a_{i_1,\ldots,i_r}]$ and a vector w_j as a tensor of order $r-1$: $\langle a, w_j \rangle = [\sum_{i_j} a_{i_1,\ldots,i_r} w_{i_j,j}]$. The computation of this inner product requires $2\prod_j m_j$ operations. Therefore, in order to evaluate GRQ, the best way is to compute the inner product of A and w_j in the order from $j = 1$ to $j = r$. This computation requires $2\sum_j \prod_{k \geq j} m_k \approx 2\prod_j m_j$ operations.

We can now consider Algorithm 4.2. In the inner loop, the equation for the first component requires us to compute the inner product of A and w_j^p $(j > 1)$, with totally $2\prod_j m_j$ operations. Since the normalization of w_1^{p+1} requires $O(m_1)$ operations, the total cost should be $2\prod_j m_j$. For each component $k > 1$, we need to compute the inner product of A and w_j^{p+1} for $j = 1, \ldots, k-1$. Since the inner product of A and w_j^{p+1} for $j = 1, \ldots, k-2$ is available from the previous step, the overall computation at step k is

to take the inner product of $\langle A, [w_j^{p+1}]_{1:k-2}\rangle$ with w_{k-1}^{p+1} in $2\prod_{j\geq k-1} m_j$ operations, and then the inner product of $\langle A, [w_j^{p+1}]_{1:k-1}\rangle$ with $[w_{k-1}^{p}]_{k+1:r}$ in $2\prod_{j\geq k} m_j$ operations. The total operation count at step $k > 1$ is therefore $2\prod_{j\geq k-1} m_j + 2\prod_{j\geq k} m_j$. Summing over k, we need $4\prod_j m_j$ operations for each outer iteration. Since the procedure described for computing Algorithm 4.2 is still valid for evaluating inner products $\langle A, [w_1, \ldots, \hat{w}_k, \ldots, w_r]\rangle$ for $k = 1, \ldots, n$ simultaneously ($\hat{w}_k$ indicates that w_k is omitted in the inner product computation), it follows that Algorithm 4.3 also requires $4\prod_j m_j$ operations.

We now show that $6\prod_j m_j$ operations are needed to compute J and b in Algorithm 4.1. Note that b can be obtained from J in $o(\prod_j m_j)$ operations, therefore we only need to show that J can be obtained in $6\prod_j m_j$ operations. This can be done in two steps. In the first step, we compute $J_{1,\ell}$ (and therefore $J_{\ell,1}$), which requires m_1 times $4\prod_{j>1} m_j$ operations from the previous analysis. In the second step, we first compute $\langle A, w_1\rangle$, which requires $2\prod_j m_j$ operations; and then recursively compute the inner products of $\langle A, w_1\rangle$ with different combinations of $r-3$ vectors from $[w_j]_{2:n}$, which only takes $O(\prod_{j>1} m_j)$ operations. Therefore totally $6\prod_j m_j$ operations are needed to compute J and b. Since $O((\sum_j m_j)^3)$ operations are required to solve the linear system, each iteration in Algorithm 4.1 costs $6\prod_j m_j + O((\sum_j m_j)^3)$ operations. This will be comparable to the computational costs of the other two algorithms if $\prod_j m_j$ is at least of order m_1^3. Note that if an iterative method is employed, then $O((\sum_j m_j)^2)$ operations are required to solve the linear system approximately. In this case, however, we may only obtain a linear convergence rate.

The computational costs for each iteration of the algorithms are summarized in Table 4.1. Algorithm 4.1 is denoted by GRQI; Algorithm 4.2 is denoted by ALS; and Algorithm 4.3 is denoted by GN.

algorithm	GRQI	ALS	GN
flops/iteration	$6\prod_{j=1}^{r} m_j + O((\sum_{j=1}^{r} m_j)^3)$	$4\prod_{j=1}^{r} m_j$	$4\prod_{j=1}^{r} m_j$

TABLE 4.1
Comparison of computational costs in flops

5. Experimental results. Since the computational cost for each method has already been discussed, we only study the convergence behavior for these methods. We give two examples: the first example compares the convergence of the algorithms with synthetic and real multi-way datasets; the second example focuses on the matrix SVD problem. "Optimal solutions" in these examples are obtained numerically up to the machine precision (denoted by ϵ_{mach}). This has been done by first using ten iterations of ALS to find an approximate solution, and then using GRQI until the error is within the machine precision. Strictly speaking, the computed optimal solution w^* may not be exact. The tensors are also not necessarily orthogonally decomposable in the examples, and we only compute rank-one approximations. However, rank-F approximations can also be obtained by using the incremental algorithm we have mentioned. Although the scheme may not lead to an optimal low-rank approximation, a reasonable approximation may still be obtained. In all of the following experiments, we report the average performance of ten runs of the algorithms with ten different randomly generated initial vectors. In each of the ten runs, the same initial vector is used for all algorithms.

