

# A Limited Memory, Quasi-Newton Preconditioner for Nonnegatively Constrained Image Reconstruction

Johnathan M. Bardsley

*Department of Mathematical Sciences, The University of Montana, Missoula, MT*

*59812-0864*

Image reconstruction gives rise to some challenging large-scale constrained optimization problems. In this paper we consider a convex minimization problem with nonnegativity constraints that arises in astronomical imaging. To solve this problem, an efficient hybrid gradient projection-reduced Newton (active set) method is used. By “reduced Newton” we mean we take Newton steps only in the inactive variables. Due to the large size of our problem, we compute approximate reduced Newton steps using the conjugate gradient (CG) iteration. We introduce a limited memory, quasi-Newton preconditioner that speeds up CG convergence. A numerical comparison is presented that demonstrates the effectiveness of this preconditioner. © 2003 Optical Society of America

*OCIS codes:* 100.1830, 100.3010, 100.3190.

## 1. Introduction

Astronomical images obtained from a ground based telescope are blurred due to random variations in the index of refraction of the atmosphere. These blurred images are further degraded by noise that enters the data during image collection. Astronomical image data is typically collected with a device known as a CCD camera, and the corresponding statistical model for the noise is well known.<sup>1</sup> Data for this problem takes the form

$$\mathbf{d} = S\mathbf{f}_{\text{true}} + \boldsymbol{\eta}, \quad (1)$$

where  $S$  is an  $N \times N$ , non-sparse, block Toeplitz, ill-conditioned matrix that characterizes atmospheric blur, and  $\mathbf{d}$ ,  $\mathbf{f}_{\text{true}}$ , and  $\boldsymbol{\eta}$  are  $N \times 1$  vectors. Here  $\mathbf{d}$  is the data;  $\mathbf{f}_{\text{true}}$  is the true image (or object) and is nonnegative;  $\boldsymbol{\eta}$  represents the noise that enters the data during the collection of the image by the CCD camera;  $N$  denotes the number of pixels in the collected image.

To accurately reconstruct  $\mathbf{f}_{\text{true}}$ , we solve an optimization problem of the form

$$\min_{\mathbf{f} \geq \mathbf{0}} J(\mathbf{f}), \quad (2)$$

where  $\mathbf{f} \geq \mathbf{0}$  means  $f_i \geq 0$  for each  $i$ . The nonnegativity constraints arise from the knowledge that, since the true image consists of positive light intensities,  $\mathbf{f}_{\text{true}}$  is nonnegative. The function  $J$  is strictly convex and depends upon prior knowledge about the noise statistics.

The standard approach taken in solving (2) when the noise has a Poisson distribution is the Richardson-Lucy (RL) Algorithm.<sup>2,3</sup> The analogue of RL for the case where the noise is Gaussian with constant variance across pixels is the MRNSD al-

gorithm.<sup>4</sup> Because  $S$  in (1) is an ill-conditioned matrix, some form of regularization is required when solving (2). For RL and MRNSD, this involves stopping the iteration before noise amplification occurs. This approach is often referred to as iterative regularization.

A different approach for solving (2) is to incorporate a penalty, or regularization, term into the function  $J$  for stability and to solve (2) via an effective optimization algorithm. When the noise statistics are well-approximated by a Poisson distribution, the gradient projection-reduced Newton-conjugate gradient (GPRNCG) algorithm has been shown to be effective and is competitive with RL.<sup>5</sup> GPRNCG is an extension of the gradient projection-conjugate gradient (GPCG) algorithm for bound constrained, quadratic optimization.<sup>6</sup> When the noise in the data is well-approximated by a Gaussian random vector with constant variance across pixels, the corresponding function  $J$  is quadratic, and hence GPCG is appropriate. Both GPRNCG and GPCG intersperse gradient projection (GP) iterations<sup>7,8</sup> with conjugate gradient (CG) iterations.<sup>9</sup> The convergence properties of GPRNCG (and analogously GPCG) can be improved dramatically if an effective preconditioner is used during CG iterations.<sup>5</sup> In this paper, we will focus on improving the performance of GPRNCG via preconditioning the imbedded CG iterations.

