RESTORING IMAGES DEGRADED BY SPATIALLY-VARIANT BLUR

JAMES G. NAGY* AND DIANNE P. O'LEARY†

February 24, 1995

Abstract. Restoration of images that have been blurred by the effects of a Gaussian blurring function is an ill-posed but well-studied problem. Any blur that is spatially invariant can be expressed as a convolution kernel in an integral equation. Fast and effective algorithms then exist for determining the original image by preconditioned iterative methods. If the blurring function is spatially variant, however, then the problem is more difficult. In this work we develop fast algorithms for forming the convolution and for recovering the original image when the convolution functions are spatially variant but have a small domain of support. This assumption leads to a discrete problem involving a banded matrix. We devise an effective preconditioner and prove that the preconditioned matrix differs from the identity by a matrix of small rank plus a matrix of small norm. Numerical examples are given, related to the Hubble Space Telescope Wide-Field / Planetary Camera. The algorithms that we develop are applicable to other ill-posed integral equations as well.

Key words. image restoration, spatially-variant point spread function, discrete ill-posed problems, convolution, first-kind integral equations, regularization.

AMS(MOS) subject classifications. 65F20, 65F30.

1. Introduction. An ideal camera or recording device would record an image so that the intensity of a small piece (pixel) of the recorded image was directly proportional to the intensity of the corresponding section of the scene being recorded. Real cameras violate this model in two ways:

- The recorded intensity of a pixel is related to the intensity in a larger neighborhood of the corresponding section of the scene. This effect in visual images is called *blurring*.
- The recorded intensities are contaminated by random noise.

The natural mathematical model of the recording operation is an integral equation of the first kind:

$$\int_{\Omega} k(s,t) f^*(t) dt = g(s) - \eta(s) = g^*(s).$$

Here the spatial coordinates are $s \in \mathcal{R}^2$ and $t \in \mathcal{R}^2$, and Ω is a closed region containing the domain of the image. The blurring of the unknown true image $f^* : \mathcal{R}^2 \rightarrow \mathcal{R}$ is modeled by convolution with the *point spread function* (*kernel*) $k : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ plus the addition of noise $\eta : \mathcal{R}^2 \rightarrow \mathcal{R}$. The function $g : \mathcal{R}^2 \rightarrow \mathcal{R}$ is the measured image, usually known only for certain discrete values of s, while g^* is the unknown exact blurred image. Since the number of measurements is finite, the model is discretized into a matrix equation

$$K\mathbf{f} \approx \mathbf{g}$$

where $K \in \mathcal{R}^{n \times n}$. (Our algorithms also have natural extensions to overdetermined systems, but we prefer to keep the notation simple by assuming that K is square.)

* Department of Mathematics, Southern Methodist University, Dallas, TX 75275. (nagy@cygnus.math.smu.edu). This work was sponsored by an NSF Postdoctoral Research Fellowship in the Mathematical Sciences.

† Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu).

We order the equations and unknowns in a natural way, with the second component of s changing the fastest.

Most often, the point spread function k is assumed to be *spatially invariant*: $k(s,t) = k(s-t)$. This assumption is popular for two reasons:

- The matrix K is then a block Toeplitz matrix with Toeplitz blocks, and matrix-vector multiplication can be accomplished quite quickly by making use of fast Fourier transforms (cf. §3).
- The point spread function is often determined by an idealized model, and spatial dependence is usually ignored. Alternatively, the point spread function can be determined experimentally by aiming the camera at a point source and recording the result. Of course, this approach is prone to error because of the added noise and the departure of the point source from a delta function, so data from multiple trials is usually averaged to reduce noise, rather than used to determine spatial dependence.

There are situations, though, in which it is important to take account of the spatial variation of the point spread function. For example, the point spread function for the original Hubble Space Telescope Wide-Field / Planetary Camera had a large amount of spatial variation because of errors in the shaping of the mirrors [1]. As another example, if the scene contains two objects moving with different velocities relative to the recording device, then the point spread function is effectively different for each [11].

Problems like these provided motivation for our project; our goal was to develop algorithms for deblurring an image with a spatially variant point spread function with cost comparable to that for spatially invariant ones. These algorithms are efficient implementations of the conjugate gradient algorithm for finding an approximate solution to $K\mathbf{f} = \mathbf{g}$ using the CGLS algorithm; see Björck [2]. The main problem with iterative methods such as this, especially when K is poorly conditioned, is that they may converge very slowly. Therefore, an effective preconditioning scheme for spatially varying point spread functions is devised.

Although we phrase our results in terms of deblurring, it is clear that our methods are applicable to more general integral equations of the first kind with spatially variant kernel functions. They will be efficient when the support of the function $k(s,t)$ (for each fixed value of s) is significantly smaller than the domain of integration and when k can be approximated well by a small number of spatially invariant kernels.

In the next section we describe our mathematical model for the point spread function. We discuss fast matrix-vector products in §3, and we develop effective preconditioning matrices in §4. Numerical results and final remarks can be found in §5 and §6.

2. The Model for the Point Spread Function. We assume that we have been given the point spread function $k(s,t)$ for several different values $s_1, \ldots, s_p$. This corresponds to knowledge of p different rows of the matrix K, or, alternatively, p different point spread functions $k_i(t) \equiv k(s_i, t)$, $i = 1, \ldots, p$. Our first task is to define the other rows of K (or $n-p$ other point spread functions) in a reasonable way.

The easiest way to extend the data is to define regions of influence for each of these measured point spread functions: we will partition our domain into p non-overlapping regions, and assume that in the ith region the point spread function is spatially invariant, so that $k(s,t) = k_i(t - s + s_i)$. Thus we have constructed a kernel defined by

$$k(s,t) \equiv \delta_1(s)\, k_1(t-s+s_1) + \cdots + \delta_p(s)\, k_p(t-s+s_p),$$

where $\delta_j(s)$ is the indicator function for the jth region, 1 if s is in the region and 0 otherwise.

In matrix terms, a spatially invariant point spread function on a two dimensional domain corresponds to a block Toeplitz matrix with Toeplitz blocks [9]. Our partitioning of the image leads to a matrix K defined by

$$K \equiv D_1 K_1 + \cdots + D_p K_p, \tag{2.1}$$

where D_i is a diagonal matrix whose jth diagonal element is 1 if the jth point is in region i, and 0 otherwise.

This set of definitions leads to a discontinuous kernel function k. This is easily remedied, however: rather than approximating k by a piecewise constant function, we can choose weights $\delta_i(s)$ to achieve higher-order interpolation between the measured kernel functions. For example, we can evaluate $k(s,t)$ by performing piecewise linear interpolation among the functions $k_1(t), \ldots, k_p(t)$ using a triangulation of 2-space with nodes $s_1, \ldots, s_p$. This corresponds to a choice of nonnegative diagonal matrices D_i so that $D_1 + \cdots + D_p = I$. Higher order interpolation could also be used.

It is clear that matrix-vector products involving these matrices K can be accomplished in about p times the amount of time required for multiplication by a spatially invariant point spread matrix. In the next section we will see that we can do much better than this if the support of each kernel function is reasonably small, resulting in small bandwidth in the Toeplitz blocks.

3. Fast Matrix-Vector Products. In this section we present a scheme for efficiently forming matrix-vector multiplies with K. Algorithms for multiplication by K^T are similar. Since notation for two-dimensional image domains can be a bit cumbersome, we begin by establishing the basic ideas for one-dimensional problems, and then extend these to two-dimensions. Note that the matrix-vector product $D_i K_i \mathbf{z}$ amounts to needing only a portion of the vector $K_i \mathbf{z}$. Thus, the following discussion focuses on portions (blocks of contiguous rows) of Toeplitz matrices and shows how to exploit this structure for fast matrix-vector multiplications.

