Technical Report

TR-2004-012

Kronecker Product Approximation for Three-Dimensional Imaging Applications

by

MIsha Kilmer, James Nagy

Mathematics and Computer Science

EMORY UNIVERSITY

Kronecker Product Approximation for Three-Dimensional Imaging Applications

James G. Nagy* Misha E. Kilmer†

October 2, 2004

Abstract

Kronecker product and tensor decompositions are used to construct approximations of severely ill-conditioned matrices that arise in three-dimensional image processing applications. Computationally efficient methods to construct the approximations are developed by exploiting structure that is inherent in many image processing problems, such as those arising in microscopy and medical imaging. It is shown that the resulting approximations provide a general, powerful tool that can be used to improve efficiency of image reconstruction algorithms.

1 Introduction

Many image processing applications model the image formation process as an integral equation,

$$g(t) = \int_\Omega k(s,t) f(s) ds + e(t)\,, \tag{1}$$

or after discretization,

$$\mathbf{g} = K\mathbf{f} + \mathbf{e}\,, \tag{2}$$

where the vector $\mathbf{g}$ and matrix K are known, and the goal is to compute an approximation of the unknown vector $\mathbf{f}$. The vector $\mathbf{e}$, which represents errors or noise in the measured data $\mathbf{g}$, is generally not known. These equations are used, for example, to model the image formation process in image restoration, as well as the projection data collection process of medical imaging devices [1, 2, 9, 15, 23, 24, 25]. In general, $\mathbf{f}$ is a vector representation of the true (unknown) image. Thus, for $m \times n \times p$ (three-dimensional) images, $\mathbf{f}$ is a vector of length $N = mnp$. For simplicity of notation, we assume throughout this paper that $\mathbf{g}$ and $\mathbf{e}$ are also vectors of length N, and thus K is an $N \times N$ matrix. However, we emphasize that this restriction is just for notational convenience, and is not a requirement for any of the results and algorithms presented in this paper.

Equation (1), and its discrete analog, (2), are classical examples of ill-posed inverse problems whose properties have been well studied [7, 10, 12, 27]. In particular, the matrix K is typically severely ill-conditioned, making it very difficult to compute accurate approximations of $\mathbf{f}$. For example, if $\mathbf{e}$ is small, it is tempting to simply ignore it, and use standard approaches to solve $K\mathbf{f} = \mathbf{g}$. However, if K is ill-conditioned, the resulting *inverse solution*, $\mathbf{f}_{\text{inv}} = K^{-1}\mathbf{g}$, is likely to

*Department of Mathematics and Computer Science, Emory University. nagy@mathcs.emory.edu

†Department of Mathematic, Tufts University. Misha.Kilmer@tufts.edu This work supported by NSF grants NSF 0139968 and NSF 0208548

be a very poor approximation of the true vector $\mathbf{f}$. In order to compute a decent approximation of $\mathbf{f}$, some form of *regularization* must be employed. Such a solution may be expressed as

$$\mathbf{f}_{\text{reg}} = K_r^\dagger \mathbf{g}$$

where $K_r^\dagger$ can be thought of as a regularized pseudo-inverse of K.

A useful tool in constructing and analyzing regularization methods is the singular value decomposition (SVD). In most cases it is not computationally feasible to compute the SVD of a large scale matrix K. One exception is when K can be decomposed into a Kronecker product of smaller matrices. A Kronecker product of two matrices A, B is defined as:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{pmatrix}.$$

In image processing applications, such as deconvolution, image restoration, and image reconstruction, it is necessary to solve large scale inverse problems of the form (2), and although it may not be possible to decompose K into a Kronecker product form, it is often the case that K is highly structured. For example, in image restoration the kernel ($k(s,t)$ in (1)) or *point spread function* (PSF), often satisfies the property $k(s,t) = k(s-t)$; that is, it is spatially invariant. In many realistic problems, a theoretical model of the PSF is not known, and it must therefore be constructed experimentally from the imaging system by generating images of "point sources". What constitutes a point source depends on the application. For example, in atmospheric imaging, the point source can be a single bright star [13]. In microscopy, though, the point source is typically a fluorescent microsphere having a diameter which is about half the diffraction limit of the lens [6]. In either case, it is the PSF that can be used to construct the matrix K.

In medical imaging applications, such as computerized tomography (CT), positron emission tomography (PET) and single photon emission tomography (SPECT), the collected data, $\mathbf{g}$, is not an image, but rather a collection of projections [15, 23, 24, 25]. A theoretical model for these projections is often expressed in terms of the radon transform (i.e., line integrals through the object). The specific structure of the matrix K in the discrete model (2) depends on many things, such as the scanning geometry (e.g., parallel, fan, or cone beam), and whether attenuation and scatter, collimator blur, and noise have been incorporated into the model [2, 18]. Obtaining good quality images relies on accurately modeling these factors, but the tradeoff comes in computational costs. An interesting observation is that in many cases, the matrix $K^T K$ has a structure very similar to those found in image restoration [24]. Thus approximation techniques (e.g., for preconditioning) used in image restoration can often be used in image reconstruction [8].

In this paper we focus on three-dimensional problems where the matrix K (or $K^T K$) is defined by a known PSF (i.e., an $m \times n \times p$ array, P). Our aim is to describe an approach to factor a structured matrix, defined by a PSF, into a sum of Kronecker products:

$$K = \sum \sum \sum A_k \otimes B_j \otimes C_i \,. \tag{3}$$

The factorization is obtained from an orthogonal tensor decomposition of the PSF, P. In section 2 we discuss orthogonal tensor decompositions of three-dimensional arrays, and show how they can be used to efficiently factor K into a sum of of Kronecker products. In section 3 we show that the factorization (3) can be used to efficiently construct an approximation of the SVD of K, and how this approximate SVD can be used in regularization methods. Some numerical experiments are presented in section 4. Section 5 contains some concluding remarks.

2 Kronecker Factorizations

A general approach to compute factorizations of the form (3) was first presented by Van Loan and Pitsianis [26]. Efficient algorithms that exploit structured matrices arising in 2-dimensional image processing applications can be found in [16, 21]. The specific structure of the matrix depends on the imposed boundary conditions, and can involve Toeplitz, circulant and Hankel matrices. To motivate our approach for the 3-dimensional problem, and to see how the Toeplitz, circulant and Hankel matrices arise, we first review how a Kronecker factorization can be computed for 2-dimensional imaging problems.

2.1 Kronecker Factorization for 2-Dimensional Problems

Consider a 2D shift invariant problem involving images with $m \times n$ pixels. The PSF is the image of a point source, and thus can be represented as an $m \times n$ array, P. In this paper, we restrict ourselves to the case that the kernel k has compact support; this means that as we observe below, our matrices will have a banded structure in the Toeplitz case and an anti-banded structure in the Hankel case.

Suppose P is a rank-1 matrix; that is,

$$P = \mathbf{b}\mathbf{a}^T \,,$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors of length m and n, respectively. Since P is the image of a point source and we have assumed k has compact support, P contains the components of the Nth column of K, where the index N depends on the i, j location of the point source. Therefore K can be written as a Kronecker product:

$$K = A \otimes B \,,$$

where A is an $n \times n$ matrix and B is an $m \times m$ matrix, which are defined as follows. Suppose that the center of the PSF is at location (i, j) and p_{ij} is the value at the center of the PSF. Then, depending on the boundary conditions, we have [16, 21]:

- Zero boundary conditions imply A and B are Toeplitz matrices defined by $\mathbf{a}$ and $\mathbf{b}$, respectively. In particular,

$$A = \begin{pmatrix} a_j & \cdots & a_1 & & \\ \vdots & \ddots & & \ddots & \\ a_n & & \ddots & & a_1 \\ & \ddots & & \ddots & \vdots \\ & & a_n & \cdots & a_j \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_i & \cdots & b_1 & & \\ \vdots & \ddots & & \ddots & \\ b_m & & \ddots & & b_1 \\ & \ddots & & \ddots & \vdots \\ & & b_m & \cdots & b_i \end{pmatrix} .$$

- Similarly, for periodic boundary conditions, A and B are circulant matrices defined by $\mathbf{a}$ and $\mathbf{b}$, respectively.

