# KRONECKER PRODUCT APPROXIMATIONS FOR IMAGE RESTORATION WITH REFLEXIVE BOUNDARY CONDITIONS

JAMES G. NAGY*, MICHAEL K. NG[†], AND LISA PERRONE[‡]

**Abstract.** Many image processing applications require computing approximate solutions of very large, ill-conditioned linear systems. Physical assumptions of the imaging system usually mean that the matrices in these linear systems have exploitable structure. The specific structure depends on (usually simplifying) assumptions of the physical model, and other considerations such as boundary conditions. When reflexive (Neumann) boundary conditions are used, the coefficient matrix is a combination of Toeplitz and Hankel matrices. Kronecker products also occur, but this structure is not obvious from measured data. In this paper we discuss a scheme for computing a (possibly approximate) Kronecker product decomposition of structured matrices in image processing, which extends previous work [9] by Kamm and Nagy to a wider class of image restoration problems.

**Key words.** image restoration, Kronecker product, singular value decomposition

**AMS Subject Classification:** 65F20, 65F30

**1. Introduction.** Image restoration is the process of reconstructing an image of an unknown scene from an observed image, where

(1.1) `observed image = distortion(original scene) + noise`

The "distortion" can arise from many sources; atmospheric turbulence, out of focus lens, and motion blurs are but a few examples. Typically the distortion is described mathematically as a *point spread function* (PSF). Specifically, a PSF is a function that specifies how points in the image are distorted. PSFs are often classified as either *spatially invariant* or *spatially variant. Spatially invariant* means that the distortion is independent of position, while *spatially variant* means that the distortion does depend on position. Spatially invariant PSFs occur most frequently in applications [8], so this is what we consider in this paper.

A PSF can be further classified as *separable* or *nonseparable.* Separable means that the distortion in the horizontal and vertical directions is independent. That is, a two-dimensional distortion is a composition of two one-dimensional distortions. The topic of separability is often ignored when discussing image restoration problems, but, as we will see, by exploiting this structure, more choices are available in terms of image restoration algorithms.

We begin with a mathematical model of the spatially-invariant image restoration problem. The image of an object can be modeled as

$$\mathbf{g} = K\mathbf{f} + \mathbf{n} \tag{1.2}$$

where $\mathbf{g}$ is an $n^2$-vector representing the distorted image of size $n$-by-$n$, $\mathbf{f}$ is a vector representing the true image, and $\mathbf{n}$ is a vector representing additive noise. $K$ is an $n^2$-by-$n^2$ blurring matrix constructed from the PSF, but it has structure that can

*Department of Mathematics and Computer Science, Emory University (email: nagy@mathcs.emory.edu). This work was supported by the National Science Foundation under Grant DMS 00-75239.

[†]Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: mng@maths.hku.hk. Research supported in part by RGC Grant Nos. 7132/00P and 7130/02P and HKU CRCG Grant Nos 10203501, 10203907 and 10204437.

[‡]Department of Mathematics and Computer Science, Emory University (email: perrone@mathcs.emory.edu).

be exploited in computations. Because the blurring model is a convolution, $\mathbf{g}$ is not completely determined by $\mathbf{f}$ in the same domain where $\mathbf{g}$ is defined. Thus in solving $\mathbf{f}$ from $\mathbf{g}$, we need some assumptions on the values of $\mathbf{f}$ outside the domain of $\mathbf{g}$. These assumptions are called the boundary conditions. The structure of the blurring matrix $K$ depends on the boundary conditions [11].

- *Periodic boundary conditions*: The image outside the domain of consideration is a repeat, in all directions, of the image inside [4]. In this case, $K$ will be a block-circulant-circulant-block (BCCB) matrix. We can use two-dimensional Fast Fourier Transforms (FFTs) to diagonalize the matrix [4], but this boundary condition may be unrealistic in many situations.
- *Zero boundary conditions*: The values of $\mathbf{f}$ outside the domain of consideration are zero [1]. In this case, $K$ will be a block-Toeplitz-Toeplitz blocks (BTTB) matrix. FFTs can be used to implement fast matrix vector multiplications for $K$.
- *Reflexive boundary conditions*: The scene immediately outside the boundary is a reflection of the original scene inside. In this case, the matrix $K$ is block-Toeplitz-plus-Hankel with Toeplitz-plus-Hankel-blocks (BTHTHB) [11]. In the following discussion, we express the matrix $K$ as the sum of a block-Toeplitz-Toeplitz-block (BTTB) matrix, a block-Toeplitz-Hankel-block (BTHB) matrix, a block-Hankel-Toeplitz-block (BHTB) matrix and a block-Hankel-Hankel-block (BHHB) matrix. Although the matrix $K$ has a complicated structure, it can be diagonalized by the two-dimensional Fast Cosine Transform (FCT) when the PSF is symmetric [11].

In [5, 10, 11], it has been shown that using reflexive boundary conditions in image restoration or reconstruction can be better than using periodic or zero boundary conditions.

Aside from the issue of boundary conditions, it is well-known that blurring matrices are in general very ill-conditioned and image restoration algorithms will be extremely sensitive to noise [4]. The ill-conditioning of the blurring matrices stems from the wide range of magnitudes of their eigenvalues [3]. Therefore excess amplification of the noise at small eigenvalues can occur. In [11], classical Tikhonov regularization is employed to attain the stability of image restoration algorithms. A fast image restoration algorithm with the reflexive boundary conditions is developed and proposed. Since the size of the matrix $K$ is very large, iterative methods with cosine transform based preconditioners are used to speed up the convergence of the algorithm.

We note that if the blur is separable, then the matrix $K$ can be further decomposed into a Kronecker product of smaller matrices. In this case we are not restricted (by size constraints) to using only iterative methods. In particular, we can use Singular Value Decomposition (SVD) based methods [6] to perform the regularization in the image restoration process. The problem is determining when a PSF is separable. We may not have an explicit mathematical formula for the PSF, and thus must recognize separability from the image data. This has been done in the case of zero boundary conditions [9]. One aim of this paper is to consider how to do this for reflexive boundary conditions.