3.1. Matrix-vector multiplication involving banded Toeplitz matrices. Fast matrix vector multiplication with Toeplitz matrices is accomplished by exploiting the relationship between Toeplitz and circulant matrices. A $\hat{\rho} \times \hat{\rho}$ circulant matrix C is a Toeplitz matrix whose rows and columns are periodic. That is, each row (column) of C is a circular shift of its previous row (column). It is well known (cf. Davis [6]) that the eigenvectors of a circulant matrix are the Fourier vectors, and the eigenvalues can be computed by Fourier transform of the first column of the matrix. Thus, FFTs can be used to compute matrix-vector products of the form $\mathbf{z} = C\mathbf{r}$ in $O(\hat{\rho} \log \hat{\rho})$ operations, using only linear storage. Specifically, if $\mathbf{c}$ is the first column of C, then $\mathbf{z}$ can be computed as

$$\mathbf{z} = \mathtt{ifft}(\mathtt{fft}(\mathbf{c}) \circ \mathtt{fft}(\mathbf{r})),$$

where $\circ$ is used to denote component-wise multiplication and $\mathtt{fft}(\cdot)$ and $\mathtt{ifft}(\cdot)$ are used to denote, respectively, forward and inverse fast Fourier transforms. Optimal efficiency of FFTs is attained when $\hat{\rho}$ is a power of 2.

If T is a $\rho \times \hat{\rho}$ Toeplitz matrix (with $\rho \leq \hat{\rho}$), then products of the form $\mathbf{z} = T\mathbf{r}$ can also be computed in $O(\hat{\rho} \log \hat{\rho})$ operations using FFTs by embedding T into a circulant matrix of larger dimension. Since our focus is on banded matrices, we illustrate this by considering the $\rho \times \hat{\rho}$ banded Toeplitz matrix

$$T = \left[\begin{array}{ccccccccc} t_\beta & \cdots & t_0 & \cdots & t_{-\beta} & & & & \\ & \ddots & \vdots & \ddots & & \ddots & & & \\ & & t_\beta & & \ddots & & t_{-\beta} & & \\ & & & \ddots & & \ddots & \vdots & \ddots & \\ & & & & t_\beta & \cdots & t_0 & \cdots & t_{-\beta} \end{array} \right], \tag{3.1}$$

where β is the bandwidth of T and $\hat{\rho} = \rho + 2\beta$. Then T can be embedded into a $\hat{\rho} \times \hat{\rho}$ circulant matrix

$$C = \left[\begin{array}{c} T \\ W \end{array} \right] \equiv \left[\begin{array}{cccccccc} t_\beta & \cdots & t_0 & \cdots & t_{-\beta} & & & \\ & \ddots & & \ddots & & \ddots & & \\ & & \ddots & & \ddots & & \ddots & \\ & & & t_\beta & \cdots & t_0 & \cdots & t_{-\beta} \\ t_{-\beta} & & & & \ddots & & \ddots & \vdots \\ \vdots & \ddots & & & & \ddots & & t_0 \\ \vdots & \ddots & \ddots & & & & \ddots & \vdots \\ t_{\beta-1} & \cdots & \cdots & t_{-\beta} & & & & t_\beta \end{array} \right]. \tag{3.2}$$

The vector $\mathbf{z} = T\mathbf{r}$ can be computed by forming the circulant matrix-vector product

$$\left[\begin{array}{c} \mathbf{z} \\ \mathbf{w} \end{array} \right] = C\mathbf{r}.$$

Since FFTs are most efficient if the dimensions of the vectors are a power of 2, it is common to embed T in an $m \times m$ circulant matrix, where m is the smallest power of 2 of magnitude at least $\hat{\rho}$. To form $\mathbf{z}$ in this case, we need to pad $\mathbf{r}$ with zeros:

$$\left[\begin{array}{c} \mathbf{z} \\ \mathbf{w} \end{array} \right] = C \left[\begin{array}{c} \mathbf{r} \\ 0 \end{array} \right].$$

We now turn to the 2-D problem. Suppose $\hat{K}$ is a $\rho \times \hat{\rho}$ banded block Toeplitz matrix with $\rho \times \hat{\rho}$ banded Toeplitz blocks:

$$\hat{K} = \left[\begin{array}{cccccccccccc} T_\beta & \cdots & T_1 & T_0 & \cdots & T_{-\beta} & & & & & & \\ & \ddots & \vdots & \vdots & \ddots & & \ddots & & & & & \\ & & T_\beta & T_{\beta-1} & & \ddots & & \ddots & & & & \\ & & & T_\beta & & & \ddots & & T_{-\beta} & & & \\ & & & & \ddots & & & \ddots & T_{1-\beta} & T_{-\beta} & & \\ & & & & & \ddots & & & \vdots & \vdots & \ddots & \\ & & & & & & T_\beta & \cdots & T_0 & T_{-1} & \cdots & T_{-\beta} \end{array} \right], \tag{3.3}$$

with each Toeplitz block $T_l^{(i)}$ having the same form as (3.1).

Matrix-vector multiplications involving a block Toeplitz matrix with Toeplitz blocks, such as $\hat{K}$, can be done efficiently using 2-D FFTs by a straightforward generalization of the 1-D case. That is, $\hat{K}$ is embedded into a $\hat{\rho} \times \hat{\rho}$ block circulant matrix, C, with $\hat{\rho} \times \hat{\rho}$ circulant blocks. If $\mathbf{r}$ is a vector of length $\hat{\rho}^2$, then $\mathbf{z} = \hat{K}\mathbf{r}$ can be computed by using 2-D FFTs to form $C\hat{\mathbf{r}}$ and stripping off the appropriate pieces. Once again, if $\hat{\rho}$ is not a power of 2, we can embed $\hat{K}$ into an $m \times m$ block circulant matrix with $m \times m$ circulant blocks, $m > \hat{\rho}$.

3.2. Matrix-vector multiplication involving Toeplitz-related sums. Now we return to our problem of forming matrix-vector products involving the point spread function

$$K = D_1 K_1 + \cdots + D_p K_p,$$

where each K_i is a banded block Toeplitz matrix.

3.2.1. Piecewise-constant convolution functions. We first consider the piecewise constant case: the jth element of the diagonal matrix D_i is 1 if the jth point is in region i and zero otherwise, and $\sum D_i = I_n$.

To form matrix-vector products $\mathbf{z} = K\mathbf{r}$, one could use the techniques described above to form $\mathbf{z}_i = K_i\mathbf{r}$, and obtain $\mathbf{z}$ as $\mathbf{z} = \sum D_i \mathbf{z}_i$. However, a substantial amount of work can be saved by taking better advantage of our partitioning of the image domain Ω. Suppose, for illustration, that we have partitioned our image domain into $p = 9$ rectangular pieces:

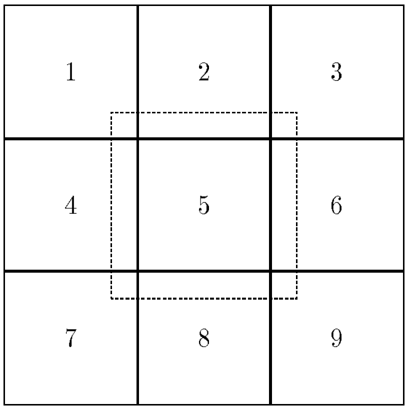


FIG. 3.1. *Example of image domain partitioning.*

Then the product $D_5 K_5 \mathbf{r}$, for example, depends on the values of $\mathbf{r}$ in region 5, as well as on values in other regions within a width β of the borders of region 5. This domain of dependence is indicated by the dotted borders. Further, if we reorder the elements of $\mathbf{r}$ so that those within the dotted borders are grouped together and ordered row-by-row (or, alternatively, column-by-column) then the matrix formed from the nonzero columns of the matrix $D_5 K_5$ has exactly the form of the matrix $\hat{K}$ in (3.3).