- In the case of reflexive boundary conditions, A and B are Toepliz-plus-Hankel matrices defined

by $\mathbf{a}$ and $\mathbf{b}$, respectively. In particular,

$$A = \begin{pmatrix} a_j & \cdots & a_1 & & \\ \vdots & \ddots & & \ddots & \\ a_n & & \ddots & & a_1 \\ & \ddots & & \ddots & \vdots \\ & & a_n & \cdots & a_j \end{pmatrix} + \begin{pmatrix} a_{j+1} & \cdots & & a_n & \\ \vdots & & & & \\ a_n & & & & a_1 \\ & & & & \vdots \\ & & a_1 & \cdots & a_{j-1} \end{pmatrix},$$

and

$$B = \begin{pmatrix} b_i & \cdots & b_1 & & \\ \vdots & \ddots & & \ddots & \\ b_m & & \ddots & & b_1 \\ & \ddots & & \ddots & \vdots \\ & & b_m & \cdots & b_i \end{pmatrix} + \begin{pmatrix} b_{i+1} & \cdots & & b_m & \\ \vdots & & & & \\ b_m & & & & b_1 \\ & & & & \vdots \\ & & b_1 & \cdots & b_{i-1} \end{pmatrix}.$$

Notice that in each case, the indices i and j that specify the center of the PSF are important in determining how A and B are formed from $\mathbf{a}$ and $\mathbf{b}$.

If P is not rank-1, then we could compute a rank-one approximation of P, and construct A and B as described above. That is,

$$P \approx \mathbf{b}\mathbf{a}^T \quad \Rightarrow \quad K \approx A \otimes B\,.$$

Note that it is easy to get this approximation by simply using the dominant singular value and corresponding vectors of P (keep in mind that although K is a large $mn \times mn$ matrix, the dimension of P is only $m \times n$). In general, we can efficiently decompose K into a sum of Kronecker products as follows. Suppose $P = U\Delta V^T$ is the SVD of P. Let $\mathbf{u}_k$ and $\mathbf{v}_k$ denote the kth columns of U and V, respectively, and assume $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_r > \delta_{r+1} = \cdots = \delta_n = 0$ (i.e., the rank of P is r). Then,

$$P = \sum_{k=1}^{r} \delta_k \mathbf{u}_k \mathbf{v}_k^T \quad \Rightarrow \quad K = \sum_{k=1}^{r} A_k \otimes B_k\,,$$

where B_k is defined by $\mathbf{b}_k = \sqrt{\delta_k}\mathbf{u}_k$, the center of the PSF, and the imposed boundary condition, as described above. A_k is constructed similarly by $\mathbf{a}_k = \sqrt{\delta_k}\mathbf{v}_k$. With this representation, we see that $A_1 \otimes B_1$ gives, in some measure, a "best" Kronecker product approximation of K.

It is important to emphasize that this Kronecker product decomposition of K can be computed fairly efficiently. Specifically, if the images are $m \times n$ pixels (and thus K is $mn \times mn$), then it requires at most $O(m^3 + n^3)$ operations to compute the matrices A_k and B_k.

2.2 Kronecker Factorization for 3-Dimensional Problems

In the 3-dimensional case, images are represented as 3-dimensional arrays; that is, $m \times n \times p$ voxels. If we say P is a rank-1 tensor, we mean P can be expressed as

$$P = \mathbf{c} \circ \mathbf{b} \circ \mathbf{a}$$

where $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors of lengths m, n, and p, respectively. The notation $\circ$ is a generalization of outer product to higher dimensions. Specifically, the (i, j, k) entry of $\mathbf{c} \circ \mathbf{b} \circ \mathbf{a}$ is given by

$$(\mathbf{c} \circ \mathbf{b} \circ \mathbf{a})_{ijk} = c_i b_j a_k$$

In this situation, the matrix K can be represented as a Kronecker product:

$$K = A \otimes B \otimes C\,,$$

where A, B and C are matrices whose structures depend on the imposed boundary condition, analogous to 2-dimensional imaging problems.

We remark that one can mix boundary conditions. For example, if we use zero boundary conditions on the x and y directions, but reflexive boundary conditions in the z direction, then C and B are Toeplitz matrices, whereas A is a Toeplitz-plus-Hankel matrix.

In the case when P is not rank-1, obtaining the best rank one approximation is a bit more difficult. In particular, there is no known tensor extension of the Eckart-Young SVD approximation theorem [20]. However, attempts have been made to generalize the SVD to high order tensors, and, moreover, approaches have been proposed to compute best rank one tensor approximations; see, for example [4, 5, 29]. If P is an $m \times n \times p$ array, our first attempt might be to try to find a scalars δ_k and vectors $\mathbf{u}_k$, $\mathbf{v}_k$ and $\mathbf{w}_k$ such that

$$P = \sum_{k=1}^{r} \delta_k \mathbf{u}_k \circ \mathbf{v}_k \circ \mathbf{w}_k\,,$$

but this turns out not to be the most convenient way to represent the tensor decomposition, because it is not always possible to orthogonally "diagonalize" a 3-dimensional tensor. However, an orthogonal decomposition can be constructed from the *high order singular value decomposition* (HOSVD) suggested by de Lathauwer, de Moor and Vandewalle [4]. The HOSVD has the form:

$$P = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \delta_{ijk} \mathbf{u}_i \circ \mathbf{v}_j \circ \mathbf{w}_k\,. \tag{4}$$

Using (4), we can construct a factorization of the matrix having the form:

$$K = \sum \sum \sum_{\delta_{ijk} \neq 0} A_k \otimes B_j \otimes C_i\,,$$

where A_k is defined by the vector $\mathbf{a}_k = \delta_{ijk}^{1/3} \mathbf{w}_k$, B_j is defined by $\mathbf{b}_j = \delta_{ijk}^{1/3} \mathbf{v}_j$, and C_i is defined by $\mathbf{c}_i = \delta_{ijk}^{1/3} \mathbf{u}_i$.

The HOSVD decomposition of a 3-dimensional array P, given by equation (4), is computed as follows (we use Matlab colon notation to specify blocks of columns, rows, etc.):

- Suppose P is an $m \times n \times p$ array. To describe the procedure from [4] for computing the HOSVD, we need to define certain matrices by "unfolding" P. In particular, we consider:

$$\begin{aligned}
P_{(1)} = \texttt{unfold}(P,1) &= \left[\; P(:,1,:) \quad P(:,2,:) \quad \cdots \quad P(:,n,:) \;\right] \in \mathbb{R}^{m \times pn} \\
P_{(2)} = \texttt{unfold}(P,2) &= \left[\; P(:,:,1)^T \quad P(:,:,2)^T \quad \cdots \quad P(:,:,p)^T \;\right] \in \mathbb{R}^{n \times pm} \\
P_{(3)} = \texttt{unfold}(P,3) &= \left[\; P(1,:,:)^T \quad P(2,:,:)^T \quad \cdots \quad P(m,:,:)^T \;\right] \in \mathbb{R}^{p \times nm}
\end{aligned}$$

We will also need to "fold" matrices into 3D arrays, for example:

$$P = \texttt{fold}(P_{(1)},1) = \texttt{fold}(P_{(2)},2) = \texttt{fold}(P_{(3)},3)\,.$$

- The next step is to compute the compact SVDs of each of the matrices $P_{(1)}$, $P_{(2)}$ and $P_{(3)}$:

$$\begin{aligned} P_{(1)} &= U\Delta_{(1)}Q_{(1)}^T \\ P_{(2)} &= V\Delta_{(2)}Q_{(2)}^T \\ P_{(3)} &= W\Delta_{(3)}Q_{(3)}^T, \end{aligned}$$

  where U, V, W are square orthogonal matrices and the $\Delta_{(i)}$ are square and diagonal. Then the vectors $\mathbf{u}_k$ in the decomposition (4) are the columns of the matrix U (i.e., the left singular vectors of $P_{(1)}$). Similarly, the vectors $\mathbf{v}_k$ are the columns of V (the left singular vectors of $P_{(2)}$), and the vectors $\mathbf{w}_k$ are the columns of W (the left singular vectors of $P_{(3)}$).