The outline of the paper is as follows. In Section 2, definitions and notations are set up. In Section 3, the Kronecker product approximation of the blurring matrix $K$ is studied, and we provide an algorithm for constructing this approximation from the given PSF. In Section 4, simulation results are presented to demonstrate the effectiveness of using this Kronecker approximation. Finally, some concluding remarks are given in Section 5.

## 2. Definitions and Notation.

In order to prove the main result of this paper, we need the following definitions and notations.

### 2.1. Toeplitz and Hankel Matrices.

Banded Toeplitz and Hankel matrices arise frequently in image restoration applications. Here we demonstrate how to use a column vector to represent these matrices:

- $\mathbf{toep}(\mathbf{a}, k)$ is an $n \times n$ banded Toeplitz matrix whose $k$th column is $\mathbf{a} \in \Re^n$. For example,

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{toep}(\mathbf{a}, 3) = \begin{bmatrix} a_3 & a_2 & a_1 & 0 \\ a_4 & a_3 & a_2 & a_1 \\ 0 & a_4 & a_3 & a_2 \\ 0 & 0 & a_4 & a_3 \end{bmatrix}$$

- $\mathbf{hank}(\mathbf{a}, k)$ is an $n \times n$ Hankel matrix with its first row and its last column defined by $[a_{k+1}, \ldots, a_n, 0, \ldots, 0]$ and $[0, \ldots, 0, a_1, \ldots, a_{k-1}]^T$ respectively, where $\mathbf{a} \in \Re^n$. For example,

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{hank}(\mathbf{a}, 3) = \begin{bmatrix} a_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_1 \\ 0 & 0 & a_1 & a_2 \end{bmatrix}$$

- We use the notation $\mathbf{Toep}(A, k)$ and $\mathbf{Hank}(A, k)$ for similar definitions with block matrices. For example:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} \quad \text{implies} \quad \mathbf{Toep}(A, 3) = \begin{bmatrix} A_3 & A_2 & A_1 & 0 \\ A_4 & A_3 & A_2 & A_1 \\ 0 & A_4 & A_3 & A_2 \\ 0 & 0 & A_4 & A_3 \end{bmatrix}$$

and

$$\mathbf{Hank}(A, 3) = \begin{bmatrix} A_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_1 \\ 0 & 0 & A_1 & A_2 \end{bmatrix}$$

- With the above notation, we can describe the blurring matrices that arise in image restoration. Let $P$ be an $n \times n$ array containing the image of a point spread function. Suppose the center of the PSF (location of the point source) is at $p_{ij}$. Let $\mathbf{p}_k^T$ be the $k$th row of $P$, and define

$$T_k = \mathbf{toep}(\mathbf{p}_k, j) \quad \text{and} \quad H_k = \mathbf{hank}(\mathbf{p}_k, j),$$

$$T = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix}.$$

Then, we can formulate the blurring matrix under

– zero boundary conditions as

$$K = \mathbf{Toep}(T, i)$$

– reflexive boundary conditions as

$$K = K_{tt} + K_{th} + K_{ht} + K_{hh}, \tag{2.1}$$

where $K_{tt} = \mathbf{Toep}(T, i)$, $K_{th} = \mathbf{Toep}(H, i)$, $K_{ht} = \mathbf{Hank}(T, i)$ and $K_{hh} = \mathbf{Hank}(H, i)$.

**2.2. The Shift Matrix.** We also need to use the shift matrix:

$$Z = \begin{bmatrix} 0 & 1 & & 0 \\ \vdots & \ddots & \ddots & \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

The name *shift matrix* comes from the fact that if we multiply a vector by $Z$, the entries are shifted up, and if we multiply by $Z^T$, the entries are shifted down. The following properties of the shift matrix will be used in Section 3.

1. $\mathbf{toep}(\mathbf{a}, k) = \begin{bmatrix} Z^{k-1}\mathbf{a} & \cdots & Z^0\mathbf{a} & \cdots (Z^{n-k})^T\mathbf{a} \end{bmatrix}$
2. If $\mathbf{e}_l$ is the $l$th column of the identity matrix, then
$$Z^k\mathbf{e}_l = \begin{cases} 0, & l = 1, 2, \ldots, k \\ \mathbf{e}_{l-k}, & l = k+1, \ldots, n \end{cases}$$
and
$$(Z^k)^T\mathbf{e}_l = \begin{cases} \mathbf{e}_{l+k}, & l = 1, 2, \ldots, n-k \\ 0, & l = n-k+1, \ldots, n \end{cases}$$
3. From Property 2, it is easy to show that:
$$(Z^k)^T(Z^k) = \text{diag}( \; [\, 0 \cdots 0 \;\; \underset{\substack{\uparrow \\ k+1 \text{ entry}}}{1} \cdots 1 \,] \; )$$
and
$$(Z^k)(Z^k)^T = \text{diag}( \; [\, 1 \cdots \underset{\substack{\uparrow \\ n-k \text{ entry}}}{1} \;\; 0 \cdots 0 \,] \; )$$
4. From Property 3, it follows that:
$$(Z^k)^T(Z^k) + (Z^{n-k})(Z^{n-k})^T = (Z^k)(Z^k)^T + (Z^{n-k})^T(Z^{n-k}) = I,$$
and thus,
$$\sum_{k=1}^{n-1} \left( (Z^k)^T(Z^k) + (Z^k)(Z^k)^T \right) = (n-1)I. \tag{2.2}$$
5. For $a, b < n$,
$$(Z^a)^T Z^b + Z^{n-a}(Z^{n-b})^T = \begin{cases} Z^{b-a} & \text{if b > a} \\ (Z^{a-b})^T & \text{if a > b} \end{cases} \tag{2.3}$$