Thus, the matrix-vector product can be formed by applying the $\hat{K}$ algorithm over each of the regions (using zero-padding along the exterior borders), and concatenating the resulting vectors.

There are several advantages to this approach:

- **Storage:** To perform matrix-vector multiplies using FFTs, we need to store the eigenvalues of the extended block circulant matrices corresponding to each region. This can be done in $\hat{\rho} \times \hat{\rho}$ dimensional arrays. If the bandwidth $\beta \leq \rho/2$, which is often the case in image restoration, then $\hat{\rho} \leq 2\rho$. Thus, an upper bound on storage requirements is p arrays of size $2\rho \times 2\rho$, which is equivalent to one array of size $2n \times 2n$. To illustrate the savings here, note that K is an $n^2 \times n^2$ matrix. Our storage requirement for a spatially variant kernel (bandwidth satisfying $\beta \leq \rho/2$) is no more than that needed for a spatially invariant one!
- **Work:** The cost in forming a matrix-vector multiplication for a general spatially invariant kernel (using FFTs) is $O(n^2 \log n)$. In the scheme presented in this section for spatially varying kernels, we need to form p $\hat{\rho}^2$-dimensional spatially invariant products. Thus the total cost is $O(p\hat{\rho}^2 \log \hat{\rho})$.
  If we consider a fixed continuous kernel discretized with different values of n, then the bandwidth β of the matrix grows linearly with n, so $\hat{\rho} = \rho + 2\beta$ also grows linearly and, since $p\rho^2 = n^2$, we must have that $p\hat{\rho}^2$ is bounded by some constant times n^2, so the complexity as $n \to \infty$ is $O(n^2 \log n)$, the same order as that of a spatially invariant kernel.
- **Parallelism:** This algorithm has inherent parallelism that is worth applying even to the more simple case of spatially invariant kernels. In this case, the region can be partitioned into congruent pieces and all of the matrices K_i are equal. The matrix-vector multiplies for each piece can be performed independently, once information about the overlap regions is exchanged. The number of pieces is limited by the need to keep $\beta < \rho/2$, but each piece can be spread over multiple processors.

3.2.2. Linear interpolation. Suppose the convolution kernel is approximated by a piecewise linear function (rather than a piecewise constant one). Suppose, for illustration, that we have measured the convolution function at 9 points, the vertices of the triangles in Figure 3.2.

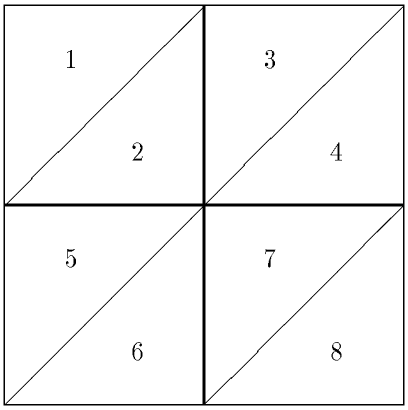


FIG. 3.2. *Example of linearly interpolated regions*

Then the convolution at the jth point is determined as a weighted average of at most three spatially invariant convolutions, those corresponding to the three vertices of the triangle containing point j, or those corresponding to the two endpoints of its line segment if point j happens to fall on a boundary.

To form matrix-vector products $\mathbf{z} = K\mathbf{r}$, we partition our domain into overlapping subdomains defining the regions of influence of each of the spatially invariant convolutions. The measured convolution function corresponding to the center point in the figure, for example, must be applied over regions 2, 3, 4, 5, 6, and 7. Matrix-vector multiplication is then performed as in §3.1: each region of influence is embedded into a rectangle. (For efficiency, it might be better to partition the region of influence and embed into multiple rectangles.) Weighted sums of the resulting point values then give the desired matrix-vector products.

4. Preconditioning. In general, an effective preconditioning scheme for a conjugate gradient iteration is one in which the preconditioner matrix C is a good approximation to the matrix K; that is, the singular values of KC^{-1} are clustered around one. For ill-posed problems, however, the preconditioner matrix is likely to be severely ill-conditioned, and early iterations will be highly contaminated with noise. To avoid this, we take the approach suggested by Hanke, Nagy, and Plemmons [8]. Specifically, the aim is to construct a matrix C that clusters the large singular values (*i.e.*, approximates K on the signal subspace), while leaving the small singular values (noise subspace) unchanged. As is shown in [8], this can be done if a spectral decomposition of C is available. This is the case for matrices that are block circulant with circulant blocks. To simplify notation in this section, we assume the matrices K_i are $n \times n$ block Toeplitz matrices with $n \times n$ Toeplitz blocks. The notation $\mathrm{BTTB}(n)$ is used to represent such matrices. Similarly, $\mathrm{BCCB}(n)$ will be used to denote matrices that are $n \times n$ block circulant with $n \times n$ circulant blocks.

4.1. Preconditioning piecewise-constant convolution functions. There are several possible schemes for approximating a $\mathrm{BTTB}(n)$ matrix with one that is $\mathrm{BCCB}(n)$; see, for example, the recent survey paper by Chan and Ng [4]. In our work, we consider the one proposed by Chan and Olkin [5], which is simply the best Frobenius norm approximation over all $\mathrm{BCCB}(n)$ matrices. This is the approach used in [8] for spatially invariant point spread functions, and we discuss next the necessary modifications to make it applicable to the spatially varying kernels we are considering in this paper.

Several options exist for approximating K. For example, if we have a parametric model that relates K_i, for each value of n, to a block Toeplitz matrix with Toeplitz blocks, then this can be used to construct C.

Although this situation may occur in some applications, it may be difficult to obtain the necessary parametric model. Another approach is to form a weighted average of the K_i, from which C can be constructed. The weights could be determined by the size of the image pieces influenced by K_i, or by weighting "important" regions of the image more heavily.

A third approach is to construct $\mathrm{BCCB}(n)$ approximations of K restricted to each region. That is, suppose each region is $\rho \times \rho$, and define

$$\tilde{K} = D_1 K_1 D_1 + D_2 K_2 D_2 + \cdots + D_p K_p D_p. \tag{4.1}$$

If the equations and unknowns are appropriately ordered, $\tilde{K}$ has the form

$$\tilde{K} = \begin{bmatrix} \tilde{K}_1 & 0 & \cdots & 0 \\ 0 & \tilde{K}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{K}_p \end{bmatrix},$$

where each $\tilde{K}_i$ is a BTTB(ρ) matrix. We use this matrix to construct a preconditioner as follows. Let C_i denote the optimal BCCB(ρ) approximation to $\tilde{K}_i$, and define C as

$$C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_p \end{bmatrix}.$$

Since each C_i is BCCB(ρ), we can write $C_i = \mathcal{F}^* \Lambda_i \mathcal{F}$, where $\mathcal{F}$ is the unitary 2-D Fourier transform matrix. Hence, a spectral decomposition of C is given by

$$C = (I \otimes \mathcal{F})^* \begin{bmatrix} \Lambda_1 & & & \\ & \Lambda_2 & & \\ & & \ddots & \\ & & & \Lambda_p \end{bmatrix} (I \otimes \mathcal{F}).$$

Using this spectral decomposition, we can apply the technique suggested in [8] for separating the signal and noise subspaces: a particular truncated spectral decomposition of C, which we denote as C_τ, is constructed and used as a preconditioner. Specifically, a truncation parameter τ is chosen, and the eigenvalues of C with magnitude less than τ are replaced by 1.