- The scalars, δ_{ijk} are entries in the $m \times n \times p$ array:

$$\Delta = \texttt{fold}\left(U^T P_{(1)}(V \otimes W), 1\right)$$

We note that if P is an $m \times n \times 1$ array (i.e., a matrix) then the HOSVD produces the SVD of P. To see this, observe:

- $P_{(1)} = P$, so the left singular vectors of $P_{(1)}$ are the left singular vectors of P.
- $P_{(2)} = P^T$, so the left singular vectors of $P_{(2)}$ are the right singular vectors of P.
- $P_{(3)}$ is a $1 \times mn$ matrix, so $W = 1$.
- $\Delta = \texttt{fold}\left(U^T P_{(1)}(V \otimes 1), 1\right) = U^T P V$, and so $P = U\Delta V^T$.

We refer the interested reader to [4] for more details on the properties of the HOSVD, and the derivation of the algorithm outlined above.

3 Computations with Kronecker Products

Now that we have a factorization of K into a sum of Kronecker products, we return to the original problem of computing solutions to the discrete ill-posed problem (2). As stated in the introduction, it is very difficult to compute accurate approximations of $\mathbf{f}$, and regularization methods must be employed to reduce sensitivity of the solution to noise in the data. A class of regularization methods, called filtering, can be defined using the SVD[1] of the matrix K [12, 27]. In particular, suppose $K = U\Sigma V^T$, where where U and V are orthogonal matrices and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)$. A naive approach to approximate $\mathbf{f}$ is to compute the *inverse solution*:

$$\hat{\mathbf{f}} = K^{-1}\mathbf{g} = V\Sigma^{-1}U^T\mathbf{g} = \sum_{i=1}^{N} \frac{\mathbf{u}_i^T\mathbf{g}}{\sigma_i}\mathbf{v}_i = \mathbf{f} + \sum_{i=1}^{N} \frac{\mathbf{u}_i^T\mathbf{e}}{\sigma_i}\mathbf{v}_i\,,$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the ith columns of U and V, respectively. Note that division by small singular values amplifies the noise. Filtering methods dampen these effects by computing a solution of the form:

$$\mathbf{f}_{\text{filt}} = \sum_{i=1}^{N} \phi_i \frac{\mathbf{u}_i^T\mathbf{g}}{\sigma_i}\mathbf{v}_i\,,$$

[1]The SVD of K should not be confused with the HOSVD of P nor the SVDs of the $P_{(i)}$ in the previous section.

where

$$\phi \approx \begin{cases} 1 & \text{for large } \sigma_i \\ 0 & \text{for small } \sigma_i \end{cases} .$$

For example, the truncated SVD (TSVD) method chooses a truncation tolerance, τ, and defines the filter factors as:

$$\phi = \begin{cases} 1 & \text{if } \sigma_i \geq \tau \\ 0 & \text{if } \sigma_i < \tau \end{cases} . \tag{5}$$

Note that we could choose τ to be a specific singular value, such as $\tau = \sigma_k$. In this case, we refer to k as the truncation index, and we write the TSVD solution as

$$\mathbf{f}_{\text{tsvd}} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T \mathbf{g}}{\sigma_i} \mathbf{v}_i \, .$$

Choosing the truncation tolerance, τ, is a nontrivial topic; some approaches that can be used include the discrepancy principle, generalized cross validation method and the L-curve [7, 12, 27].

We remark that other well known regularization methods, such as Tikhonov regularization, can be expressed in terms of an SVD filtering approach. Note that in our application, the matrix K is too large to explicitly compute an SVD. An exception, however, occurs when the PSF, P, is rank-1. As we have seen, in this case, we can obtain the factorization

$$K = A \otimes B \otimes C \, ,$$

where A is a $p \times p$ matrix, B is an $n \times n$ matrix, and C is an $m \times m$ matrix, by looking at the three-dimensional PSF. It is possible to explicitly compute SVDs of A, B and C, and use properties of Kronecker products [14] to obtain

$$\begin{aligned} K &= A \otimes B \otimes C \\ &= (U_a \Sigma_a V_a^T) \otimes (U_b \Sigma_b V_b^T) \otimes (U_c \Sigma_c V_c^T) \\ &= (U_a \otimes U_b \otimes U_c)(\Sigma_a \otimes \Sigma_b \otimes \Sigma_c)(V_a \otimes V_b \otimes V_c)^T \\ &= U \Sigma V^T \, . \end{aligned}$$

When implementing filtering algorithms, such as the TSVD, we do not need to explicitly form U, Σ and V, but instead we can work with the smaller factors corresponding to A, B and C.

A problem arises when P is not rank-1, in which case K can only be factored into a sum of Kronecker products. As described above, we can easily handle a single Kronecker product, but unfortunately the SVD of a sum of matrices is not (in general) the sum of its SVDs. To overcome this difficulty, we use an approach advocated in [16] to construct an approximate SVD decomposition. Using the approach outlined in section 2, factor K as

$$K = \sum \sum_{\delta_{ijk} \neq 0} \sum A_k \otimes B_j \otimes C_i \, .$$

Suppose $\delta_{i_0 j_0 k_0} = \max \delta_{ijk}$, and compute the SVDs

$$A_{k_0} = U_a \Sigma_a V_a^T \, , \quad B_{j_0} = U_b \Sigma_b V_b^T \, , \quad C_{i_0} = U_c \Sigma_c V_c^T \, .$$

Then an approximate SVD, $K \approx U \Sigma V^T$, is given by

$$U = U_a \otimes U_b \otimes U_c \, , \quad V = V_a \otimes V_b \otimes V_c \, , \quad \Sigma = \text{diag}\left(U^T K V\right) \, .$$

By exploiting properties of Kronecker products, the computation of Σ is relatively inexpensive, provided there are only a few $\delta_{ijk} \neq 0$. That is, since $U^T K V = \sum\sum_{\delta_{ijk}\neq 0}\sum U_a^T A_k V_a \otimes U_b^T B_j V_b \otimes U_c^T C_i V_b$ and we only need the diagonal elements of this matrix, Σ can be computed in $O(mnp)$ arithmetic operations. In our applications, we have found that we need only use terms in the decomposition corresponding to $|\delta_{ijk}| \geq \gamma$, where γ is some prescribed tolerance.

Using this approximate SVD in filtering methods, we can compute approximate solutions to (2). Of course, it may be possible to compute better solutions with a different regularization method that does not require an SVD. For example, we might consider using an iterative algorithm, such as the conjugate gradient method, where regularization is enforced by truncating the iterations (i.e., early termination) [7, 12, 27]. Preconditioning (which amounts to constructing an approximation $M \approx K$ and applying the iterative method to $M^{-1}K$) can be used with these methods to accelerate convergence, but care must be taken for ill-posed problems since linear systems of the form $M\mathbf{z} = \mathbf{w}$ must be solved at each iteration. If M is a good approximation of K, then M will be ill-conditioned, and therefore inverting M may cause extensive noise amplification during the early iterations.