5.1. Example 1. The optimal solution is denoted as w^*, and the residue $R(w)$ of w is defined by (1.1). We do not report convergence results for GRQ since its behavior is similar to $R(w)$ (both are of the order $\|w - w^*\|_2^2$).

In Table 5.1, we consider a random low-rank $40 \times 30 \times 40$ tensor generated as the sum of 20 rank-one tensors — each rank-one tensor $x \otimes y \otimes z$ is generated with components of x, y and z uniformly distributed in $(0, 1)$. The 2-norm of this tensor is 569. For each of the ten runs, we start with a randomly generated initial vector, having an average residue of 448. The residue of w^* is 74. As we can see, a rank-one approximation reduces the residue by a factor of 6. The condition number of the Jacobian at w^* for GRQI is about 2, which explains why, in this case, both ALS and GN converge relatively quickly. Another interesting observation is that all three algorithms converge to the optimal solution from a randomly generated starting approximation. We believe this is related to the dominance of the optimal rank-one decomposition in this example.

Table 5.2 shows the results with a random $10 \times 15 \times 20 \times 20$ tensor. Each entry of the tensor is an independently generated Gaussian variable with mean 0 and standard deviation 1. The norm of the tensor is 246.00 and the optimal approximation has residue 245.68. Therefore, the optimal rank-one approximation

$\|w_k - w^*\|_2$				
p	1	2	3	4
GRQI	7.4×10^{-3}	1.0×10^{-7}	5.4×10^{-16}	ϵ_{mach}
ALS	1.2×10^{-2}	1.6×10^{-4}	4.1×10^{-7}	3.2×10^{-9}
GN	1.8×10^{-2}	6.8×10^{-4}	3.0×10^{-5}	1.4×10^{-6}
$R(w_k) - R(w^*)$				
p	1	2	3	4
GRQI	1.2×10^{-1}	2.8×10^{-11}	ϵ_{mach}	ϵ_{mach}
ALS	3.5×10^{-1}	6.2×10^{-5}	3.9×10^{-10}	5.0×10^{-13}
GN	7.4×10^{-1}	1.0×10^{-3}	1.9×10^{-6}	4.6×10^{-9}

TABLE 5.1
Positive low-rank random $40 \times 30 \times 40$ *tensor*

.

performs very poorly. Also in this case, we observe that GRQI and GN may converge to non-optimal approximations if we start with random approximations. Therefore we use starting approximations that are generated by randomly perturbing the computed optimal solution so that they are close to the optimal solution. The condition number of the Jacobian for GRQI at w^* is 58.4, which is relatively large. Thus both ALS and GN converge much more slowly than GRQI.

$\|w_k - w^*\|_2$				
p	1	2	3	4
GRQI	1.3×10^{-3}	3.8×10^{-6}	6.0×10^{-11}	2.4×10^{-15}
ALS	1.9×10^{-2}	1.0×10^{-2}	7.6×10^{-3}	5.8×10^{-3}
GN	2.4×10^{-2}	1.8×10^{-2}	1.5×10^{-2}	1.3×10^{-2}
$R(w_k) - R(w^*)$				
p	1	2	3	4
GRQI	1.3×10^{-7}	8.1×10^{-13}	ϵ_{mach}	ϵ_{mach}
ALS	4.6×10^{-5}	8.3×10^{-6}	2.8×10^{-6}	1.2×10^{-6}
GN	1.6×10^{-4}	9.7×10^{-5}	6.8×10^{-5}	5.3×10^{-5}

TABLE 5.2
Gaussian random $10 \times 15 \times 20 \times 20$ *tensor*

Table 5.3 uses the growth curve data of French girls from [19]. The dataset is a $12 \times 30 \times 8$ tensor, indicating 12 ages, 30 French girls, and 8 additional variables. The rank-one approximation performs very well for this real data, partially because all variables are positive. While the original tensor norm is 70808, the optimal solution reduces the residue to 8123. The condition number of Jacobian is 2.3, and we observe similar convergence behaviors as shown in Table 5.1.