In order to show that C_τ is a good preconditioner, we need to show that the large singular values of KC_τ^{-1} are clustered around 1, while the small singular values remain unchanged (*i.e.*, cluster around 0). To get to this point, we first need to consider the difference $K - \tilde{K}$, which depends on the differences $D_iK_i - D_iK_iD_i$. Note that premultiplication with D_i ensures that this difference is zero in all rows corresponding to variables not in region i. Using (3.3), we see that in the rows corresponding to region i, nonzeros occur only in columns corresponding to points in the neighboring regions that are coupled to the ith region. Referring to Figure 3.1, this means that the nonzeros for block 5, for example, occur only for those points that are outside the boundary of region 5 but inside the dotted borders. Since there are fewer than $4\beta\hat{\rho}$ of these points, there are at most $4\beta\hat{\rho}$ nonzero columns, and we have

$$\operatorname{rank}(D_iK_i - D_iK_iD_i) \leq 4\beta\hat{\rho}.$$

Assuming that $p\hat{\rho}^2 < \gamma^2 n^2$ for some constant γ, the following lemma has been established.

LEMMA 4.1. *Let K be given as in (2.1) and $\tilde{K}$ be defined by (4.1) with $p\hat{\rho}^2 < \gamma^2 n^2$. Then*

$$K - \tilde{K} = U,$$

where rank$(U) \leq 4\beta\gamma n\sqrt{p}$.

We note that in most image processing applications, the extent of the point spread function is small compared to the image dimensions; thus, $\beta \ll n$. Even if β is large, it is often the case that the point spread function is large only within a small radius of the center point, meaning that the large nonzero elements in the matrix are few in number. In this case, we would obtain results of the form

$$K - \tilde{K} = U + V,$$

where V has small norm, and $\mathrm{rank}(U) = O(\hat{\beta} n)$, with $\hat{\beta} \ll n$. In view of these remarks, we assume, as is done in [8] for the spatially invariant case, that the entries $k^{(i)}_{\mu,\nu}$ of K_i are obtained from an infinite sequence $\{k^{(i)}_{\mu,\nu}\}$ satisfying

$$\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} |k^{(i)}_{\mu,\nu}| \leq M < \infty.$$

Since the entries of $\tilde{K}_i$ come from these same generating sequences, the following lemma holds (cf. [3, Corollary 1]).

LEMMA 4.2. *For all $\varepsilon > 0$, there exists an $N > 0$, such that for all $n > N$,*

$$\tilde{K}_i - C_i = U_i + V_i,$$

where $\mathrm{rank}(U_i) = O(\rho)$ and $||V_i|| < \varepsilon$.

Using the above lemma, the following corollary is easily established.

COROLLARY 4.3. *For all $\varepsilon > 0$, there exists an $N > 0$, such that for all $n > N$,*

$$\tilde{K} - C = U + V,$$

where $\mathrm{rank}(U) = O(n)$ and $||V|| < \varepsilon$.

We now say how well C approximates K.

THEOREM 4.4. *For all $\varepsilon > 0$, there exists an $N > 0$, such that for all $n > N$,*

$$K - C = U + V,$$

where $\mathrm{rank}(U) = O(n)$ and $||V|| < \varepsilon$.

Proof. Let $\varepsilon > 0$ be given. Then, from Corollary 4.3 and the remarks following Lemma 4.1, there exists an $N > 0$ such that for all $n > N$,

$$\begin{aligned} K - C &= K - \tilde{K} + \tilde{K} - C \\ &= U_1 + V_1 + U_2 + V_2 \\ &= U + V, \end{aligned}$$

where

$$\mathrm{rank}(U) \equiv \mathrm{rank}(U_1 + U_2) = O(n)$$

and

$$||V|| \equiv ||V_1 + V_2|| < \varepsilon.$$

□

The above theorem implies that, asymptotically, C is a good approximation to K. To show C_τ is a good preconditioner, we establish the previously discussed clustering property of the singular values in the following theorem.

THEOREM 4.5. *Let a tolerance $\tau > 0$ be given, and define C_τ as above. Then given $\varepsilon > 0$, there exists an $N > 0$ such that for all $n > N$, at most $O(n)$ singular values of KC_τ^{-1} lie outside the interval*

$$(1 - \varepsilon - \tau, 1 + \varepsilon + \tau) \cup [0, \varepsilon + \tau).$$

The details of the proof follow exactly the same lines as those given in [8, Thm. 6.1], and are therefore omitted. As in [8], for moderately ill-posed problems with little noise, we can take $\tau \approx 0$. In this case, standard convergence analysis of the conjugate gradient algorithm implies that at most $O(n)$ iterations are needed to compute an accurate solution, rather than the $O(n^2)$ implied by the size of the problem. We note that in our numerical tests (see Section 5) the preconditioned iteration converged in far fewer iterations than predicted by this analysis.

4.1.1. Preconditioning piecewise linear convolution functions. In the case of piecewise linear convolution, we have

$$\tilde{K} = \tilde{D}_1 K_1 D_1 + \tilde{D}_2 K_2 D_2 + \cdots + \tilde{D}_p K_p D_p. \tag{4.2}$$

where D_i is still a segment of the identity, but the $\tilde{D}_i$ are general nonnegative diagonal matrices that sum to the identity.

We still get

$$\tilde{K} = \begin{bmatrix} \tilde{K}_1 & 0 & \cdots & 0 \\ 0 & \tilde{K}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{K}_p \end{bmatrix},$$

but now each $\tilde{K}_i$ is a weighted sum of three $\text{BTTB}(n)$ matrices.

Each $\tilde{K}_i$ needs to be approximated by a $\text{BCCB}(\rho)$ matrix so that the difference has small rank plus small norm. This can be done if each matrix K_i differs from its *neighboring* ones by a matrix of small rank plus a matrix of small norm. In this case, Lemma 4.2 applies. We can construct the circulant approximation on each subregion based on one of the three vertex kernels or on an average of the three. We obtain the following result.

THEOREM 4.6. *Suppose that there exists an $N > 0$ such that for all $n > N$ and for all $\epsilon > 0$, for each pair of neighboring vertices i and j in the triangulation of Fig. 3.2, the corresponding kernel matrices K_i and K_j satisfy*

$$D_k(K_i - K_j) = U_{ij} + V_{ij}$$

where D_k is a diagonal matrix with ones corresponding to rows in the regions containing the two vertices and zeros elsewhere, and where rank$(U_{ij}) = O(\rho)$ and $\|V_{ij}\| < \epsilon$. Then we can construct a block circulant matrix C so that

$$K - C = U + V,$$

where rank$(U) = O(n)$ and $\|V\| < \varepsilon$.