One preconditioning approach for ill-posed problems, proposed in [11], uses a TSVD like approach with an approximate SVD of K. Specifically, define a preconditioner $M_\tau = U\Sigma_\tau V^T$, where $K \approx U\Sigma V^T$, and $\Sigma_\tau = \text{diag}(\sigma_1, \ldots, \sigma_k, 1, \ldots 1)$. To understand why this approach works, we need to understand a little about truncated iteration regularization. It can be shown (see [7, 12, 19, 27]) that the early iterations filter out components of the solution corresponding to the small singular values of the matrix. That is, the early iterations tend to reconstruct mostly the good part of the solution, and noise components of the solution are filtered out. It is this part of the iteration that we want to accelerate. At some point, though, the noise components start to be reconstructed, and the iterations begin to be corrupted with noise; this part of the iteration we do not want to accelerate.

For a preconditioned iterative method, it is the singular values of the preconditioned system $M^{-1}K$ that we must consider. By clustering all of the singular values around one, we no longer have the information to distinguish between the signal and noise subspaces. However, if we use the preconditioner M_τ with a good approximation of the SVD, then the preconditioned system has the form:

$$M_\tau^{-1} K \approx V \Delta V^T \,,$$

where $\Delta \approx \text{diag}(1, \ldots, 1, \sigma_{k+1}, \ldots, \sigma_n)$. In this case, the large singular values (i.e., those corresponding to the signal subspace) are clustered at one, and are well separated from the small singular values (those corresponding to the noise subspace). Iterative algorithms, especially conjugate gradient type methods, tend to converge very quickly in this situation.

4 Numerical Experiments

In this section we present some numerical experiments that illustrate the effectiveness of our Kronecker product and SVD approximations on two test problems. In each example a known true image is artificially blurred with a known PSF, and 1% Gaussian white noise is added to the blurred image. That is, for $\mathbf{f}$ as in (2), we determined $\mathbf{e}$ so that

$$\frac{\|\mathbf{e}\|_2}{\|K\mathbf{f}\|_2} = .01.$$

When describing the PSF, we assume spatial invariance, and use the notation

$$k(x, y, z) = k(s_x - t_x, s_y - t_y, s_z - t_z) = k(s - t)\,.$$

All experiments were done using MATLAB 7.0.

4.1 Example 1. Medical Imaging

In this example we simulate blurring effects caused by partial volume averaging in spiral CT [28], where it has been shown that the PSF can be approximated well by a three-dimensional Gaussian. We note that a Gaussian is separable, and thus our approximation techniques are ideally suited for this kind of problem. To construct such a PSF, we averaged three different Gaussians,

$$k(x,y,z) = (k_1(x,y,z) + k_2(x,y,z) + k_3(x,y,z))/3\,, \tag{6}$$

where

$$k_i(x,y,z) = \frac{1}{\sqrt{(2\pi)^3\sigma_i^3}} e^{-(x^2+y^2+z^2)/2\sigma_i^2}\,,$$

and each σ_i was chosen randomly to satisfy $1 \le \sigma_i \le 2$. The true image, which is included in MATLAB, consists of 27 image slices, each 128×128, of a human head. Fig. 1 shows slices of the true image, and Fig. 2 shows the corresponding noisy blurred slices.

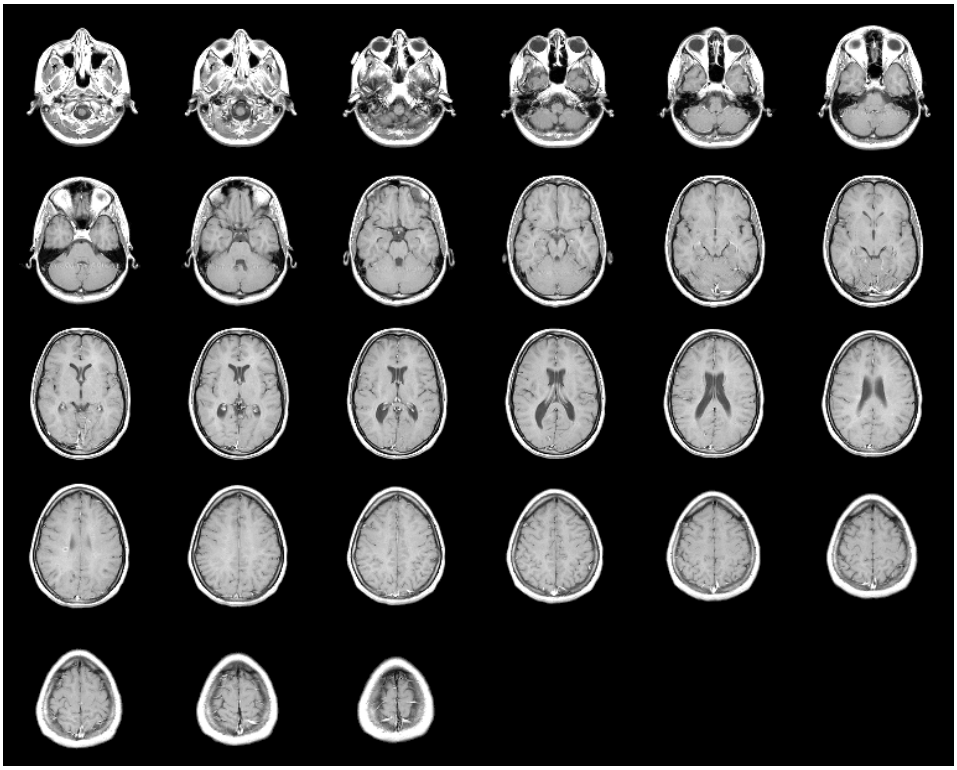

Figure 1: True image for example 1.

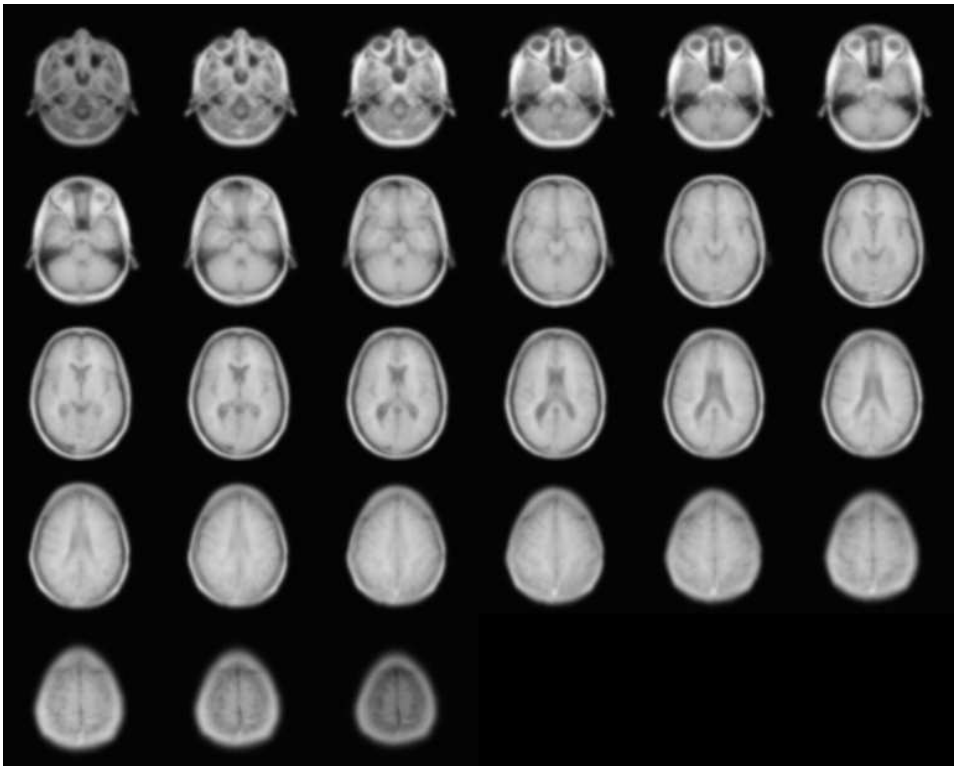

Figure 2: Noisy blurred image for example 1.