**2.3. Kronecker Product Matrices.** Here we state some properties and definitions related to Kronecker products. A Kronecker product is defined to be:

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

Two transformations that we need are:

- The **vec** operator transforms 2-dimensional arrays into 1-dimensional vectors by stacking columns. For example:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{vec}(X) = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}$$

- The **tilde** transformation reorders the entries of a block matrix as follows. If $K$ is a block matrix:

$$K = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix}$$

  Then,

$$\tilde{K} = \mathbf{tilde}(K) = \begin{bmatrix} \mathbf{vec}(K_{11})^T \\ \vdots \\ \mathbf{vec}(K_{n1})^T \\ \vdots \\ \mathbf{vec}(K_{1n})^T \\ \vdots \\ \mathbf{vec}(K_{nn})^T \end{bmatrix}$$

  Note that in image restoration, with reflexive boundary conditions, from (2.1), we have:

$$\tilde{K} = \tilde{K}_{tt} + \tilde{K}_{th} + \tilde{K}_{ht} + \tilde{K}_{hh} \tag{2.4}$$

Van Loan and Pitsianis [12] show, for a general block matrix, that

$$\left\| K - \sum_{k=1}^{s} (A_k \otimes B_k) \right\|_F = \left\| \tilde{K} - \sum_{k=1}^{s} \tilde{\mathbf{a}}_k \tilde{\mathbf{b}}_k^T \right\|_F$$

where $\tilde{\mathbf{a}}_k = \mathbf{vec}(A_k)$ and $\tilde{\mathbf{b}}_k = \mathbf{vec}(B_k)$. The best Kronecker product approximation is obtained by finding the SVD of $\tilde{K}$. In particular, if

$$\tilde{K} = \sum_{k=1}^{r} \tilde{\sigma}_k \tilde{\mathbf{u}}_k \tilde{\mathbf{v}}_k^T$$

then the above Frobenius norm is minimized by taking

$$\tilde{\mathbf{a}}_k = \sqrt{\tilde{\sigma}_k}\tilde{\mathbf{u}}_k \quad \text{and} \quad \tilde{\mathbf{b}}_k = \sqrt{\tilde{\sigma}_k}\tilde{\mathbf{v}}_k.$$

The problem with this approach is that we need to compute the principal singular values and vectors of an $n^2 \times n^2$ matrix, $\tilde{K}$. The purpose of this paper is to show, for image restoration problems, how this computational effort can be reduced substantially by computing principal singular values and vectors of arrays of size at most $n \times n$.

## 3. Kronecker Product Approximation.

Let $K$ be the $n^2 \times n^2$ blurring matrix for a spatially invariant image restoration problem using reflexive boundary conditions (see Section 1), and suppose $P$ is the $n \times n$ PSF image array. In this section we show how a Kronecker product approximation of this $n^2 \times n^2$ matrix can be accomplished by computing the principal singular values and vectors of an $n \times n$ array related to $P$. This has been done for zero boundary conditions [9]. The case for reflexive boundary conditions is a bit more difficult to derive. To simplify notation, we consider only one term in the sum of Kronecker products; we describe how to extend this to an arbitrary number of terms later.

Our aim is, given the PSF, $P$, with center $p_{ij}$, find vectors $\mathbf{a}$ and $\mathbf{b}$ of length $n$ such that the matrices

$$\begin{aligned} A_t &= \mathbf{toep}(\mathbf{a}, i)\,, \quad & A_h &= \mathbf{hank}(\mathbf{a}, i) \\ B_t &= \mathbf{toep}(\mathbf{b}, j)\,, \quad & B_h &= \mathbf{hank}(\mathbf{b}, j) \end{aligned}$$

minimize $||K - (A_t + A_h) \otimes (B_t + B_h)||_F$ over all such Kronecker products. We first state the main result, and the corresponding algorithm that comes from it. The proof will come later.

THEOREM 3.1. *Let $P$ be an $n \times n$ PSF, with center $p_{ij}$. Let $R$ be the Cholesky factor of the $n \times n$ symmetric Toeplitz matrix with its first row $[n, 1, 0, 1, 0, 1, \cdots]$. Then*

$$||K - (A_t + A_h) \otimes (B_t + B_h)||_F = ||RPR^T - (R\mathbf{a})(R\mathbf{b})^T||_F.$$

We prove this theorem later. First we note that the Frobenius norm in the left hand side involves matrices with dimension $n^2 \times n^2$, and the Frobenius norm in the right hand side involves matrices with dimension $n \times n$. Based on this theorem,the algorithm for constructing the Kronecker product approximation of $K$ is as follows:

**Algorithm:** `Construct the approximation` $K \approx A \otimes B$

```
• Compute $R$
• Construct $P_r = RPR^T$
• Compute the SVD: $P_r = \sum \sigma_k \mathbf{u}_k \mathbf{v}_k^T$
• Construct the vectors: $\mathbf{a} = \sqrt{\sigma_1} R^{-1}\mathbf{u}_1$ and $\mathbf{b} = \sqrt{\sigma_1} R^{-1}\mathbf{v}_1$
```

- `Construct the matrices:` $A_t = \mathbf{toep}(\mathbf{a}, i)$, $A_h = \mathbf{hank}(\mathbf{a}, i)$, $B_t = \mathbf{toep}(\mathbf{b}, j)$, `and` $B_h = \mathbf{hank}(\mathbf{b}, j)$

In our experience, for real PSFs, the singular values of $P_r$ decay very quickly (see the numerical example in Section 4) to zero. In fact, it is often the case that $\sigma_1 >> \sigma_2 \approx \ldots \approx \sigma_n \approx 0$. Thus, $K \approx A \otimes B$ is generally a very good approximation. However, if a rank $s$ approximation is desired, where $1 < s \leq \text{rank}(P_r)$, the last two steps of the algorithm can be easily modified, as follows, to produce the approximation $K \approx \sum_{k=1}^{s} A_k \otimes B_k$.