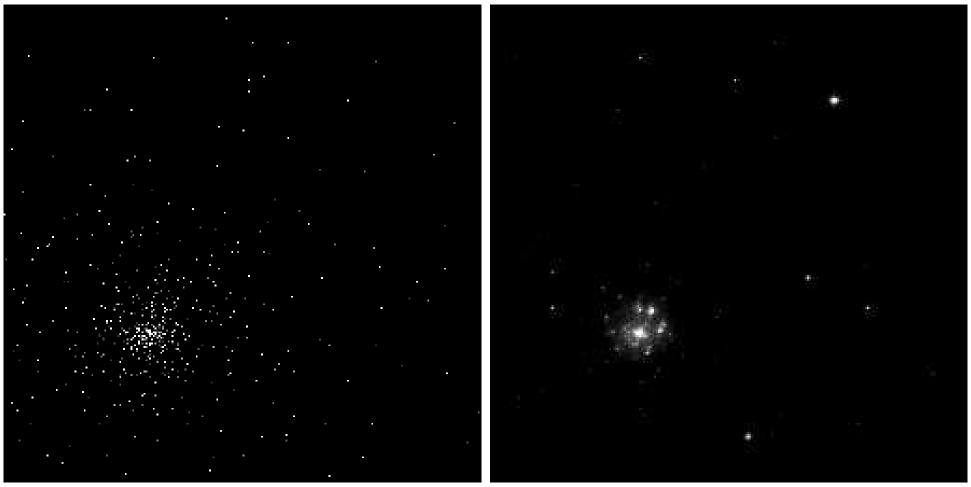
5. Numerical Results. In this section we present numerical results using conjugate gradients (CG) and preconditioned conjugate gradients (PCG) for image deblurring. We illustrate the effectiveness of using multiple point spread functions, and we study the behavior of our preconditioning scheme.

In particular, we have observed that using a preconditioner in restoring smooth images does not reduce the number of iterations necessary to reach a given relative error tolerance. However, if the original image has many rough contours, then the preconditioning can be very effective. These properties are illustrated in the second example, using a checkerboard-patterned image with varying block sizes. We begin, though, with a more realistic problem used to study the importance of accounting for spatial dependence in the reconstruction of Hubble Space Telescope data.

All tests were performed on images of size 256×256 using Matlab on a DEC Alpha 3000/600. Displays were generated using Xv. The truncation parameter τ for the preconditioner was always taken to be zero.

An Astronomy Example: As mentioned in Section 1, the errors in shaping the mirrors of the Wide-Field Planetary Camera for the Hubble Space Telescope (HST) resulted in image degradation with a large amount of spatial variation. Although repairs to the HST have been made, the importance of restoring older images has not diminished. For example, images of particular regions of space taken at several different times are used by astronomers to determine the distances and speeds of moving objects such as stars and galaxies.

In this first example, we apply our scheme to data used by astronomers to test and compare image restoration algorithms for HST images; see for example the Newsletter of the STScI's Image Restoration Project [7], and Katsaggelos, Kang and Banham [10]. This data, obtained via anonymous ftp from `ftp.stsci.edu`, in the directory `/software/stsdas/testdata/restore/sims/star_cluster`, is intended to simulate a star cluster image taken by the HST before the camera was fixed. Figure 5.1 displays the true star cluster, and the image as would be given by the HST.[1]



Original image. Observed HST image.

FIG. 5.1. *Original and HST simulated images.*

The BTTB(n) matrices K_i can be determined from the known properties of the camera or, somewhat more reliably, can be constructed experimentally from astronomical data. Each K_i represents the effect of imaging a point source in the region defined by D_i. We approximate the point source by an isolated star in a particular region. Figure 5.2 is a mesh plot of one of these point source images after normaliza-

[1] In order to avoid oversaturation in the star cluster pictures, each pixel value v was truncated to $\min(\max(v, 50), 500)$.

tion. If we unstack this image row-wise, we obtain the central column of the banded BTTB(n) matrix K_i. The bandwidth of K_i is determined by the extent of the point source image (the number of nonzero pixel values) and by the size of the image that is to be restored.

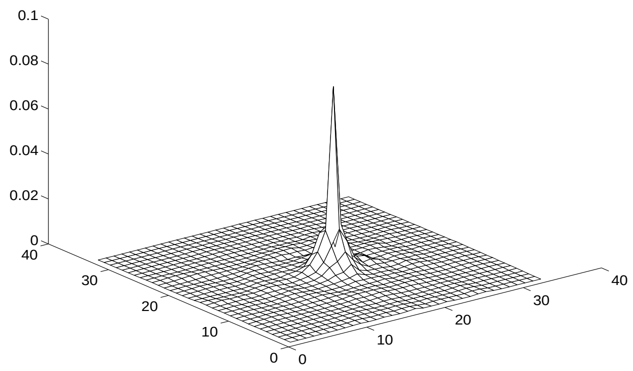


FIG. 5.2. *Mesh plot of the point source image* PSF*06 (normalized).*

The degraded image in Figure 5.1 was obtained by convolving each of the 470 stars with a different point spread function, and adding both readout noise (essentially white Gaussian) and Poisson noise to the blurred image; see the README file at the abovementioned ftp site for details. Although 470 PSFs were used to generate the blurred image, only 25 are available. These PSFs are evenly distributed in the image domain, centered at positions $(28+50*k, 28+50*l)$, $k, l = 0, 1, 2, 3, 4$. We denote them as PSF00, PSF01, ..., PSF24. Figure 5.3 shows the image domain and the locations of the PSFs.

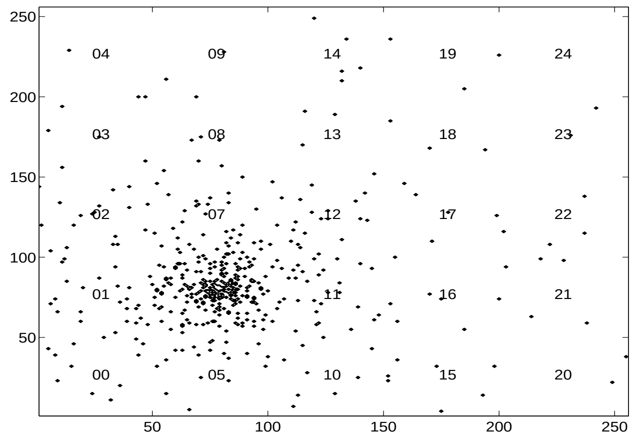


FIG. 5.3. *Locations of the 25 PSFs in the image domain.*

Since we are given 25 PSFs in various regions of the image, we can use any number of them in our tests. Here we report numerical results for the following piecewise-constant cases:

- **One PSF:** Since most restoration algorithms are based on spatially invariant PSFs, we used CG and PCG to compute restorations using only one PSF. The best solution we were able to obtain using a single PSF occurred with PSF11, and the worst solution was computed using PSF04. We emphasize that, in general, a priori knowledge of which single PSF produces the best restoration would not be known.
- **Four PSFs:** Restorations using CG and PCG were computed using the four PSFs PSF06, PSF08, PSF16, and PSF18. This corresponds to dividing the domain into four 128×128 regions, each having a single PSF. We note, though, that matrix-vector multiplications (as discussed in Section 3) can be done on smaller sized regions. In our computations, we used 64×64 regions in the multiplications.
- **All PSFs:** Finally, restorations using CG and PCG were computed using all 25 PSFs. In this case the image does not divide into uniformly sized regions. For our computations, we included 52 rows in the five regions regions along the bottom edge of the image (see Figure 5.3); all others had 51. Similarly, regions along the right edge had 52 columns, and all others had 51.

Because the true image is available, we are able to compute the relative error in our results by taking the Frobenius norm of the difference between the true and computed images and dividing by the norm of the true image. Figure 5.4 plots the relative errors vs. number of iterations for CG and PCG. These plots and the resulting images shown in Figure 5.5 illustrate that much better restorations can be obtained by using multiple PSFs. For example, the bright star in the upper right corner of the image appears to be a star cluster when only one PSF is used, but is rendered more accurately when we allow spatial variation in the PSFs. The preconditioner substantially reduces the number of iterations needed to compute a good restoration, but as the theory predicts (see Section 4), the more BTTB(n) matrices K_i that comprise K, the less effective is the preconditioner.