At the  $k$ th GPRNCG iteration, CG is used to approximately minimize a quadratic function  $q_k$ , where  $q_k$  is the quadratic Taylor approximation of  $J$  at the most recent iterate restricted to the inactive indices (see Section 3 for details). Dramatic changes in the set of inactive indices from iteration to iteration result in dramatic changes in the form of the Hessian of  $q_k$ . This greatly increases the difficulty of finding a good

preconditioner for CG, since in order for the preconditioner to be effective, its form must change as the form of the Hessian of  $q_k$  changes. Preconditioning strategies for the closely related GPCG algorithm are discussed in Ref. 10. Here it is stated that "[t]he only possible preconditioning ... is diagonal preconditioning". In our experience, we have found diagonal preconditioners to be ineffective, but, despite the above statement, we have found that there *are* effective non-diagonal preconditioners. In Ref. 5, a sparse preconditioner is presented and is effective for use with both GPCG and GPRNCG. Unfortunately, the use of this preconditioner requires the solution of a linear system involving the preconditioning matrix. In order to take full advantage of the sparsity structure of the preconditioning matrix, a sparse Choleski factorization with a reordering scheme is used. This preconditioner is very effective, but is expensive, particularly for large  $N$ . In addition, building this preconditioner requires that one obtain a sparse approximation of the blurring matrix  $S$ . This sparse approximation will in turn depend upon the PSF  $s$ , and for certain PSFs, will be too expensive to implement. For the case in which implementation of the sparse preconditioner is deemed too expensive, we introduce a limited memory, quasi-Newton preconditioner for use with the GPRNCG algorithm. This preconditioner is effective and is inexpensive to implement. In addition, the cost of implementation does not depend upon the PSF.

This paper is organized as follows: In Section 2 we present an integral equation used to model image formation. We also discuss discretization of this equation as well as the statistics of the noise in the data, which motivates our choice of cost function  $J$ . In Section 3 we present the GPRNCG algorithm. In Section 4 we present the

quasi-Newton preconditioner. Numerical results appear in Section 5.

## 2. The Test Problem

A model for pixelated data obtained from a linear, monochromatic, translation-invariant optical imaging system<sup>11</sup> is

$$d_{i,j} = \int \int s(x_i - x, y_j - y) f_{\text{true}}(x, y) dx dy + \eta_{i,j}, \quad (3)$$

for  $i = 1, \dots, n_x$ ,  $j = 1, \dots, n_y$ . Here  $f_{\text{true}}$  denotes the light source, or object. We will assume that the object is incoherent.<sup>12</sup> Then the object represents an energy density, or photon density, and hence, is nonnegative. The  $s$  in (3) is called the point spread function (PSF) and is the response of the imaging system to a point light source. With an incoherent object, the PSF is also nonnegative. The two-dimensional array  $d$  in (3) is called the (discrete, noisy, blurred) image, and  $\eta$  represents noise processes in the formation of the image.

For computational purposes, we discretize the integral in (3), e.g., using midpoint quadrature, to obtain

$$d_{i,j} = \sum_{i'} \sum_{j'} s_{i-i', j-j'} [f_{\text{true}}]_{i',j'} + \eta_{i,j}, \quad 1 \leq i \leq n_x, 1 \leq j \leq n_y. \quad (4)$$

We refer to the array with components  $s_{i,j}$  as the discrete PSF. We assume that the quadrature error is negligible and that the components  $s_{i,j}$  are nonnegative.

With lexicographical ordering of unknowns, equation (4) can be rewritten as equation (1), where  $\mathbf{d}$ ,  $\mathbf{f}_{\text{true}}$ , and  $\boldsymbol{\eta}$  are  $N \times 1$  vectors with  $N = n_x \cdot n_y$ , and  $S$  is an  $N \times N$  non-sparse matrix. We refer to  $S$  as the blurring matrix. Since  $s_{i,j} \geq 0$ ,

$$S\mathbf{f} \geq \mathbf{0} \quad \text{whenever} \quad \mathbf{f} \geq \mathbf{0}. \quad (5)$$