- `For` $k = 1, 2, \ldots, s$, `construct the vectors:`
  $\mathbf{a}_k = \sqrt{\sigma_k} R^{-1}\mathbf{u}_k$ `and` $\mathbf{b}_k = \sqrt{\sigma_k} R^{-1}\mathbf{v}_k$
- `For` $k = 1, 2, \ldots, s$, `construct the matrices:`
  $A_{tk} = \mathbf{toep}(\mathbf{a}_k, i)$, $A_{hk} = \mathbf{hank}(\mathbf{a}_k, i)$, $B_{tk} = \mathbf{toep}(\mathbf{b}_k, j)$,
  `and` $B_{hk} = \mathbf{hank}(\mathbf{b}_k, j)$

In this case, the statement of theorem 3.1 becomes

$$||K - \sum_{k=1}^{s}(A_{tk} + A_{hk}) \otimes (B_{tk} + B_{hk})||_F = ||RPR^T - \sum_{k=1}^{s}(R\mathbf{a}_k)(R\mathbf{b}_k)^T||_F.$$

We now proceed to prove theorem 3.1. From Van Loan and Pitsianis [12], we have

$$\begin{aligned} ||K - (A_t + A_h) \otimes (B_t + B_h)||_F &= ||\tilde{K} - \mathbf{vec}(A_t + A_h)\mathbf{vec}(B_t + B_h)^T||_F \\ &= ||\tilde{K} - (\tilde{\mathbf{a}}_t + \tilde{\mathbf{a}}_h)(\tilde{\mathbf{b}}_t + \tilde{\mathbf{b}}_h)^T||_F \end{aligned}$$

where $\tilde{\mathbf{a}}_t = \mathbf{vec}(A_t)$, $\tilde{\mathbf{a}}_h = \mathbf{vec}(A_h)$, $\tilde{\mathbf{b}}_t = \mathbf{vec}(B_t)$, and $\tilde{\mathbf{b}}_h = \mathbf{vec}(B_h)$.

LEMMA 3.2. *Let $P$ be an $n \times n$ PSF with center $p_{ij}$, let $Z$ be the $n \times n$ shift matrix, and define $\tilde{K}_{tt}$, $\tilde{K}_{th}$, $\tilde{K}_{ht}$, and $\tilde{\mathrm{K}}_{\mathrm{hh}}$ as in (2.4). Then*

*(i)* $\tilde{K}_{tt} = D_{t,i}\tilde{P}D_{t,j}^T$
*(ii)* $\tilde{K}_{th} = D_{t,i}\tilde{P}D_{h,j}^T$
*(iii)* $\tilde{K}_{ht} = D_{h,i}\tilde{P}D_{t,j}^T$
*(iv)* $\tilde{K}_{hh} = D_{h,i}\tilde{P}D_{h,j}^T$

*where*

$$\tilde{P} = \begin{bmatrix} P & P & \cdots & P \\ P & P & \cdots & P \\ \vdots & \vdots & & \vdots \\ P & P & \cdots & P \end{bmatrix} \in \Re^{n^2 \times n^2}, \tag{3.1}$$

*$D_{t,k}$ and $D_{h,k}$ are block-diagonal matrices given by*

$$D_{t,k} = \text{diag}\,[Z^{k-1}, Z^{k-2}, \cdots, Z^1, Z^0, Z^T, \cdots, (Z^{n-k})^T] \in \Re^{n^2 \times n^2} \tag{3.2}$$

*and*

$$D_{h,k} = \text{diag}\,[Z^k, Z^{k+1}, \cdots, Z^{n-1}, Z^n, (Z^{n-1})^T, \cdots, (Z^{n-k+1})^T] \in \Re^{n^2 \times n^2}. \tag{3.3}$$

*Proof.* We only prove (i); similar techniques can be used to establish the other relations. First observe that

$$K_{tt} = \begin{bmatrix} \mathbf{toep}(\mathbf{p}_i, j) & \cdots & \mathbf{toep}(\mathbf{p}_1, j) & & \\ \vdots & \ddots & \vdots & \ddots & \\ \mathbf{toep}(\mathbf{p}_n, j) & \cdots & \mathbf{toep}(\mathbf{p}_i, j) & \cdots & \mathbf{toep}(\mathbf{p}_1, j) \\ & \ddots & \vdots & \ddots & \vdots \\ & & \mathbf{toep}(\mathbf{p}_n, j) & \cdots & \mathbf{toep}(\mathbf{p}_i, j) \end{bmatrix},$$

where $\mathbf{p}_k^T$ is the $k$th row of $P$. Denote the $k$th block column of $K_{tt}$ as $[K_{tt}]_k$; that is

$$[K_{tt}]_k = \begin{bmatrix} \mathbf{toep}(\mathbf{p}_{i-k+1}, j) \\ \vdots \\ \mathbf{toep}(\mathbf{p}_n, j) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} i-k \qquad \text{if} \quad 1 \le k \le i$$

and

$$[K_{tt}]_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{toep}(\mathbf{p}_{i-k+1}, j) \\ \vdots \\ \mathbf{toep}(\mathbf{p}_n, j) \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} k-i \qquad \text{if} \quad i+1 \le k \le n.$$

Then, for $1 \le k \le i$, we have

$$\begin{aligned} \left[\tilde{K}_{tt}\right]_k &= \begin{bmatrix} \mathbf{vec}(\mathbf{toep}(\mathbf{p}_{i-k+1}, j))^T \\ \vdots \\ \mathbf{vec}(\mathbf{toep}(\mathbf{p}_n, j))^T \\ 0^T \\ \vdots \\ 0^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{p}_{i-k+1}^T & \cdots & \mathbf{p}_{i-k+1}^T \\ \vdots & & \vdots \\ \mathbf{p}_n^T & \cdots & \mathbf{p}_n^T \\ 0^T & \cdots & 0^T \\ \vdots & & \vdots \\ 0^T & \cdots & 0^T \end{bmatrix} \begin{bmatrix} (Z^{j-1})^T & & & & \\ & \ddots & & & \\ & & Z^0 & & \\ & & & Z & \\ & & & & \ddots \\ & & & & & Z^{n-j} \end{bmatrix} \\ &= Z^{i-k} \begin{bmatrix} P & \cdots & P \end{bmatrix} D_{t,j}^T. \end{aligned}$$