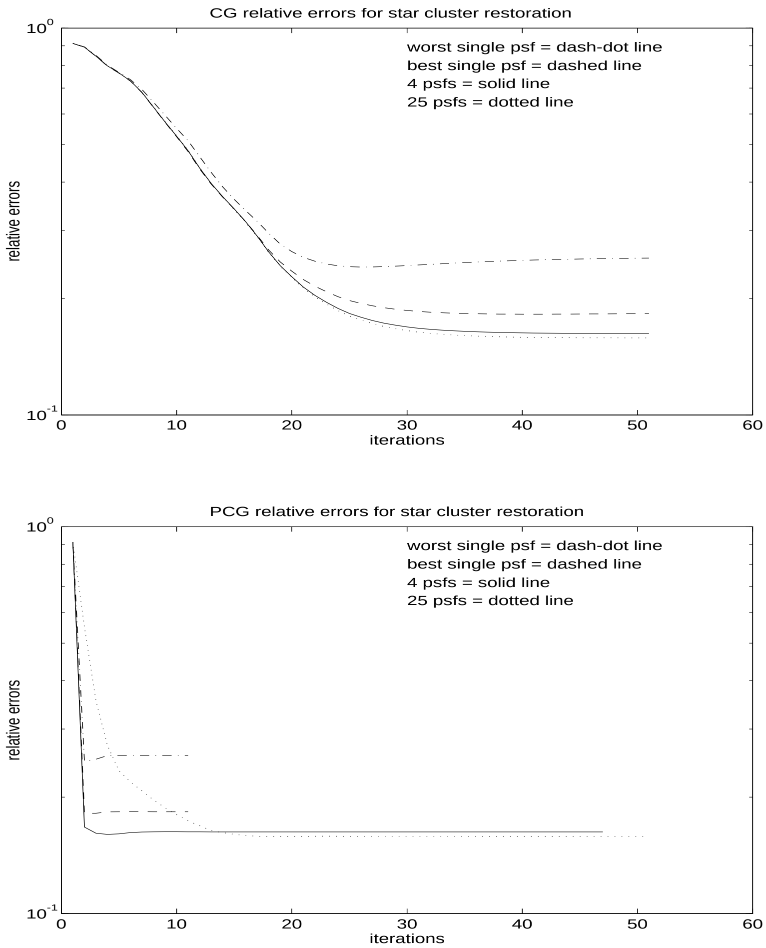
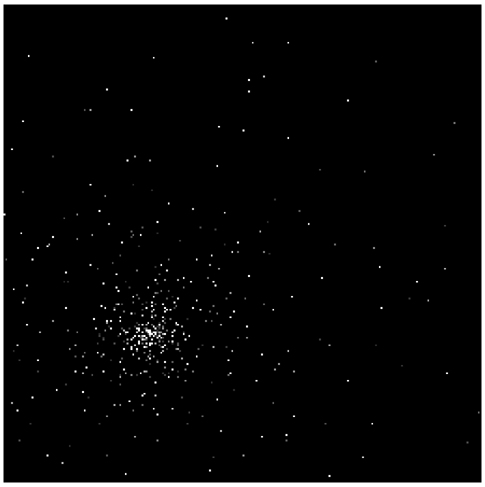
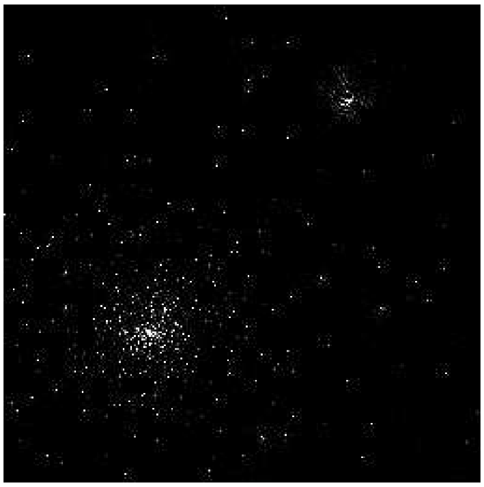


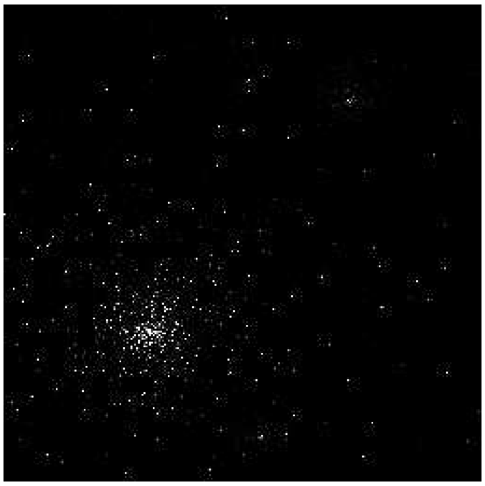
FIG. 5.4. *Relative errors using CG and PCG.*



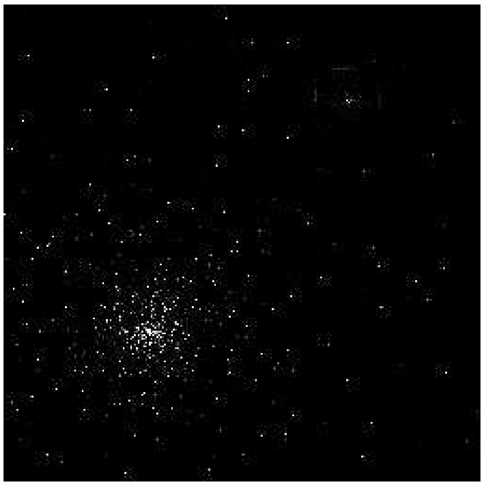
True image.



Best single PSF solution (2 iterations).



Solution using 4 PSFs (3 iterations).



Solution using 25 PSFs (39 iterations).

FIG. 5.5. *Comparisons of PCG computed solutions.*

Checkerboard Example: The star cluster example shows that using several point spread functions can considerably improve restoration. Additionally, we see that preconditioning can have a dramatic effect on convergence rates. The star images have a large amount of high frequency information, so the point of our experiments with checkerboard images was to determine the effectiveness of the preconditioner on images that are progressively smoother: a checker board image with varying block sizes. Figure 5.6 shows the images and their blurred versions. The dimensions of the small blocks in the images are 4×4 pixels, 16×16 and 32×32. The degraded images were constructed by dividing the images into four regions of size 128×128, and applying one of the point spread functions PSF06, PSF08, PSF16, and PSF18 to each of the regions. Normally distributed random noise was added, scaled so that the Frobenius norm of the noise was 0.001 times the Frobenius norm of the blurred image.

We used CG and PCG, with four PSFs and the preconditioner described in Section 4, to restore each of the checkerboard images. The relative error plots are shown in Figure 5.7. We see that for smooth images (block size = 32), the preconditioner does not perform well. However, for rougher images (block size = 4), the preconditioner is very effective in reducing the number of iterations needed to compute a good restoration. Computed solutions are shown in Figure 5.8. Since the error is monotonically decreasing as the number of iterations increases, we chose to avoid the question of termination criteria and simply compare solutions that resulted from approximately equal amounts of work: the PCG solutions at iteration 14 and the CG solutions at iteration 20. Using the error plots in Figure 5.7 as a guide, for block size = 4 we expect the 14 iterations of PCG to provide a better restoration than 20 iterations of CG. Although at first glance this does not appear to be the case, enlargements of a 32×32 central portion of the two solutions, shown in Figure 5.9, reveals that indeed a better restoration is obtained by PCG.

6. Final Remarks. We have developed effective iterative methods for solving convolution problems in which the matrix has piecewise-constant or piecewise-linear spatial variance. This scheme has inherent parallelism, and in future work we will develop parallel algorithms based on these ideas.