Because of (4),  $S$  is block Toeplitz with Toeplitz blocks (BTTB), and hence multiplication by both  $S$  and  $S^*$  can be implemented using one two dimensional fast Fourier transform (FFT) and one inverse fast Fourier transform (IFFT), each of size  $2N \times 2N$ .<sup>13</sup>

To accurately reconstruct  $\mathbf{f}_{\text{true}}$ , we apply Tikhonov regularization (i.e., stabilization with an additive penalty term<sup>13</sup>) and incorporate prior knowledge of the statistics of the noise and the nonnegativity of  $\mathbf{f}_{\text{true}}$ . This yields an optimization problem of the form of (2), where the cost functional  $J : \mathbb{R}^N \rightarrow \mathbb{R}$  has the form

$$J(\mathbf{f}) = \ell(S\mathbf{f}; \mathbf{d}) + \frac{\alpha}{2} \mathbf{f}^T \mathbf{f}. \quad (6)$$

Here  $\ell$  is a fit-to-data function, and the regularization parameter  $\alpha > 0$  quantifies the trade-off between data fidelity and stability.

#### A. *The Fit-To-Data Function*

In order to determine the form of the cost function  $J$  in (6), we need to define the fit-to-data function  $\ell$ . For computational convenience,  $\ell$  can be taken to have least squares form,

$$\ell(S\mathbf{f}, \mathbf{d}) = \frac{1}{2} \|S\mathbf{f} - \mathbf{d}\|^2. \quad (7)$$

This is appropriate when the noise in the data is independent with zero mean and common variance across pixels. (see the Gauss-Markov Theorem<sup>13</sup>) The resulting  $J$  is then quadratic.

Astronomical image data is typically collected with a device known as a CCD camera. The following statistical model (see Ref. 1) applies to image data from a

CCD detector array:

$$d_i = n_{\text{obj}}(i) + n_0(i) + g(i), \quad i = 1, \dots, N. \quad (8)$$

Here  $d_i$  is the  $i^{\text{th}}$  component of the vector  $\mathbf{d}$  and is the data acquired by a read-out of pixel  $i$  of the CCD detector array;  $n_{\text{obj}}(i)$  is the number of object dependent photoelectrons;  $n_0(i)$  is the number of background electrons; and  $g(i)$  is the readout noise. The random variables  $n_{\text{obj}}(i)$ ,  $n_0(i)$ , and  $g(i)$  are assumed to be independent of one another and of  $n_{\text{obj}}(j)$ ,  $n_0(j)$ , and  $g(j)$  for  $i \neq j$ . The random variable  $n_{\text{obj}}(i)$  has a Poisson distribution with Poisson parameter  $[\mathbf{Sf}_{\text{true}}]_i$ ;  $n_0(i)$  is a Poisson random variable with a fixed positive Poisson parameter  $\beta$ ; and  $g(i)$  is a Gaussian random variable with mean 0 and fixed variance  $\sigma^2$ .

When the statistical model for the noise is given by (8), the fit-to-data function (7) does not incorporate this prior knowledge. Unfortunately, the log-likelihood for the mixed Poisson-Gaussian model (8) has an infinite series representation (see Ref. 1) which is computationally intractable. To approximate it we observe, as in Ref. 1, that the random variable  $g(i) + \sigma^2$  has a Gaussian distribution with mean and variance both equal to  $\sigma^2$ . For large  $\sigma^2$ , this is well-approximated by a Poisson random variable with Poisson parameter  $\sigma^2$ .<sup>14</sup> Then  $d_i + \sigma^2$  is well-approximated by a Poisson random variable with Poisson parameter  $\lambda_i = [\mathbf{Sf}]_i + \beta + \sigma^2$ . From this we obtain the corresponding negative Poisson log likelihood functional

$$\ell(\mathbf{Sf}, \mathbf{d}) = \sum_{i=1}^N ([\mathbf{Sf}]_i + \beta + \sigma^2) - \sum_{i=1}^N (d_i + \sigma^2) \log([\mathbf{Sf}]_i + \beta + \sigma^2). \quad (9)$$

In Ref. 5, it is shown that when the data noise model has the form of (8), using (9) instead of (7) results in much improved reconstructions, particularly in areas of the

image in which the light intensity is low. In this paper, we will assume statistical model (8), and hence, the fit-to-data function  $\ell$  will be given by (9).