Similarly, for $i+1 \le k \le n$, we have

$$\left[\tilde{K}_{tt}\right]_k = (Z^{k-i})^T \begin{bmatrix} P & \cdots & P \end{bmatrix} D_{t,j}^T,$$

and, therefore,

$$\tilde{K}_{tt} = \begin{bmatrix} \tilde{T}_1 \\ \vdots \\ \tilde{T}_i \\ \vdots \\ \tilde{T}_n \end{bmatrix} = \begin{bmatrix} Z^{i-1} \begin{bmatrix} P & \cdots & P \end{bmatrix} D_{t,j}^T \\ \vdots \\ Z^0 \begin{bmatrix} P & \cdots & P \end{bmatrix} D_{t,j}^T \\ \vdots \\ (Z^{n-i})^T \begin{bmatrix} P & \cdots & P \end{bmatrix} D_{t,j}^T \end{bmatrix} = D_{t,i} \tilde{P} D_{t,j}^T .$$

□

According to Lemma 3.2, we have

$$\tilde{K} = (D_{t,i} + D_{h,i}) \tilde{P} (D_{t,j} + D_{h,j})^T .$$

The next lemma states the forms of the vectors $\tilde{\mathbf{a}}_t$, $\tilde{\mathbf{a}}_h$, $\tilde{\mathbf{b}}_t$ and $\tilde{\mathbf{b}}_h$.

LEMMA 3.3. *Let $\tilde{\mathbf{a}}_t$, $\tilde{\mathbf{a}}_h$, $\tilde{\mathbf{b}}_t$ and $\tilde{\mathbf{b}}_h$ be defined as above. Then*

$$\tilde{\mathbf{a}}_t + \tilde{\mathbf{a}}_h = (D_{t,i} + D_{h,i}) \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \vdots \\ \mathbf{a} \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{b}}_t + \tilde{\mathbf{b}}_h = (D_{t,j} + D_{h,j}) \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \\ \vdots \\ \mathbf{b} \end{bmatrix}$$

The proof of Lemma 3.3 is similar to that of Lemma 3.2. We just note that $\tilde{\mathbf{a}}_t = \mathbf{vec}(A_t)$, $\tilde{\mathbf{a}}_h = \mathbf{vec}(A_h)$, $\tilde{\mathbf{b}}_t = \mathbf{vec}(B_t)$, and $\tilde{\mathbf{b}}_h = \mathbf{vec}(B_h)$ where $A_t = \mathbf{toep}(\mathbf{a}, i)$, $A_h = \mathbf{hank}(\mathbf{a}, i)$, $B_t = \mathbf{toep}(\mathbf{b}, j)$, and $B_h = \mathbf{hank}(\mathbf{b}, j)$.

From Lemmas 3.2 and 3.3, we obtain:

$$\begin{aligned} &\tilde{K} - (\tilde{\mathbf{a}}_t + \tilde{\mathbf{a}}_h)(\tilde{\mathbf{b}}_t + \tilde{\mathbf{b}}_h)^T \\ &= (D_{t,i} + D_{h,i}) \left( \tilde{P} - \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \vdots \\ \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{b}^T & \mathbf{b}^T & \cdots & \mathbf{b}^T \end{bmatrix} \right) (D_{t,j} + D_{h,j})^T. \end{aligned}$$

The next lemma states that the matrix on the right hand side of the above relation can be simplified.

LEMMA 3.4. *There is an orthogonal matrix $Q$ such that*

$$Q^T \left( \tilde{P} - \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \vdots \\ \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{b}^T & \mathbf{b}^T & \cdots & \mathbf{b}^T \end{bmatrix} \right) Q = n \begin{bmatrix} P - \mathbf{a}\mathbf{b}^T & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} .$$

*Furthermore, $Q$ has the form*

$$Q = \frac{1}{\sqrt{n}} \begin{bmatrix} I & * & \cdots & * \\ I & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ I & * & \cdots & * \end{bmatrix} .$$

*Proof.* Let $c_k = \dfrac{\sqrt{k}}{\sqrt{k+1}}$ and $s_k = \dfrac{1}{\sqrt{k+1}}$ and note that for any matrix $X$,

$$\begin{bmatrix} c_k I & s_k I \\ -s_k I & c_k I \end{bmatrix} \begin{bmatrix} \sqrt{k} X \\ X \end{bmatrix} = \begin{bmatrix} \sqrt{k+1} X \\ 0 \end{bmatrix}.$$

Define the orthogonal matrix $Q_k$ as

$$Q_k = \begin{bmatrix} c_k I & 0 & \cdots & 0 & -s_k I & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & I & 0 & 0 & \cdots & 0 \\ s_k I & 0 & \cdots & 0 & c_k I & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I \end{bmatrix} \begin{array}{l} \\ \\ \\ \\ \leftarrow \ (k+1)\text{th block row.} \\ \\ \\ \\ \end{array}$$

Using $\tilde{P}$ as given in (3.1), it follows that

$$Q_{n-1}^T \cdots Q_2^T Q_1^T \left( \tilde{P} - \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \vdots \\ \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{b}^T & \mathbf{b}^T & \cdots & \mathbf{b}^T \end{bmatrix} \right) Q_1 Q_2 \cdots Q_{n-1}$$
$$= n \begin{bmatrix} P - \mathbf{a}\mathbf{b}^T & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Furthermore, a simple induction argument can be used to show that

$$Q_1 Q_2 \cdots Q_{n-1} = \begin{bmatrix} \left(\prod_{k=1}^{n-1} c_k\right) I & * & * & \cdots & * \\ \left(s_1 \prod_{k=2}^{n-1} c_k\right) I & c_1 I & * & \cdots & * \\ \left(s_2 \prod_{k=3}^{n-1} c_k\right) I & 0 & c_2 I & \cdots & * \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_{n-1} I & 0 & 0 & \cdots & c_{n-1} I \end{bmatrix}.$$