7. Acknowledgements. We are grateful to Douglas Currie and Bert Rust for introducing us to astronomical image processing.

REFERENCES

[1] J. Biretta, *WFPC and WFPCC 2 Instrumental Characteristics*, in The Restoration of HST Images and Spectra II, R. J. Hanisch and R. L. White, eds., Space Telescope Science Institute, Baltimore, MD, 1994, pp. 224–235.

[2] Å. Björck, *Least squares methods*, in Handbook of Numerical Analysis, P. Ciarlet and J. Lions, eds., vol. 1, Amsterdam, 1989, Elsevier/North Holland.

[3] R. H. Chan, J. G. Nagy, and R. J. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740–1768.

[4] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, preprint, (1995).

[5] T. F. Chan and J. A. Olkin, *Preconditioners for Toeplitz-block systems*, Numer. Algor., 6 (1993), pp. 89–101.

[6] P. J. Davis, *Circulant Matrices*, Wiley, New York, 1979.

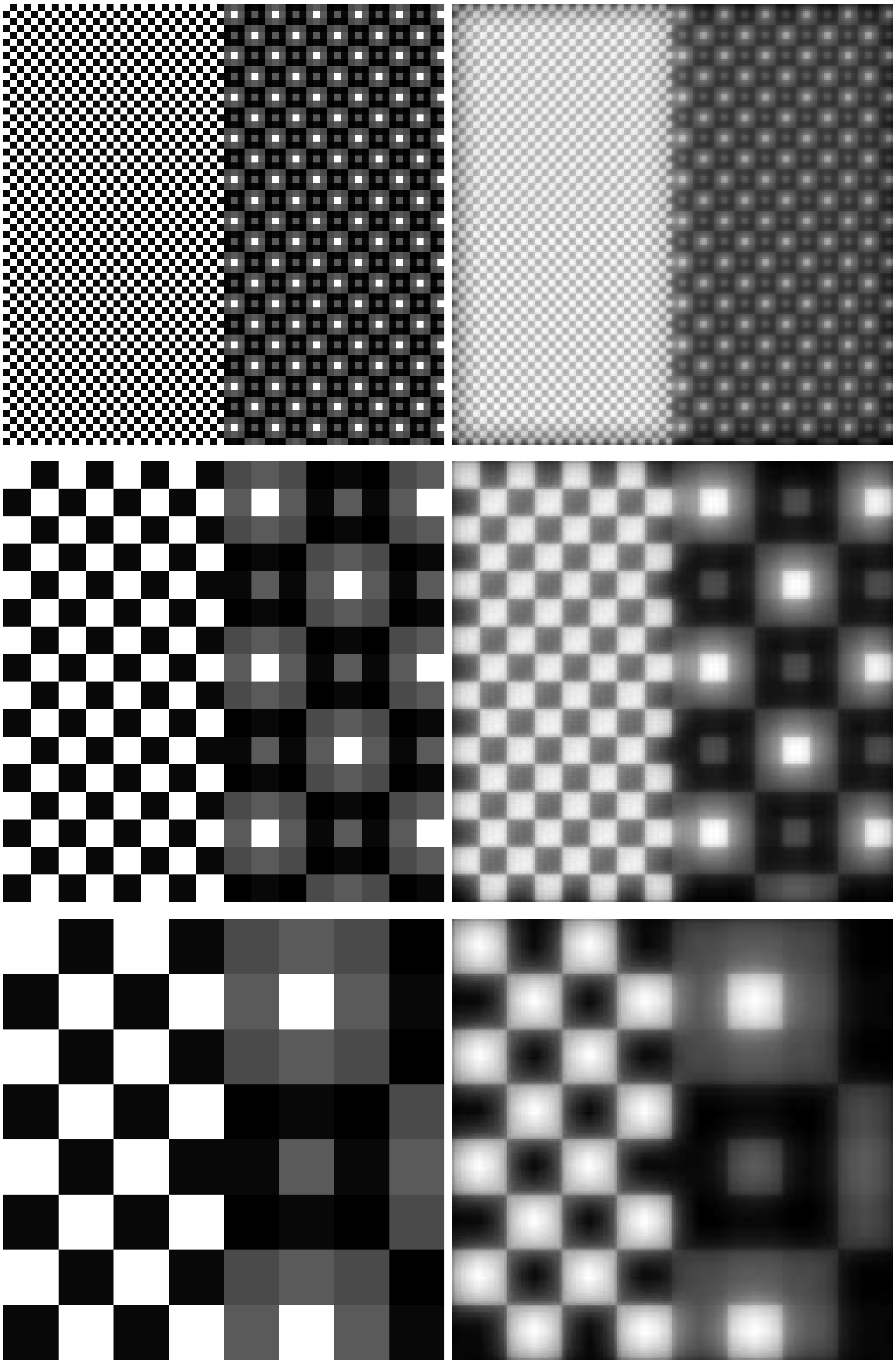


FIG. 5.6. *Original and degraded checker board images.*

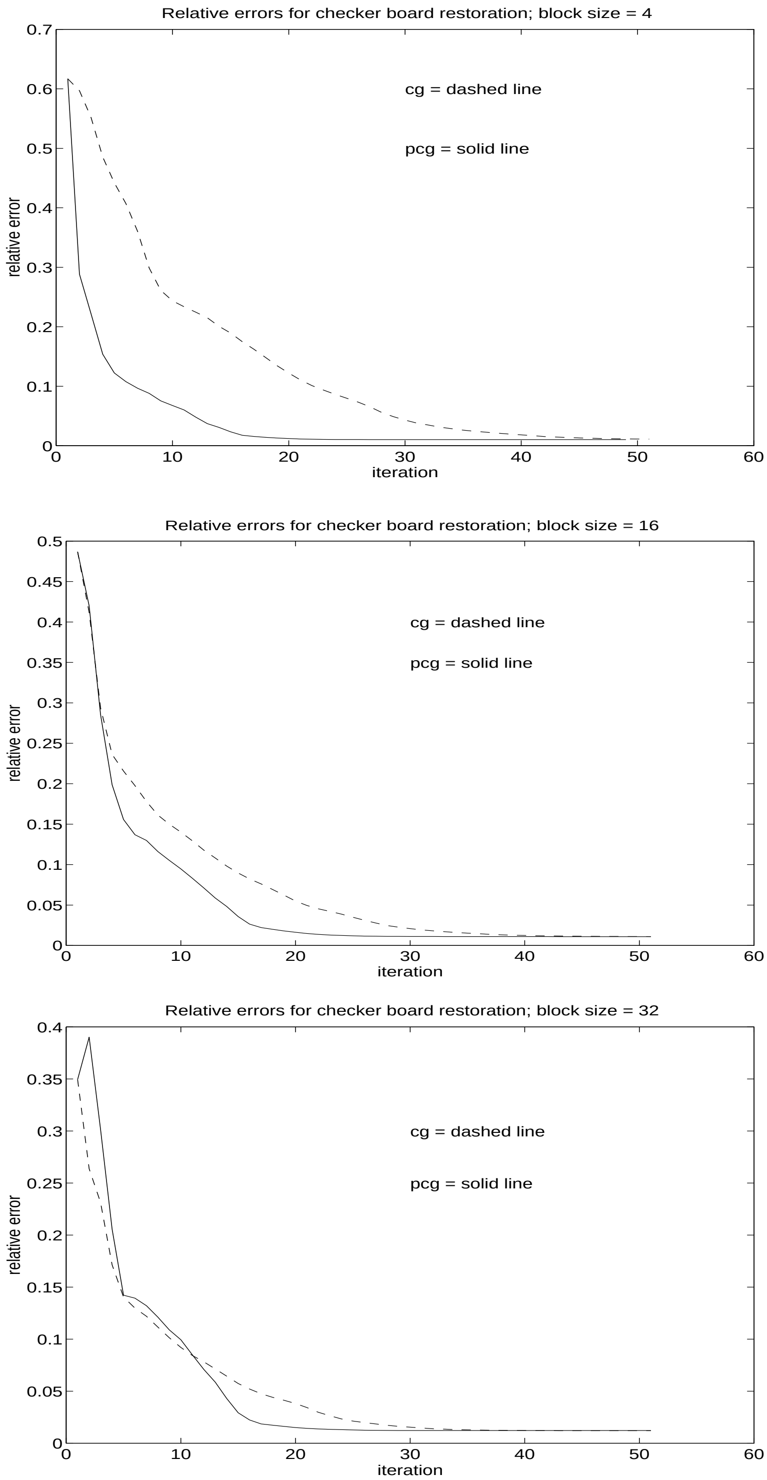


FIG. 5.7. *CG and PCG relative errors for checker board images.*