5.2. Example 2. In this example, we compare three algorithms for computing a singular value of a matrix. GRQI is the method of treating the matrix as a three-dimensional tensor and applying Algorithm 4.1, which is described in Section 4.1. RQI denotes the standard Rayleigh quotient iteration applied to the equivalent eigenvalue problem. NRQI denotes the standard Rayleigh quotient with separate normalization of x and y after each iteration, as described in Section 4.1. We generate a random 40×50 matrix with entries uniformly distributed in $(0, 1)$. Table 5.4 reports the convergence of singular values (denoted by σ) and singular vectors (denoted by w) obtained by the algorithms after each iteration. After the 4th iteration, the estimated condition numbers are 2.4 for GRQI, and of the order 10^{16} for both NRQI and RQI. For this problem, GRQI not only is much better conditioned, but also seems to converge faster. From the table, we also see that all of the algorithms achieve the machine precision after four iterations.

We shall mention that it is possible to define the inverse iteration procedure for eigenvalue problems in a more well-conditioned way by introducing a constraint $x^T \delta x = 0$ (see, e.g. [29]). This leads to a more traditional Newton type method, which is not equivalent to our formulation. Our formulation has the

$\|w_k - w^*\|_2$				
p	1	2	3	4
GRQI	1.4×10^{-1}	3.7×10^{-4}	1.8×10^{-9}	5.0×10^{-16}
ALS	4.2×10^{-2}	6.7×10^{-4}	9.7×10^{-6}	1.4×10^{-7}
GN	1.1×10^{-1}	1.2×10^{-2}	1.3×10^{-3}	1.5×10^{-4}
$R(w_k) - R(w^*)$				
p	1	2	3	4
GRQI	5.0×10^{3}	1.0×10^{-1}	9.0×10^{-13}	ϵ_{mach}
ALS	5.8×10^{2}	1.5×10^{-1}	3.2×10^{-5}	6.9×10^{-9}
GN	3.1×10^{3}	4.9×10^{1}	6.1×10^{-1}	7.4×10^{-3}

TABLE 5.3
Growth curve data of French girls

$\|w_k - w^*\|_2$				
p	1	2	3	4
GRQI	6.0×10^{-3}	1.5×10^{-8}	4.0×10^{-15}	ϵ_{mach}
NRQI	7.2×10^{-2}	2.3×10^{-4}	2.6×10^{-11}	ϵ_{mach}
RQI	7.2×10^{-2}	2.3×10^{-4}	2.6×10^{-11}	ϵ_{mach}
$\|\sigma_k - \sigma^*\|$				
p	1	2	3	4
GRQI	7.4×10^{-4}	2.1×10^{-14}	ϵ_{mach}	ϵ_{mach}
NRQI	1.1×10^{-1}	1.8×10^{-6}	9.7×10^{-15}	ϵ_{mach}
RQI	1.1×10^{-1}	1.8×10^{-6}	8.8×10^{-15}	ϵ_{mach}

TABLE 5.4
40×50 *singular value problem*

advantage that the eigenvalue structure of the J matrix is better understood (see Section 4.1). However, the exact relationship between this method and the traditional Newton's method requires further investigation.

6. Conclusions. We have shown that if a tensor of order higher than 2 is orthogonally decomposable, then the decomposition is unique and can be computed by repeatedly applying a rank-one approximation algorithm. Furthermore, even if the tensor to be approximated is not orthogonally decomposable, incremental rank-one approximation can still be useful due to its simplicity.

Based on these observations, it is important to study numerical aspects of the rank-one tensor approximation problem. Specifically, we are able to prove a local linear convergence rate of the popular alternating least squares algorithm (ALS). In addition, based on a formulation that generalizes the Rayleigh quotient variational method for symmetric matrix eigenvalue problems, we are able to derive a generalized Rayleigh quotient-Newton iteration (GRQI), which locally has a quadratic convergence rate. For dense high order tensors, the computation is likely to be dominated by tensor-vector products. Therefore locally, this method can be more efficient than ALS even after the cost of matrix factorization is taken into consideration. We have also pointed out that ALS can be viewed as a nonlinear Gauss-Seidel procedure for approximately solving the linear system in GRQI. This relationship implies that more sophisticated iterative algorithms can be applied.

Many open problems still remain though. For example, general properties of high order tensor decompositions are still not well understood. It might also be interesting to study numerical methods that directly compute a rank-F tensor approximation, instead of using the incremental rank-one approximation procedure suggested in this paper.

Acknowledgements. We are grateful to the referees for many constructive comments and suggestions that led to substantial improvement of this paper. One referee also brought reference [21] into our attention.

REFERENCES

[1] J. T. Berge, J. D. Leeuw, and P. Kroonenberg, *Some additional results on principal components analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 52 (1987), pp. 183–191.

[2] A. Bunse-Gerstner, R. Byers, and V. Mehrmann, *Numerical methods for simultaneous diagonalization*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 927–949.