By observing that

$$s_i \left( \prod_{k=i+1}^{n-1} c_k \right) = \frac{1}{\sqrt{i+1}} \left( \frac{\sqrt{i+1}}{\sqrt{i+2}} \frac{\sqrt{i+2}}{\sqrt{i+3}} \cdots \frac{\sqrt{n-1}}{\sqrt{n}} \right) = \frac{1}{\sqrt{n}},$$

the lemma is proved. □

We now have all of the tools needed to prove our main Theorem.
*Proof of Theorem 3.1.* Using Lemma 3.4., we obtain

$$\tilde{K} - (\tilde{\mathbf{a}}_t + \tilde{\mathbf{a}}_h)(\tilde{\mathbf{b}}_t + \tilde{\mathbf{b}}_h)^T$$

$$= (D_{t,i}+D_{h,i})QQ^T\left(\tilde{P}-\begin{bmatrix}\mathbf{a}\\\mathbf{a}\\\vdots\\\mathbf{a}\end{bmatrix}\begin{bmatrix}\mathbf{b}^T & \mathbf{b}^T & \cdots & \mathbf{b}^T\end{bmatrix}\right)QQ^T(D_{t,j}+D_{h,j})^T$$

$$= (D_{t,i}+D_{h,i})\hat{Q}\begin{bmatrix}P-\mathbf{a}\mathbf{b}^T & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & 0\end{bmatrix}\hat{Q}^T(D_{t,j}+D_{h,j})^T,$$

where

$$\hat{Q}=\begin{bmatrix}I & * & \cdots & *\\ I & * & \cdots & *\\ \vdots & \vdots & & \vdots\\ I & * & \cdots & *\end{bmatrix}.$$

The next task is to study the form of the matrix $(D_{t,i}+D_{h,i})\hat{Q}$. Note that if we find the $QR$ factorizations

$$(D_{t,i}+D_{h,i})\hat{Q}=\tilde{Q}_i\tilde{R}_i\,,\quad (D_{t,j}+D_{h,j})\hat{Q}=\tilde{Q}_j\tilde{R}_j\,, \tag{3.4}$$

then we have

$$\begin{aligned}
&\|\tilde{K}-(\tilde{\mathbf{a}}_t+\tilde{\mathbf{a}}_h)(\tilde{\mathbf{b}}_t+\tilde{\mathbf{b}}_h)^T\|_F\\
&=\left\|\tilde{Q}_i\tilde{R}_i\begin{bmatrix}P-\mathbf{a}\mathbf{b}^T & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & 0\end{bmatrix}\tilde{R}_j^T\tilde{Q}_j^T\right\|_F\\
&=\left\|\begin{bmatrix}R_i & * & \cdots & *\\ 0 & * & \cdots & *\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & *\end{bmatrix}\begin{bmatrix}P-\mathbf{a}\mathbf{b}^T & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & 0\end{bmatrix}\begin{bmatrix}R_j^T & 0 & \cdots & 0\\ * & * & \cdots & 0\\ \vdots & \vdots & & \vdots\\ * & * & \cdots & *\end{bmatrix}\right\|_F\\
&=\|R_i(P-\mathbf{a}\mathbf{b}^T)R_j^T\|_F\,,
\end{aligned}$$

where $R_i$ and $R_j$ are, respectively, the $n\times n$ leading principal submatrices of the $n^2\times n^2$ upper triangular matrices $\tilde{R}_i$ and $\tilde{R}_j$. The next task is to determine the matrices $R_i$ and $R_j$. By (3.4), we can determine $R_i$ by finding the Cholesky factorization of the $n\times n$ leading principal submatrix of

$$\hat{Q}^T(D_{t,i}+D_{h,i})^T(D_{t,i}+D_{h,i})\hat{Q}\,.$$

Because the first column block of $\hat{Q}$ contains identity matrices, it is not hard to see that this submatrix is simply

$$R_i^TR_i=\begin{bmatrix}I & I & \cdots & I\end{bmatrix}(D_{t,i}+D_{h,i})^T(D_{t,i}+D_{h,i})\begin{bmatrix}I\\ I\\ \vdots\\ I\end{bmatrix}.$$

Similarly, we obtain

$$R_j^T R_j = \begin{bmatrix} I & I & \cdots & I \end{bmatrix} (D_{t,j} + D_{h,j})^T (D_{t,j} + D_{h,j}) \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}.$$

Let us consider $R_i^T R_i$ for the case that $i \leq \frac{n}{2}$. To simplify the presentation, we define a matrix $W$ as

$$W = (D_{t,i} + D_{h,i}) \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}.$$

Using (3.2) and (3.3) with $k = i$, we have

$$W = \begin{bmatrix} Z^{i-1} & + & Z^{i} \\ \vdots & & \vdots \\ Z^{1} & + & Z^{2i-2} \\ Z^{0} & + & Z^{2i-1} \\ (Z^{1})^T & + & Z^{2i} \\ \vdots & & \vdots \\ (Z^{n-2i})^T & + & Z^{n-1} \\ (Z^{n-2i+1})^T & + & Z^{n} \\ (Z^{n-2i+2})^T & + & (Z^{n-1})^T \\ \vdots & & \vdots \\ (Z^{n-i})^T & + & (Z^{n-i+1})^T \end{bmatrix}.$$