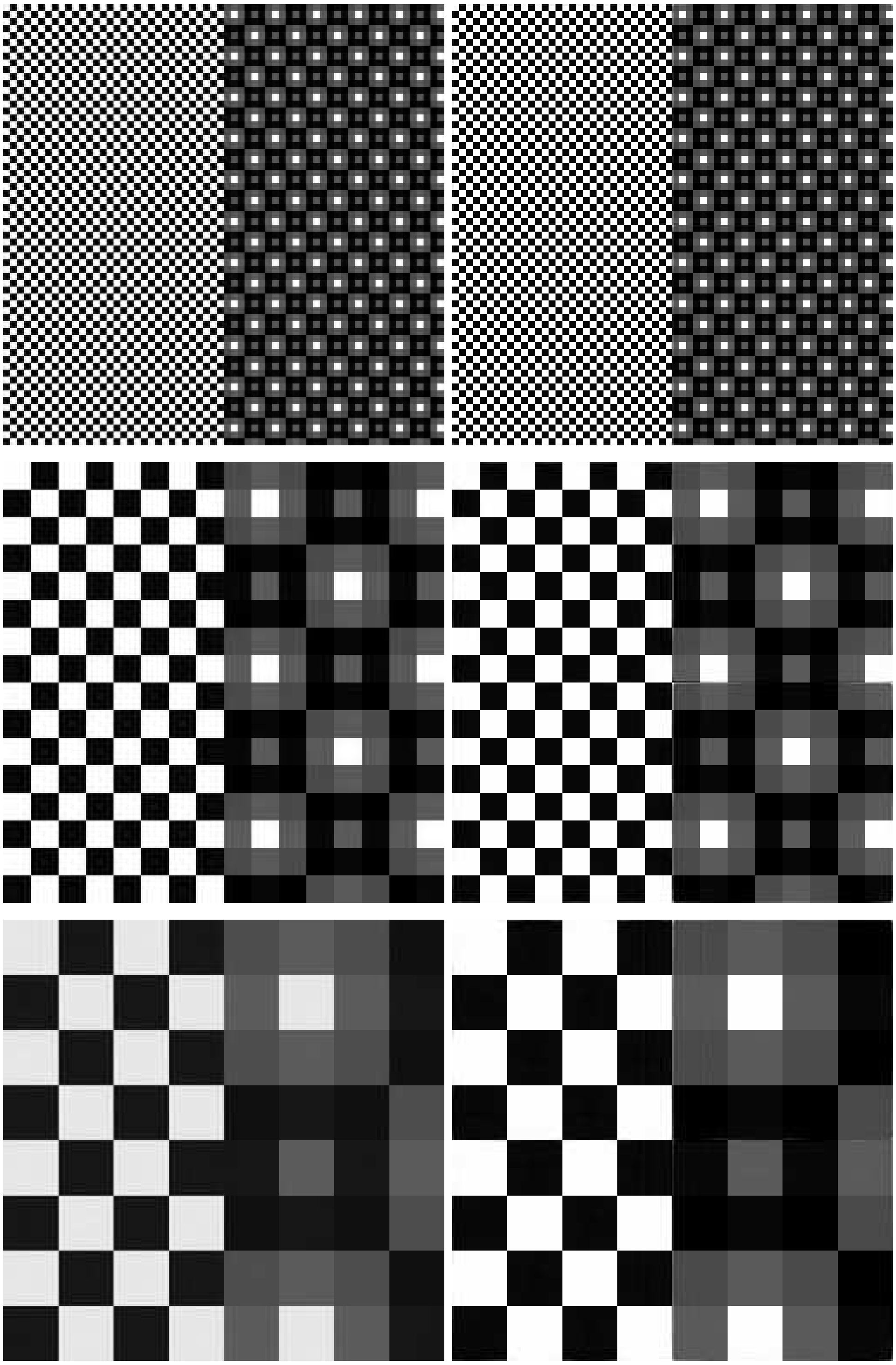


FIG. 5.8. *Computed solutions after 20 iterations of CG (left) and 14 iterations of PCG (right) for each checker board image.*

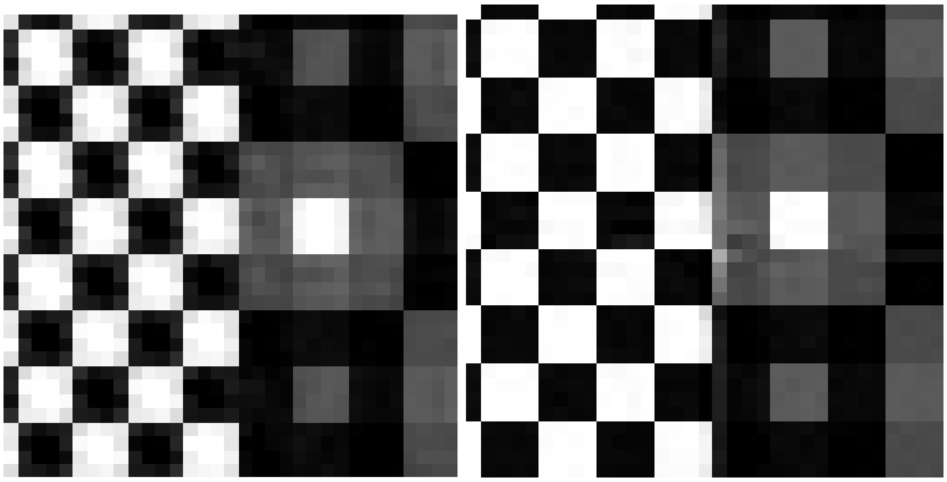


FIG. 5.9. *Enlargement of a 32×32 central portion of CG (left) and PCG (right) restorations for block size = 4*

[7] R. J. HANISCH, *WF/PC simulation data sets*, in Newsletter of STScI's Image Restoration Project, R. J. Hanisch, ed., Summer 1993, pp. 76–77.

[8] M. HANKE, J. G. NAGY, AND R. J. PLEMMONS, *Preconditioned iterative regularization*, in Numerical Linear Algebra, L. Reichel, A. Ruttan, and R. S. Varga, eds., de Gruyter, Berlin, 1993, pp. 141–163.

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

[10] A. K. KATSAGGELOS, M. G. KANG, AND M. R. BANHAM, *Adaptive regularized restoration algorithms applied to HST images*, in The Restoration of HST Images and Spectra II, R. J. Hanisch and R. L. White, eds., 1994.

[11] A. E. SAVAKIS AND H. J. TRUSSELL, *Blur identification by residual spectral matching*, IEEE Trans. Image Processing 2 (1993), pp. 141–151.