Since  $\beta$  and  $\sigma^2$  are both positive parameters and  $S\mathbf{f} \geq \mathbf{0}$  whenever  $\mathbf{f} \geq \mathbf{0}$ , the regularized Poisson likelihood functional  $J$  defined by (6) and (9) is infinitely differentiable for  $\mathbf{f} \geq \mathbf{0}$ . Its gradient is

$$\text{grad } J(\mathbf{f}) = S^T ((S\mathbf{f} + \beta - \mathbf{d}) ./ (S\mathbf{f} + \beta + \sigma^2)) + \alpha\mathbf{f}, \quad (10)$$

where  $./$  denotes component-wise division. Its Hessian is

$$\text{Hess } J(\mathbf{f}) = S^T W(\mathbf{f}) S + \alpha I, \quad (11)$$

where  $W(\mathbf{f}) = \text{diag}(\mathbf{w})$  and the components of  $\mathbf{w}$  are given by

$$w_i = \frac{d_i + \sigma^2}{([S\mathbf{f}]_i + \beta + \sigma^2)^2}. \quad (12)$$

Assuming that  $d_i + \sigma^2 \geq 0$  for each  $i$ , which will be true with high probability,<sup>5</sup> the diagonal entries of  $W(\mathbf{f})$  are bounded below by 0. And hence, the spectrum of the Hessian will be bounded below by  $\alpha$  for all  $\mathbf{f} \geq \mathbf{0}$ . Hence,  $J$  is strictly convex. This will guarantee convergence to the global minimizer of our optimization algorithm.

### 3. The Optimization Algorithm

In this section, we present the optimization algorithm that we will use for solving (2), (6), (9). We begin with preliminary notation and definitions. We define the feasible set  $\Omega$  by

$$\Omega = \{\mathbf{f} \in \mathbb{R}^N \mid \mathbf{f} \geq \mathbf{0}\}.$$

The active set for a vector  $\mathbf{f} \in \Omega$  is given by

$$\mathcal{A}(\mathbf{f}) = \{i \mid f_i = 0\}.$$

The complementary set of indices is called the inactive set and is denoted by  $\mathcal{I}(\mathbf{f})$ .

The inactive, or free, variables consist of the components  $f_i$  for which the index  $i$  is in the inactive set. Thus  $f_i > 0$  whenever  $\mathbf{f} \in \Omega$  and  $i \in \mathcal{I}(\mathbf{f})$ .

Given  $J : \mathbb{R}^N \rightarrow \mathbb{R}$ , the reduced gradient of  $J$  at  $\mathbf{f} \in \Omega$  is given by

$$[\text{grad}_R J(\mathbf{f})]_i = \begin{cases} \frac{\partial J(\mathbf{f})}{\partial f_i}, & i \in \mathcal{I}(\mathbf{f}) \\ 0, & i \in \mathcal{A}(\mathbf{f}) \end{cases}$$

and the reduced Hessian is given by

$$[\text{Hess}_R J(\mathbf{f})]_{ij} = \begin{cases} \frac{\partial^2 J(\mathbf{f})}{\partial f_i \partial f_j}, & \text{if } i \in \mathcal{I}(\mathbf{f}) \text{ or } j \in \mathcal{I}(\mathbf{f}) \\ \delta_{ij}, & \text{otherwise.} \end{cases}$$

Let  $D_{\mathcal{I}}$  denote the diagonal matrix with components

$$[D_{\mathcal{I}}(\mathbf{f})]_{ii} = \begin{cases} 1, & i \in \mathcal{I}(\mathbf{f}) \\ 0, & i \in \mathcal{A}(\mathbf{f}). \end{cases} \quad (13)$$

Then

$$\text{grad}_R J(\mathbf{f}) = D_{\mathcal{I}}(\mathbf{f}) \text{grad } J(\mathbf{f}), \quad (14)$$

$$\text{Hess}_R J(\mathbf{f}) = D_{\mathcal{I}}(\mathbf{f}) \text{Hess } J(\mathbf{f}) D_{\mathcal{I}}(\mathbf{f}) + D_{\mathcal{A}}(\mathbf{f}), \quad (15)$$

where  $D_{\mathcal{A}}(\mathbf{f}) = I - D_{\mathcal{I}}(\mathbf{f})$ .