[3] J.-F. Cardoso, *Infomax and maximum likelihood for source separation*, IEEE Letters on Signal Processing, 4 (1997), pp. 112–114.

[4] J.-F. Cardoso and P. Comon, *Independent component analysis, a survey of some algebraic methods*, in IEEE International Symposium on Circuits and Systems, vol. 2, 1996, pp. 93–96.

[5] J.-F. Cardoso and B. Laheld, *Equivariant adaptive source separation*, IEEE Trans. on Signal Processing, 44 (1996), pp. 3017–3030.

[6] J.-F. Cardoso and A. Souloumiac, *Jacobi angles for simultaneous diagonalization*, SIAM J. Mat. Anal. Appl., 17 (1996), pp. 161–164.

[7] J. Carroll and J. Chang, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[8] P. Comon, *MA identification using fourth order cumulants*, Signal Processing, Eurasip, 26 (1992), pp. 381–388.

[9] ———, *Independent Component Analysis, a new concept ?*, Signal Processing, Elsevier, 36 (1994), pp. 287–314. Special issue on Higher-Order Statistics.

[10] P. Comon and B. Mourrain, *Decomposition of quantics in sums of powers of linear forms*, Signal Processing, Elsevier, 53 (1996), pp. 93–107. special issue on High-Order Statistics.

[11] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[12] B. Flury and W. Gautschi, *An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 169–184.

[13] B. Flury and B. Neuenschwander, *Simultaneous diagonalization algorithms with applications in multivariate statistics*, in Approximation and computation (West Lafayette, IN, 1993), vol. 119 of Internat. Ser. Numer. Math., Birkhäuser Boston, Boston, MA, 1994, pp. 179–205.

[14] G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[15] H. Harmann, *Modern factor analysis*, University of Chicago Press, USA, 1977.

[16] R. Harshman, *Foundations of the PARAFAC procedure: model and conditions for an 'explanatory' multi-mode factor analysis*, in UCLA Working Papers in phonetics, vol. 16, 1970, pp. 1–84.

[17] R. Harshman and M. Lundy, *PARAFAC: parallel factor analysis*, Comp. Stat. & Data Anal., 18 (1994), pp. 39–72.

[18] R. Henrion, *N-way principal component analysis. theory, algorithms and applications*, Chem. Intell. Lab. Syst., 25 (1994), pp. 1–23.

[19] J. Janssen, F. Marcotorchino, and J. Proth, eds., *International Symposium on Data Analysis, the Ins and Outs of Solving Real Problems*, New York, 1987, Plenum Press.

[20] H. Kiers, *An alternating least squares algorithm for PARAFAC2 and three-way dedicom*, Comp. Stat. & Data Anal., 16 (1993), pp. 103–118.

[21] T. G. Kolda, *Orthogonal tensor decompositions*, tech. report, Sandia National Laboraties, March 2000.

[22] P. Kroonenberg, *Three-mode principal component analysis: Theory and applications*, DSWO Press, Leiden, 1983.

[23] P. Kroonenberg and J. D. Leeuw, *Principal component analysis of three-mode data by means of alternating least square algorithms*, Psychometrika, 45 (1980), pp. 69–97.

[24] J. Kruskal, *Three way arrays: Rank and uniqueness of trilinear decomposition with applications to arithmetic complexity and statistics*, Linear Algebra Appl., 18 (1977), pp. 95–138.

[25] ———, *Rank, decomposition, and uniqueness for 3-way and n-way arrays*, in Multiway data analysis, 1989, pp. 7–18.

[26] D. D. Lee, U. Rokni, and H. Sompolinsky, *Algorithms for independent components analysis and higher order statistics*, in Advances in Neural Information Processing Systems 12, S. Solla, T. Leen, and K.-R. Müller, eds., MIT Press, 2000, pp. 491–497.

[27] D. Leibovici and R. Sabatier, *A singular value decomposition of a k-way array for a principal component analysis of multiway data, pta-k*, Linear Algebra Appl., 269 (1998), pp. 307–329.

[28] S. Leurgans, R. Ross, and R. Abel, *A decomposition for three-way arrays*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 1064–1083.

[29] G. Peters and J. Wilkinson, *Inverse iteration, ill-conditioned equations and Newton's method*, SIAM Review, 21 (1979), pp. 339–360.

[30] L. Tucker, *Some mathematical notes on three mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[31] H. Yang and S. Amari, *Adaptive on-line learning algorithms for blind separation — maximum entropy and minimum mutual information*, Neural Computation, 9 (1997), pp. 1457–1482.