After multiplication and rearrangement of terms,

$$\begin{aligned} R_i^T R_i &= W^T W \\ &= Z^0 Z^0 + \sum_{k=1}^{n-1} \left( (Z^k)^T Z^k + Z^k (Z^k)^T \right) + \sum_{k=1}^{i-1} \left( Z^{2k-1} + (Z^{2k-1})^T \right) + \\ &\quad + \sum_{k=i}^{n-i} \left( Z^{2k-1} + (Z^{2k-1})^T \right), \end{aligned}$$

where the second summation utilizes (2.3) and the remaining terms arise directly from multiplication. Using (2.2) and simplifying, we get

$$R_i^T R_i = nI + \sum_{k=1}^{n-i} \left( Z^{2k-1} + (Z^{2k-1})^T \right).$$

Since in this case $i \leq \frac{n}{2}$, we have $2n - 2i - 1 \geq n - 1$, so the largest exponent on $Z$ in the sum will be *at least* large enough to 'fill' every other off-diagonal of $R_i^T R_i$ with 1's. Therefore, $R_i^T R_i$ is the $n \times n$ symmetric Toeplitz matrix with first row given by

$$\begin{bmatrix} n & 1 & 0 & 1 & 0 & 1 & \cdots \end{bmatrix}.$$

Using $j$ in place of $i$ in the argument above, $R_j^T R_j$ yields the same matrix when $j \leq \frac{n}{2}$. In the cases of $i > \frac{n}{2}$ and $j > \frac{n}{2}$, similar proofs generate identical results. Thus, for all possible values of $i$ and $j$, $R_i^T R_i = R_j^T R_j =$ the $n \times n$ symmetric Toeplitz matrix described above. By setting $R = R_i = R_j$, we complete the proof of our main theorem.

**4. Numerical Examples.** Now that we have a Kronecker sum approximation for the blurring matrix under reflexive boundary conditions, we illustrate how it can be used for an image restoration example. Recall that the image formation model is given in equation (1.2). The test data we use is shown in Figures 4.1 and 4.2. The $256 \times 256$ blurred and noisy image, $\mathbf{g}$, shown on the right side of Figure 4.1, and its corresponding true image, $\mathbf{f}$, on the left, have been excised from larger $512 \times 512$ images. Blurring was performed on the larger image so that the natural boundary elements would contribute to the blur, and 0.1% Gaussian white noise was added to the pixel values. All numerical tests reported here were performed on the smaller image using fabricated (reflexive and zero [9]) boundary conditions. All computations were done in Matlab 6.1.

The PSF, shown in Figure 4.2 is an example of blurring that occurs in wavefront coding, where a cubic phase filter is used to improve depth of field resolution in light efficient wide aperture optical systems [2]. A plot of the first 60 singular values of the PSF is also shown in Figure 4.2. Note that the largest singular value dominates the spectrum by an order of magnitude. In fact, for all singular values smaller than $\sigma_5$, it dominates the spectrum by two orders of magnitude. In our experience, this is typical in image restoration problems, whether the PSF is symmetric or nonsymmetric. For this reason, a Kronecker sum approximation with $s \leq 5$ can generally provide excellent restorations. We remark that since the PSF is nonsymmetric (and cannot be approximated well by a symmetric PSF), using a cosine transform based preconditioner with Tikhonov regularization [10] is not effective.

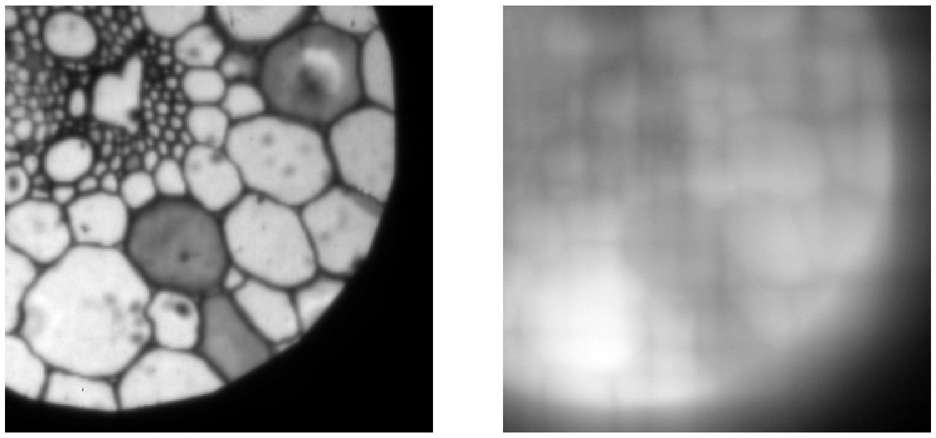

FIG. 4.1. *The true image, and the blurred, noisy image to be restored.*

As in [9], the Kronecker product decomposition is used to construct an approximate SVD of $K$, which can then be used in image restoration algorithms. That is, suppose $K$ is approximated by $T = \sum_{k=1}^{s} A_k \otimes B_k$, where $A_k$ and $B_k$ are $n \times n$ Toeplitz plus Hankel matrices computed according to the algorithm in Section 3. An approximate SVD for $K$ can be computed as

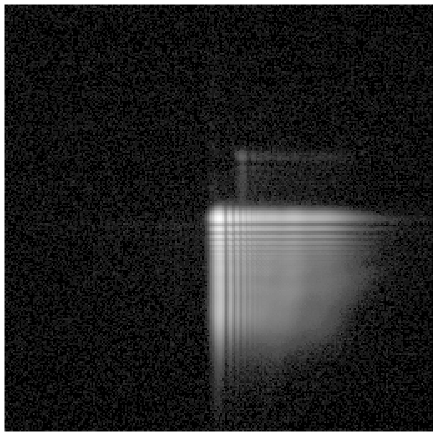
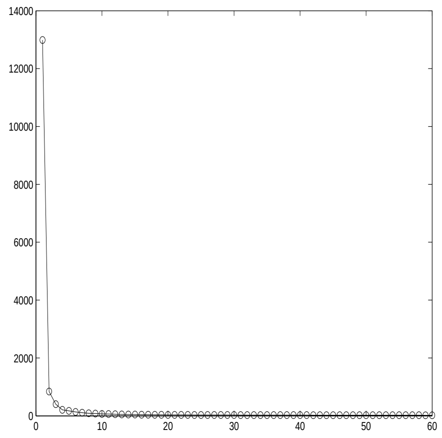