To solve (2) (6), (9), we use the gradient projection-reduced Newton-conjugate gradient (GPRNCG) algorithm.<sup>5</sup> This algorithm combines gradient projection (GP)

iterations<sup>7,8</sup> for active set identification with conjugate gradient (CG) iterations for approximately solving the reduced Newton system

$$\text{Hess}_R J(\mathbf{f}_k^{GP})\mathbf{p} = -\text{grad}_R J(\mathbf{f}_k^{GP}). \quad (16)$$

Here  $\mathbf{f}_k^{GP}$  is the most recent gradient projection iterate.

At outer iteration  $k$ , we will denote the GP iterates by  $\{\mathbf{f}_{k,j}^{GP}\}$ . We stop the GP iterations once either

$$j = GP_{max}, \quad (17)$$

or

$$J(\mathbf{f}_{k,j-1}^{GP}) - J(\mathbf{f}_{k,j}^{GP}) \leq \gamma_{GP} \max\{J(\mathbf{f}_{k,i-1}^{GP}) - J(\mathbf{f}_{k,i}^{GP}) \mid i = 1, \dots, j-1\}, \quad (18)$$

where  $0 < \gamma_{GP} < 1$ . We define  $\mathbf{f}_k^{GP}$  to be the final GP iterate. We then perform CG iterations with CG iterates  $\{\mathbf{p}^j\}$  until either

$$j = CG_{max}, \quad (19)$$

or

$$q_k(\mathbf{p}^{j-1}) - q_k(\mathbf{p}^j) \leq \gamma_{CG} \max\{q_k(\mathbf{p}^{i-1}) - q_k(\mathbf{p}^i) \mid i = 1, \dots, j-1\}, \quad (20)$$

where  $0 < \gamma_{CG} < 1$ , and

$$q_k(\mathbf{p}) = J(\mathbf{f}_k^{GP}) + \langle \text{grad}_R J(\mathbf{f}_k^{GP}), \mathbf{p} \rangle + \frac{1}{2} \langle \text{Hess}_R J(\mathbf{f}_k^{GP}) \mathbf{p}, \mathbf{p} \rangle. \quad (21)$$

The final CG iterate is used as a search direction in a projected line search to obtain  $\mathbf{f}_{k+1}$  (see Ref. 5 for details). The GPRNCG algorithm is then given as follows:

### Gradient Projection-Reduced Newton-CG (GPRNCG) Algorithm

**Step 0:** Select initial guess  $\mathbf{f}_0$ , and set  $k = 0$ .

**Step 1:** Given  $\mathbf{f}_k$ .

(1) Take gradient projection steps until (17) or (18) is satisfied.

Return updated  $\mathbf{f}_k^{GP}$ .

**Step 2:** Given  $\mathbf{f}_k^{GP}$ .

(1) Do CG iterations to approximately minimize the quadratic

(21) until (19) or (20) is satisfied. Return  $\mathbf{p}_k$ .

(2) Using  $\mathbf{p}_k$  as the search direction, perform a projected line

search to obtain  $\mathbf{f}_{k+1}$ .

(3) Update  $k := k + 1$  and return to Step 1.

The following theorem holds for the GPRNCG algorithm applied to the problem of solving (2) for  $J$  defined by (6), (9). For a proof see Ref. 5.

**Theorem 3.1** *The iterates  $\{\mathbf{f}_k\}$  generated by GPRNCG converge to the global minimizer  $\mathbf{f}^*$  of (2), (6), (9). Furthermore, the optimal active set is identified in finitely many iterations. More precisely, there exists an integer  $m_0$  such that for all  $k \geq m_0$ ,  $\mathcal{A}(\mathbf{f}_k) = \mathcal{A}(\mathbf{f}^*)$ .*

#### 4. The Preconditioner

Because CG is used in GPRNCG to approximately solve the linear system (16), the question of preconditioning naturally arises. Unfortunately, changes in  $\mathcal{A}(\mathbf{f}_k^{GP})$  from iteration to iteration result in significant changes in the matrix  $\text{Hess}_R J(\mathbf{f}_k^{GP})$ .