The PSF in this example cannot be represented as a single Kronecker product factorization, $A \otimes B \otimes C$. This can be seen in Fig. 3, which shows a plot of the largest 20 δ_{ijk} values of the HOSVD of the PSF. Recall that our algorithm requires that we first choose the number of terms

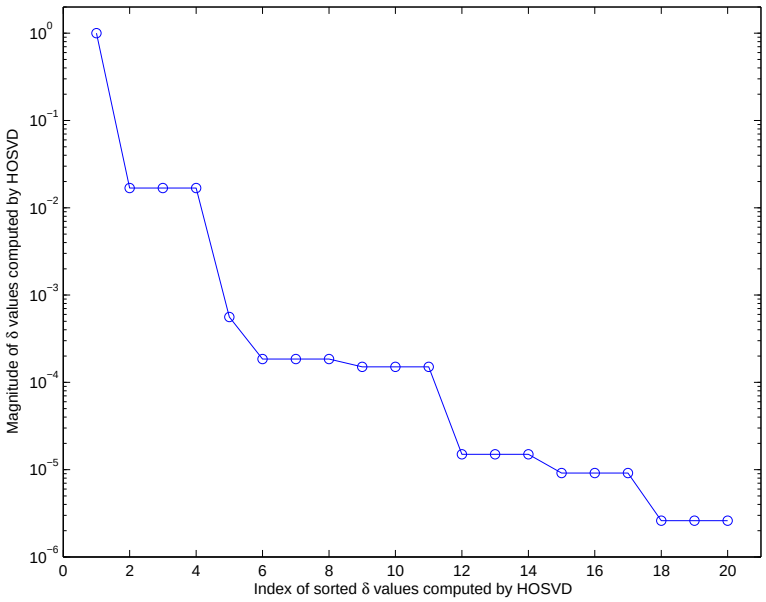

Figure 3: Largest 20 δ_{ijk} values computed by the HOSVD method (see equation (4)) for the PSF in example 1.

in our HOSVD expansion to use to get an approximate factorization K as a sum of Kronecker products. From that factorization, we use the approximate (or exact, if only one term is used) SVD to build the TSVD regularized solution using a truncation tolerance of τ. The computed reconstruction using one term in the Kronecker product SVD approximation scheme with $\tau = 0.05$ is shown in Fig. 4. This result clearly shows that, for this example, one term of the Kronecker product SVD approximation works very well in restoring the image using the TSVD methods. Additional terms in the triple sum were taken in the approximation, but the computed restorations remained essentially the same. This is likely due to the fact that the largest δ_{ijk} value (see Fig. 3) is well separated from the others, and therefore very little information is added to the dominant singular values when more terms are taken in the approximation.

Although the quality of the TSVD restorations are encouraging, as mentioned in section 3, better reconstructions might be computed with algorithms that use the original matrix, K. Iterative methods can be implemented efficiently, given the spatially invariant structure of the kernel from which K is derived, using fast Fourier transforms (FFT) for the matrix-vector multiplication. There are many choices for iterative methods, including conjugate gradient type algorithms. For our tests, we use the conjugate gradient method for least squares problems (CGLS) [3], and its preconditioned version (PCGLS). For a preconditioner, we use the approximate SVD discussed in section 3 with one term in the Kronecker product approximation of K. As with the TSVD results reported above, we use the truncation tolerance $\tau = 0.05$. The convergence history (i.e., a plot of the relative errors between the exact and computed solution at each iteration) is shown in Fig. 5. The dashed horizontal line on this plot shows the error level of the best TSVD reconstruction computed earlier. Although not dramatic for this particular example, the preconditioner is effective at reducing the

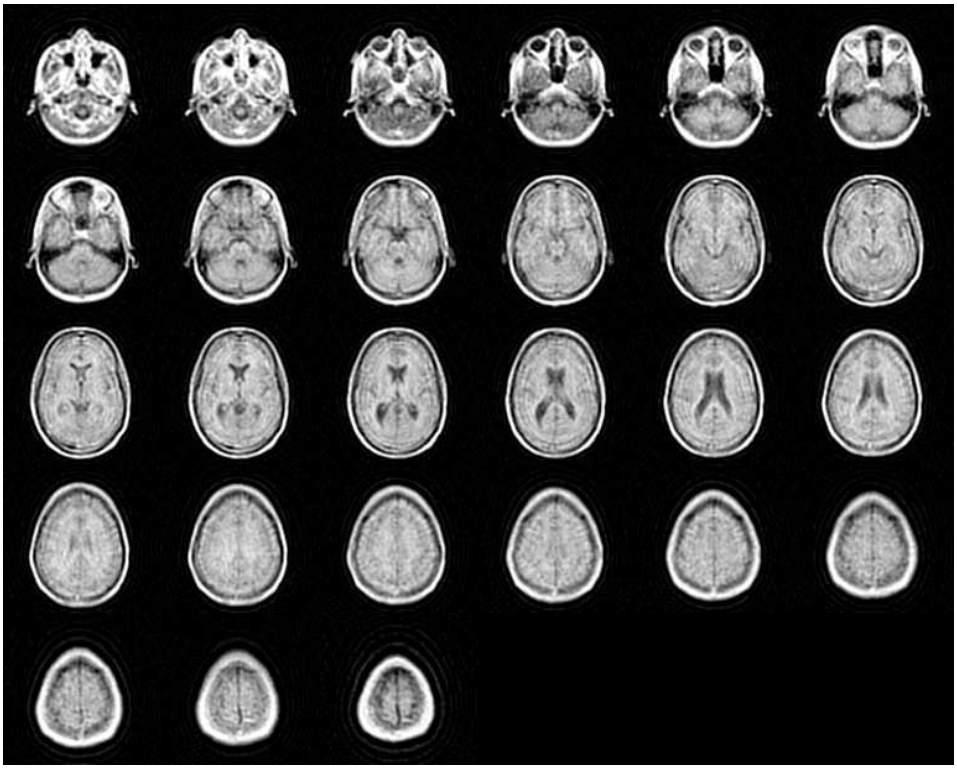

Figure 4: TSVD computed restoration (for example 1) using 1 term in the Kronecker product SVD approximation scheme.

number of iterations[2] needed to compute very good restorations. Fig. 6 shows the result after 6 iterations of PCGLS. We remark that the cost per iteration of PCGLS is about 1.7 times that of CGLS, which implies that 6 iterations of PCGLS is about the same cost as 10 iterations of CGLS. Therefore, although the difference is not especially dramatic, the gain in convergence speed offsets the extra work required by the preconditioner.

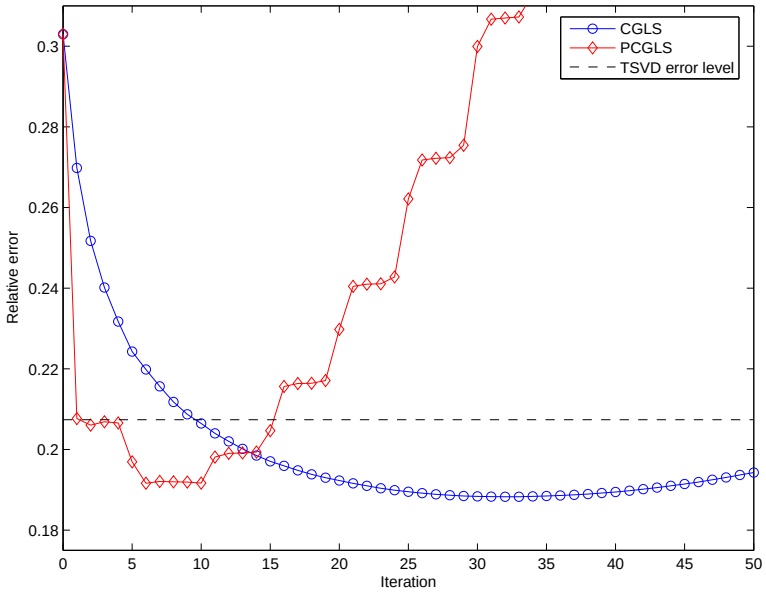

Figure 5: Convergence history of CGLS and PCGLS for example 1.