FIG. 4.2. *Image of the nonsymmetric PSF, and a plot of its first 60 singular values.*

$$\begin{array}{rcl} K & \approx & U\Sigma V^T, \\ U & = & U_A \otimes U_B, \\ V & = & V_A \otimes V_B, \\ \Sigma & = & \text{diag}(U^T T V) \\ & = & \text{diag}(U^T(A_1 \otimes B_1 + A_2 \otimes B_2 + \ldots + A_s \otimes B_s)V), \end{array}$$

where $A_1 = U_A \Sigma_A V_A^T$ and $B_1 = U_B \Sigma_B V_B^T$. Since $s$ is usually small ($s \leq 5$) the cost of the above scheme is only $O(n^3)$ (as opposed to $O(n^6)$, which is the cost of computing an SVD of $K$ directly). It is therefore computationally viable to consider, for example, using this approximation with the truncated singular value decomposition (TSVD). The TSVD solution is given by

$$\mathbf{f}_{TSVD} = V\Sigma^+ U^T \mathbf{g}, \quad \Sigma^\dagger = \mathrm{d}iag(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_t}, 0, \ldots, 0),$$

where $t$ is called a truncation index, or regularization parameter. The truncation index is problem dependent; several approaches may be used to choose an appropriate value [3, 7]. For our experiments, we use generalized cross validation (GCV):

$$t = \arg\min_k G(k) = \arg\min_k \frac{||K\mathbf{f}_k - \mathbf{g}||_2^2}{(N-k)^2},$$

where $N$ is the number of pixels in the image, and

$$\mathbf{f}_k = V\text{diag}(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}, 0, \ldots, 0)U^T \mathbf{g}.$$

We computed TSVD restorations using the SVD approximation based on sums of $s$ Kronecker products, for several values of $s$, but the results were visually indistinguishable, so only those for $s = 1$ are reported here. Figure 4.3 shows TSVD restorations using reflexive (relative error 0.3358) and zero (relative error 0.6862) boundary conditions. As expected, the reflexive boundary condition has addressed the problem of ringing effects at the image boundary.

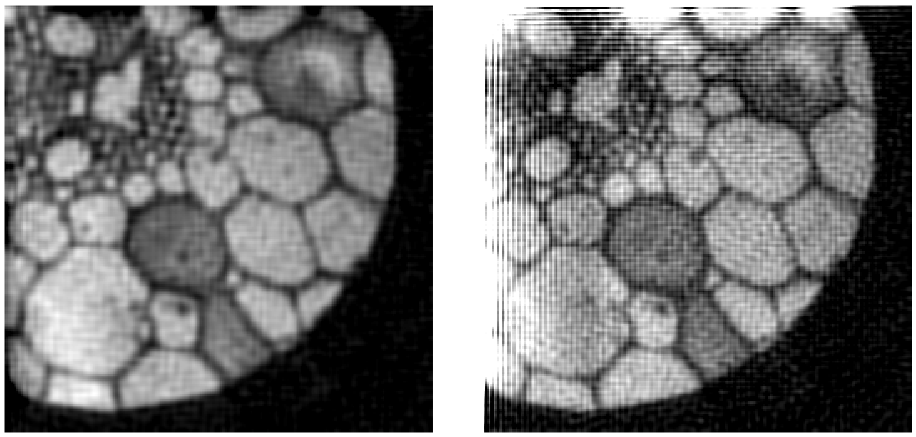

FIG. 4.3. *TSVD restoration with reflexive boundary conditions and TSVD restoration with zero boundary conditions.*

## 5. Concluding Remarks.

In this paper, we have studied SVD-based regularization methods for solving image restoration problems with reflexive boundary conditions. We have shown that a Kronecker product decomposition of block-Toeplitz-plus-Hankel with Toeplitz-plus-Hankel-block matrices from image restoration problems can be determined by computing the singular value decomposition of weighted point spread functions. Numerical results suggest that the reflexive boundary condition provides an effective model for image restoration problems in terms of the minimization of the ringing effects near the boundary. We also find that the SVD-based regularization method using the Kronecker product decomposition is efficient in terms of the computational cost of solving image restoration problems.

## REFERENCES

[1] H. ANDREWS AND B. HUNT, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[2] E. R. DOWSKI AND W. T. CATHEY, *Extended depth of field through wavefront coding*, Applied Optics, 34 (1995), pp. 1859–1866.

[3] H. W. ENGL, M. HANKE, AND A. NEUBAUER, *Regularization of Inverse Problems*, Kluwer Academic Publishers, Dordrecht, 2000.

[4] R. GONZALEZ AND R. WOODS, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

[5] D. GRISHIN AND T. STROHMER, *Fast multi-dimensional scattered data*. preprint, 2002.

[6] P. C. HANSEN, *Regularization tools: A Matlab package for the analysis and solution of discrete ill-posed problems*, Numerical Algorithms, 6 (1994), pp. 1–35.

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

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

[9] J. KAMM AND J. G. NAGY, *Optimal Kronecker product approximation of block Toeplitz matrices*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 155–172.

[10] M. NG, R. CHAN, T. CHAN, AND A. YIP, *Cosine transform preconditioners for high resolution image reconstruction*, Linear Algebra Appl., 316 (2000), pp. 89–104.

[11] M. K. NG, R. H. CHAN, AND W. TANG, *A fast algorithm for deblurring models with Neumann boundary conditions*, SIAM J. Sci. Comput., 21 (1999), pp. 851–866.

[12] C. F. VAN LOAN AND N. P. PITSIANIS, *Approximation with Kronecker products*, in Linear Algebra for Large Scale and Real Time Applications, M. S. Moonen and G. H. Golub, eds., Kluwer Publications, 1993, pp. 293–314.