This makes preconditioning very difficult. Theorem 3.1 tells us that eventually in the

GPRNCG iteration the optimal active set  $\mathcal{A}(\mathbf{f}^*)$  is identified. Once this occurs, the optimization problem becomes unconstrained, and standard preconditioning strategies can be applied. One possible strategy, then, is to precondition the CG iterations only once the optimal active set has been identified. In practice, however, identification of the optimal active set does not typically occur until late in the iteration, and, even then, one has no way of knowing precisely when this has occurred. Because of this, other strategies must be developed. In Ref. 5, a sparse preconditioner is used for this purpose. At each outer iteration  $k$  a sparse approximation to  $\text{Hess}_R J(\mathbf{f}_k^{GP})$  is built, and a sparse Choleski factorization of the matrix is used to compute the preconditioning step. Although this preconditioner is effective, it is expensive. It also depends upon the PSF, and hence, its effectiveness and the computational cost of implementation change with the application. This motivates the need for a more generally applicable and computationally efficient preconditioner.

#### A. *A Limited Memory, Quasi-Newton Preconditioner*

The idea of using limited memory, quasi-Newton matrices as preconditioners for CG iterations was introduced in Ref. 15. We extend this idea for use with active set methods in nonnegatively constrained image reconstruction. In particular, we present a preconditioning strategy that incorporates the limited memory BFGS (LBFGS) recursion into the GPRNCG algorithm. At iteration  $k$  we will denote the preconditioner by  $M_k$ .

In order to simplify our presentation, we begin with the unconstrained optimization problem, i.e. we remove the constraint  $\mathbf{f} \geq \mathbf{0}$  from (2). In this case, when

GPRNCG is implemented, (16) takes the form

$$\text{Hess } J(\mathbf{f}_k^{GP})\mathbf{p} = \text{grad } J(\mathbf{f}_k^{GP}) \quad (22)$$

for each  $k$ , and hence, the difficulties that arise due to changes in  $\mathcal{A}(\mathbf{f}_k^{GP})$  are avoided.

This allows a direct application of the preconditioning strategies found in Ref. 15.

At iteration  $k$ , previous GPRNCG iterates and their corresponding gradients can be used to build an inverse preconditioner  $M_k^{-1}$  via the BFGS recursion. More specifically, during Stage 1 of GPRNCG, we save the vectors

$$\mathbf{s}_{k,j}^{GP} := \mathbf{f}_{k,j+1}^{GP} - \mathbf{f}_{k,j}^{GP} \quad j = 0, \dots, N_k, \quad (23)$$

$$\mathbf{y}_{k,j}^{GP} := \text{grad } J(\mathbf{f}_{k,j+1}^{GP}) - \text{grad } J(\mathbf{f}_{k,j}^{GP}) \quad j = 0, \dots, N_k. \quad (24)$$

During Stage 2 we save

$$\mathbf{s}_k^{CG} := \mathbf{f}_{k+1} - \mathbf{f}_k^{GP}, \quad (25)$$

$$\mathbf{y}_k^{CG} := \text{grad } J(\mathbf{f}_{k+1}) - \text{grad } J(\mathbf{f}_k^{GP}). \quad (26)$$

In order to simplify notation, we denote the collection of these saved vectors by  $\{\mathbf{s}_l\}$  and  $\{\mathbf{y}_l\}$ , where

$$\{\mathbf{s}_l\}_{l=0}^{N_{total}} = \{\mathbf{s}_{0,0}^{GP}, \dots, \mathbf{s}_{0,N_0}^{GP}, \mathbf{s}_0^{CG}, \mathbf{s}_{1,0}^{GP}, \dots, \mathbf{s}_{k,N_k}^{GP}\}, \quad (27)$$

where  $N_{total}$  is the number of total saved vectors. The sets  $\{\mathbf{y}_l\}$  and  $\{\mathbf{f}_l\}$  are defined similarly. The BFGS recursion is then given by

$$H_{l+1} = V_l^T H_l V_l + \rho_l \mathbf{s}_l \mathbf{s}_l^T, \quad (28)$$

where

$$\rho_l = \frac{1}{\mathbf{y}_l^T \mathbf{s}_l}, \quad V_l = 1 - \rho_l \mathbf{y}_l \mathbf{s}_l^T. \quad (29)$$

The inverse preconditioner is then defined by  $M_k^{-1} = H_{N_{total}+1}$ .