[2]Of course, in practice the relative error is not available and one must decide when it is time to stop iterating. The stopping parameter is a regularization parameter, just as τ is a regularization parameter for TSVD, and some of the methods mentioned earlier for choosing the TSVD parameter are applicable for choosing the stopping parameter.

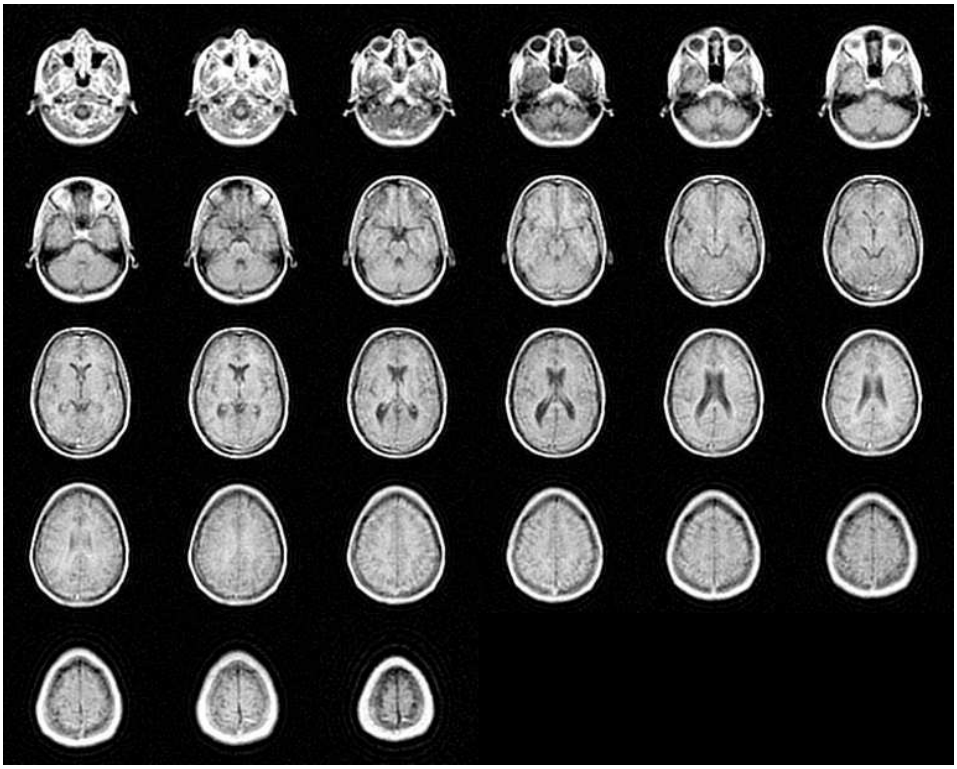

Figure 6: Slices of the restored image (for example 1) after 6 iterations of PCGLS.

4.2 Example 2. Microscopy

In this example we simulate blurring effects caused by optical limits in 3D microscopy. In particular, we consider a simulation of two-photon microscopy [6], where it is known that the PSF has a parametric form given by

$$k(x, y, z) = \frac{e^{-4r^2(x,y)/\omega^2(z)}}{\omega^4(z)},$$

where

$$\omega^2(z) = \omega_0^2 \left(1 + \left(\frac{z}{z_R}\right)^2\right), \quad \text{and} \quad r^2(x, y) = x^2 + y^2 .$$

Typical values for the constants are $\omega_0 = 0.25\mu$m and $\omega_0 \leq z_R \leq 10\omega_0$. In the simulation described in this example, we used $\omega_0 = 0.25$ and $z_R = 1$. The true image, which was obtained from `http://www.wadsworth.org/spider_3d/page_two.html`, consists of a stack of 20 images, each 128×103, of a dendrite. Fig. 7 shows the true image, and Fig. 8 shows the noisy blurred image.

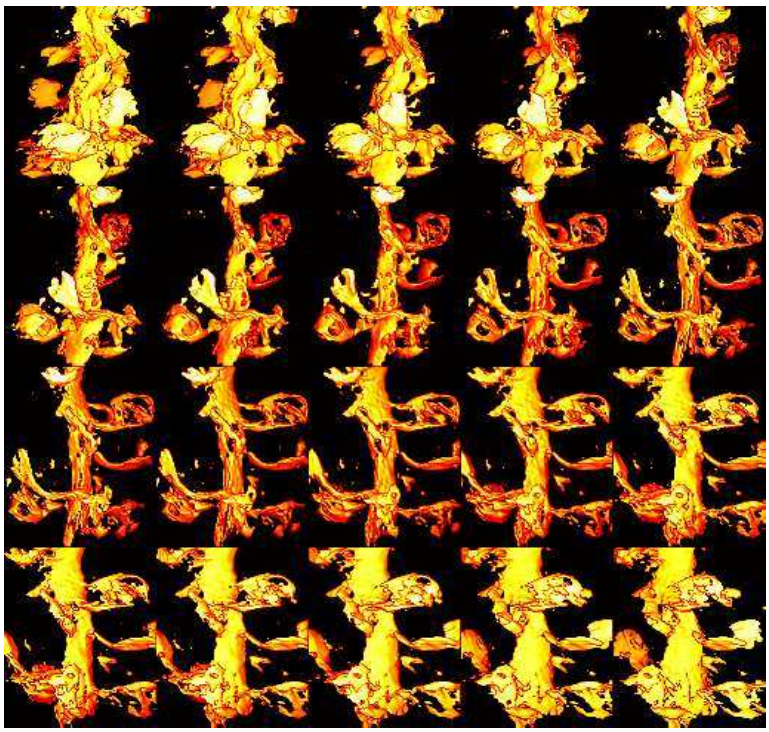

Figure 7: True image for example 2.

As with the previous example, the PSF in this example cannot be represented as a single Kronecker product factorization, $A \otimes B \otimes C$. This can be seen in Fig. 3, which shows a plot of the largest 50 δ_{ijk} values of the HOSVD of the PSF. In fact, in this example, the δ_{ijk} values decay more slowly, suggesting that it may be necessary to use more terms in the Kronecker product SVD approximation. To illustrate that this is indeed the case, we computed simple reconstructions with a TSVD filter, with a truncation tolerance $\tau = 0.01$, using 1 and 3 terms in the Kronecker product SVD approximation scheme. The results are shown in Figs. 10 and 11, respectively.

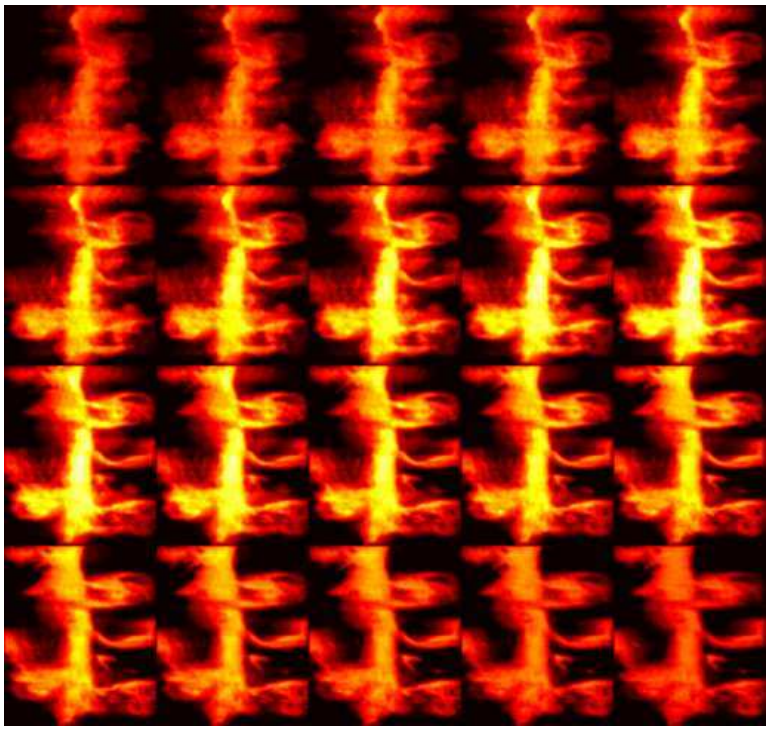

Figure 8: Noisy blurred image for example 2.

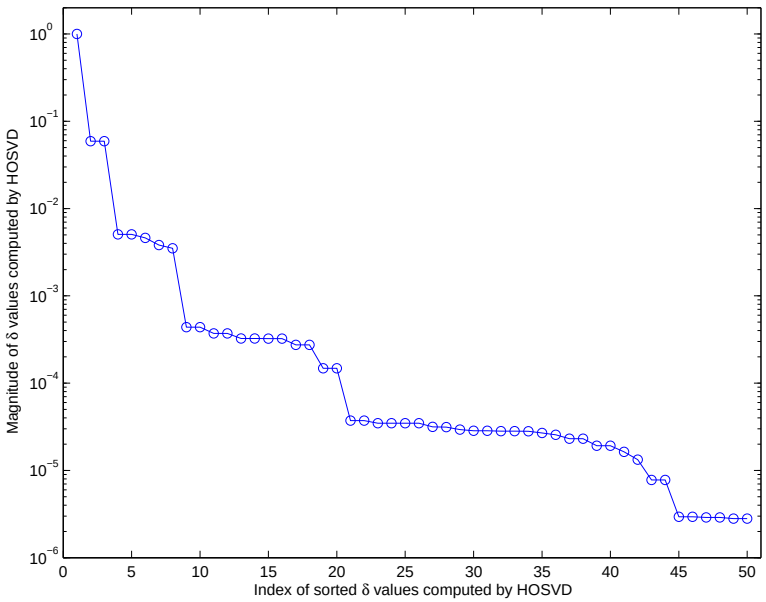

Figure 9: Largest 50 δ_{ijk} values computed by the HOSVD method for the PSF in example 2.

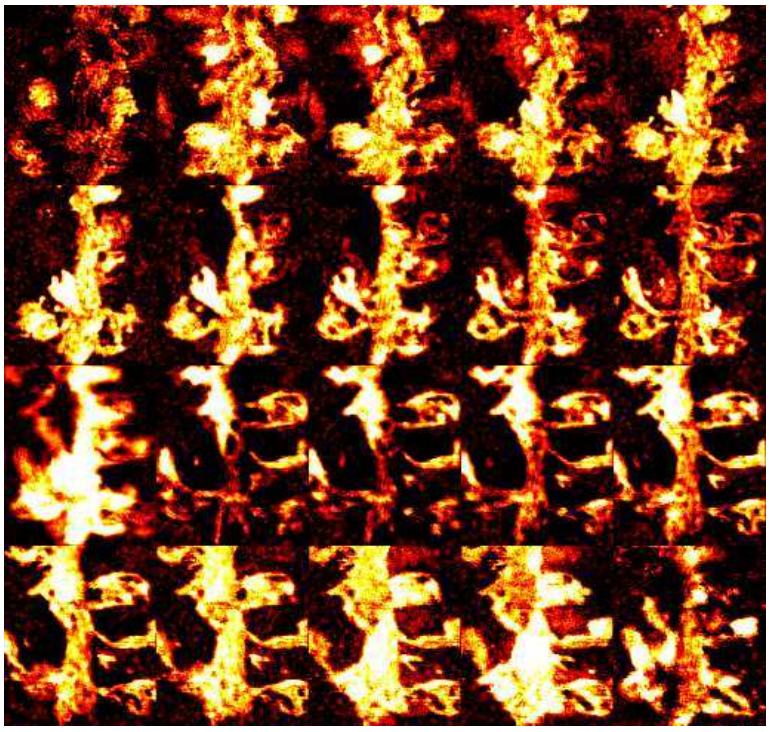

Figure 10: TSVD computed restoration (for example 2) using 1 term in the Kronecker product SVD approximation scheme.

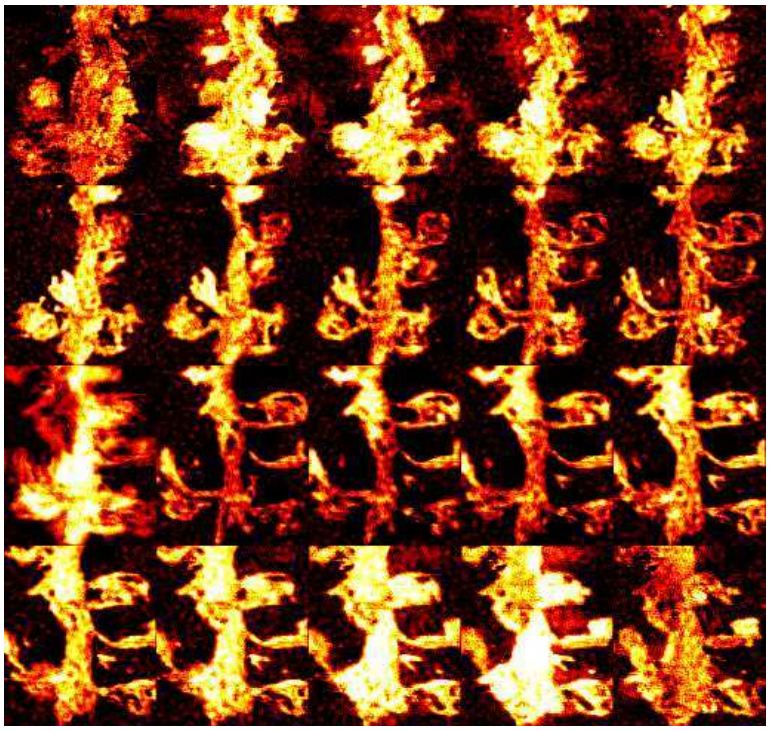

Figure 11: TSVD computed restoration (for example 2) using 3 terms in the Kronecker product SVD approximation scheme.

It is possible that slightly better restorations can be obtained by increasing the number of terms in the approximation. However, as with the previous example, we are more likely to obtain better reconstructions by using an iterative algorithm, or by incorporating additional information into the formulation of the problem. As an illustration, we computed restorations using a modified residual norm steepest descent method (MRNSD) [17, 22], and its preconditioned version (PMRNSD), which enforces a nonnegativity constraint at each iteration. The cost per iteration of this method is essentially the same as CGLS and PCGLS. For a preconditioner, we use the approximate SVD with 1 term in the Kronecker product approximation of K, and truncation tolerance $\tau = 0.01$. The convergence history (i.e., a plot of the relative errors at each iteration) is shown in Fig. 12. The dashed horizontal line on this plot shows the error level of the best TSVD reconstruction computed earlier. This plot illustrates the slow convergence of MRNSD, and the dramatic effect that can occur when using an appropriate preconditioner. The computed solution after 9 iterations of PMRNSD is shown in Fig. 13.