Unfortunately, using this strategy to obtain  $M_k^{-1}$  requires the storage of a (typically) dense matrix at each iteration. One can circumvent this difficulty by saving instead a certain number (say  $m$ ) of the vectors  $\{\mathbf{s}_l, \mathbf{y}_l\}$  and using the recursion

$$\begin{aligned}
H_{l+1} &= (V_l^T \cdots V_{l-m+1}^T) H_{l+1}^0 (V_{l-m+1} \cdots V_l) \\
&\quad + \rho_{l-m+1} (V_l^T \cdots V_{l-m+2}^T) \mathbf{s}_{l-m+1} \mathbf{s}_{l-m+1}^T (V_{l-m+2} \cdots V_l) \\
&\quad + \rho_{l-m+2} (V_l^T \cdots V_{l-m+3}^T) \mathbf{s}_{l-m+2} \mathbf{s}_{l-m+2}^T (V_{l-m+3} \cdots V_l) \\
&\quad + \cdots \\
&\quad + \rho_l \mathbf{s}_l \mathbf{s}_l^T.
\end{aligned}$$

The inverse preconditioner is then given by  $M_k^{-1} = H_{N_{total}+1}$ . From this expression, an efficient algorithm (see Algorithm 9.1 in Ref. 9) can be derived to compute  $M_k^{-1} \mathbf{p}$ , where  $\mathbf{p} \in \mathbb{R}^N$ .

### The LBFGS Two-Loop Recursion:

**for**  $i = N_{total}, N_{total} - 1, \dots, N_{total} - m + 1$

$$\alpha_i = \rho_i \mathbf{s}_i^T \mathbf{p};$$

$$\mathbf{p} = \mathbf{p} - \alpha_i \mathbf{y}_i;$$

**end (for)**

$$\mathbf{r} = M_{k,0}^{-1} \mathbf{p};$$

**for**  $i = N_{total} - m + 1, N_{total} - m + 2, \dots, N_{total}$

$$\beta_i = \rho_i \mathbf{y}_i^T \mathbf{r};$$

$$\mathbf{r} = \mathbf{r} + (\alpha_i - \beta_i) \mathbf{s}_i;$$

**end (for)**

**stop** with result  $M_k^{-1}\mathbf{p} = \mathbf{r}$ .

Excluding the expense of computing  $M_{k,0}^{-1}\mathbf{p}$ , this scheme requires  $4mN$  multiplications. In the case where  $m \ll N$ , which is typical, this constitutes a significant savings in computational cost when compared with (28), (29). Also, the storage requirements are much less. Finally, the LBFGS recursion allows for the initial matrix  $M_{k,0}^{-1}$  to change from iteration to iteration, which is an important feature when preconditioning strategies for GPRNCG and similar algorithms are considered. In this paper, though, we will restrict our attention to the case where  $M_{k,0}^{-1} = I$ .

We see from the LBFGS algorithm above that as long as  $M_{k,0}$  is symmetric, the resulting (implicitly defined) LBFGS matrix  $M_k$  will be symmetric as well. It will also be positive definite as long as  $\langle \mathbf{s}_l, \mathbf{y}_l \rangle > 0$  for each  $l = N_{total} - m + 1, \dots, N_{total}$ .<sup>9</sup> To see that this will always be true for our applications, notice

$$\begin{aligned} \langle \mathbf{s}_l, \mathbf{y}_l \rangle &= \langle \mathbf{s}_l, \text{grad } J(\mathbf{f}_l) - \text{grad } J(\mathbf{f}_l) \rangle \\ &= \langle \mathbf{s}_l, \text{Hess } J(\mathbf{f}_l + t\mathbf{s}_l)\mathbf{s}_l \rangle \\ &> 0. \end{aligned} \tag{30}$$

The second equality follows from the mean value theorem for the function  $h(t) = \langle \mathbf{s}_l, \text{grad } J(\mathbf{f}_l + t\mathbf{s}_l) \rangle$  for some  $t \in (0, 1)$ . The last inequality follows from the fact that  $\text{Hess } J(\mathbf{f})$  is positive definite for all  $\mathbf{f} \in \Omega$  (see Section 2).