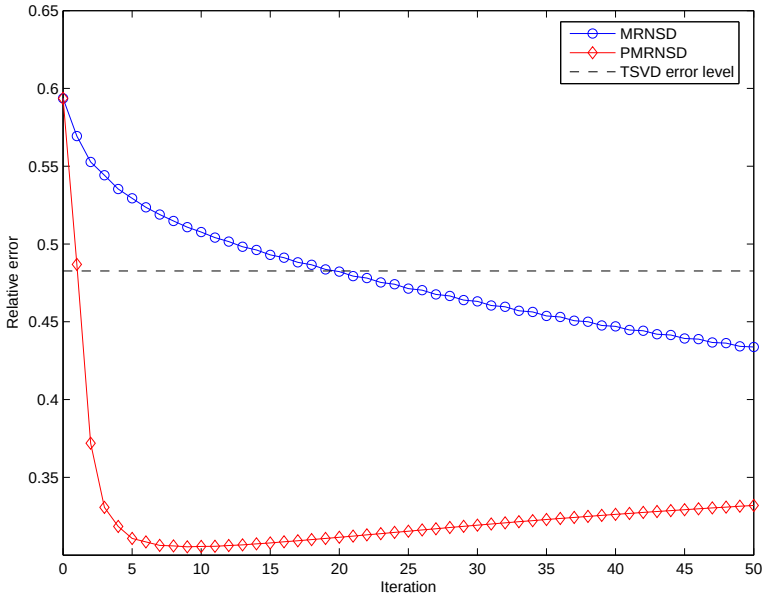

Figure 12: Convergence history of MRNSD and PMRNSD for example 2.

5 Concluding Remarks

The results we obtained with the Kronecker product and SVD approximations are very promising. We note that approaches other than the HOSVD can be used for computing a tensor decomposition of the PSF. For example, de Lathauwer, de Moor and Vandewalle [5] suggest using a power method that can, in particular, be used to find the best rank-1 tensor approximation of the PSF. Zhang and Golub [29] also study generalizations of the SVD to higher dimensions, and consider several algorithms for tensor decompositions.

We attempted to use the power method proposed in [5]. Some examples presented in [5] show that the power method can improve the HOSVD rank-1 approximation, but we did not observe any improvement in our application. Moreover, our numerical experiments indicate that the HOSVD performed very well for Kronecker product approximations. However, we have not fully investigated other orthogonal tensor decompositions, and it is possible that an alternative approach may produce

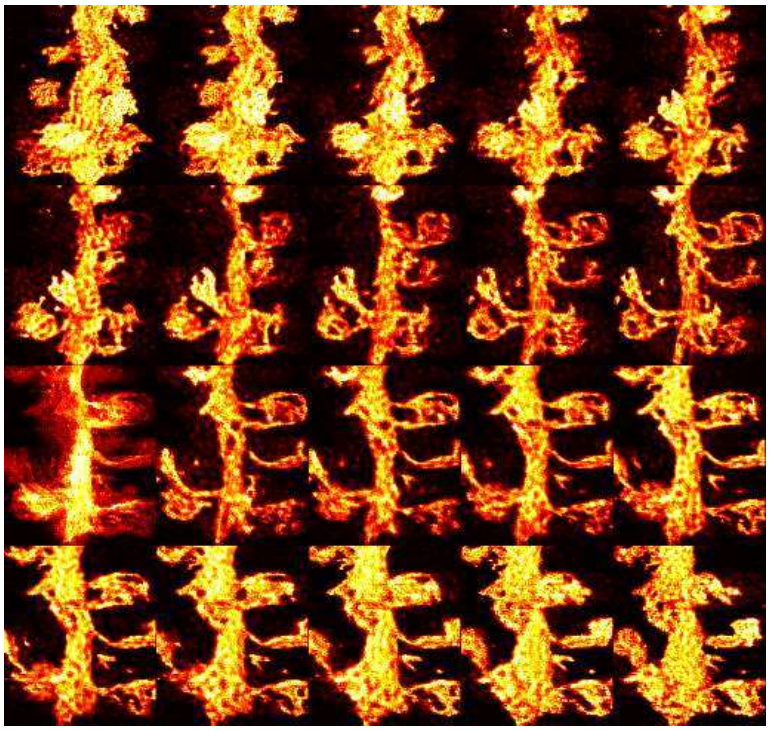

Figure 13: Restored image (for example 2) after 9 iterations of PMRNSD.

better results.

References

[1] H. Andrews and B. Hunt. *Digital Image Restoration*. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[2] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing Ltd., London, 1998.

[3] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, 1996.

[4] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.

[5] L. de Lathauwer, B. de Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21:1324–1342, 2000.

[6] A. Diaspro, M. Corosu, P. Ramoino, and M. Robello. Two-photon excitation imaging based on a compact scanning head. *IEEE Engineering in Medicine and Biology*, pages 18–22, September/October 1999.

[7] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht, 2000.

[8] J. A. Fessler. Preconditioning methods for shift-variant image reconstruction. *ICIP 97*, 1:185–188, 1997.

[9] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1992.

[10] C. W. Groetsch. *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*. Pitman, Boston, 1984.

[11] M. Hanke, J. G. Nagy, and R. J. Plemmons. Preconditioned iterative regularization. In L. Reichel, A. Ruttan, and R. S. Varga, editors, *Numerical Linear Algebra*, pages 141–163. de Gruyter, Berlin, 1993.

[12] P. C. Hansen. *Rank-deficient and discrete ill-posed problems*. SIAM, Philadelphia, PA, 1997.

[13] J. W. Hardy. Adaptive optics. *Scientific American*, 270(6):60–65, 1994.

[14] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, New York, NY, 1994.

[15] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[16] J. Kamm and J. G. Nagy. Optimal Kronecker product approximation of block Toeplitz matrices. *SIAM J. Matrix Anal. Appl.*, 22:155–172, 2000.

[17] L. Kaufman. Maximum likelihood, least squares, and penalized least squares for PET. *IEEE Trans. Med. Imag.*, 12:200–214, 1993.

[18] P. Khurd, Y. Xing, I. T. Hsiao, and G. Gindi. Fast preconditioned conjugate gradient reconstruction for 2D SPECT. In *Proc. of 2002 IEEE NSS/MIC*, pages 10–16, Norfolk, Virginia, 2002.

[19] M. Kilmer and G. W. Stewart. Iterative regularization and MINRES. *SIAM J. Matrix Anal. Appl.*, 21:613–628, 1999.

[20] T. G. Kolda. Orthogonal tensor decompositions. *SIAM J. Matrix Anal. Appl.*, 23:243–255, 2001.

[21] J. .G. Nagy, M. K. Ng, and L. C. Perrone. Kronecker product approximation for image restoration with reflexive boundary conditions. *SIAM J. Matrix Anal. Appl.*, to appear, 2004.

[22] J. G. Nagy and Z. Strakoš. Enforcing nonnegativity in image reconstruction algorithms. In et. al. D. C. Wilson, editor, *Mathematical Modeling, Estimation, and Imaging*, volume 3461, pages 182–190. SPIE, 2000.

[23] National Research Council Institute of Medicine. *Mathematics and Physics of Emerging Biomedical Imaging*. National Acadamy Press, Washington, D.C., 1997.

[24] F. Natterer. *The Mathematics of Computerized Tomography*. SIAM, Philadelphia, PA, 2001.

[25] F. Natterer and F. Wübbeling. *Mathematical Methods in Image Reconstruction*. SIAM, Philadelphia, PA, 2001.

[26] C. F. Van Loan and N. P. Pitsianis. Approximation with Kronecker products. In M. S. Moonen and G. H. Golub, editors, *Linear Algebra for Large Scale and Real Time Applications*, pages 293–314. Kluwer Publications, 1993.

[27] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, PA, 2002.

[28] G. Wang, M. W. Vannier, M. W. Skinner, M. G. P. Cavalcanti, and G. W. Harding. Spiral CT image deblurring for cochlear implantation. *IEEE Trans. Med. Imag.*, 17:251–262, 1998.

[29] T. Zhang and G. H. Golub. Rank-one approximation to high order tensors. *SIAM J. Matrix Anal. Appl.*, 23:534–550, 2001.