### B. Incorporating Nonnegativity Constraints

In this section, we extend the approach taken in Section A for use on problems in which nonnegativity constraints are enforced.

It is shown in Ref. 5 that if  $M_k$  is known to be an effective preconditioner for CG applied to (22), using

$$D_{\mathcal{I}}(\mathbf{f}_k^{GP})M_kD_{\mathcal{I}}(\mathbf{f}_k^{GP}) + D_{\mathcal{A}}(\mathbf{f}_k^{GP}). \quad (31)$$

as a preconditioner for CG applied to (16) can be effective. This is motivated by the definition of  $\text{Hess}_R J(\mathbf{f}_k^{GP})$  given in equation (15). Unfortunately, when the LBFGS recursion is used,  $M_k^{-1}$ , not  $M_k$ , is known. Naively, one might use

$$D_{\mathcal{I}}(\mathbf{f}_k^{GP})M_k^{-1}D_{\mathcal{I}}(\mathbf{f}_k^{GP}) + D_{\mathcal{A}}(\mathbf{f}_k^{GP}) \quad (32)$$

as an inverse preconditioner for CG applied to (16). Unfortunately, this approach has shown itself to be ineffective. Instead we advocate using LBFGS recursion as was discussed in Section A with the vectors  $\{\mathbf{s}_l, \mathbf{y}_l\}$  replaced by

$$\{\hat{\mathbf{s}}_l, \hat{\mathbf{y}}_l\}, \quad l = N_{total} - m + 1, \dots, N_{total}, \quad (33)$$

where

$$\hat{\mathbf{s}}_l = D_{\mathcal{I}}(\mathbf{f}_k^{GP})\mathbf{s}_l, \quad \hat{\mathbf{y}}_l = D_{\mathcal{I}}(\mathbf{f}_k^{GP})\mathbf{y}_l. \quad (34)$$

We denote the resulting inverse preconditioning matrix by  $\widehat{M}_k^{-1}$ .

We see from the LBFGS recursion that  $[\widehat{M}_k]_{ii} = 1$  for all  $i \in \mathcal{A}(\mathbf{f}_k^{GP})$ . (Note that  $[\text{Hess}_R J(\mathbf{f}_k^{GP})]_{ii} = 1$  for all  $i \in \mathcal{A}(\mathbf{f}_k^{GP})$ .) We also see that  $\widehat{M}_k^{-1}$  is symmetric. In order to guarantee that it is positive definite, the curvature condition

$$\langle \hat{\mathbf{s}}_l, \hat{\mathbf{y}}_l \rangle > 0 \quad (35)$$

must be satisfied for each of the vectors  $\hat{\mathbf{s}}_l$  and  $\hat{\mathbf{y}}_l$  in (33). We note that for some  $t \in (0, 1)$ ,

$$\langle \hat{\mathbf{s}}_l, \hat{\mathbf{y}}_l \rangle = \langle D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l, D_{\mathcal{I}}(\mathbf{f}_{l+1})\text{Hess } J(\mathbf{f}_{l+1} + t\mathbf{s}_l)\mathbf{s}_l \rangle, \quad (36)$$

$$\begin{aligned} &= \langle D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l, D_{\mathcal{I}}(\mathbf{f}_{l+1})\text{Hess } J(\mathbf{f}_{l+1} + t\mathbf{s}_l)D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l \rangle \\ &\quad + \langle D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l, D_{\mathcal{I}}(\mathbf{f}_{l+1})\text{Hess } J(\mathbf{f}_{l+1} + \tau\mathbf{s}_l)D_{\mathcal{A}}(\mathbf{f}_{l+1})\mathbf{s}_l \rangle. \end{aligned} \quad (37)$$

Once the active set is identified, i.e. for all  $k \geq m_0 + m$ , where  $m_0$  is given in Theorem 3.1 and  $m$  is the number of saved vectors in the LBFGS recursion,  $D_{\mathcal{A}}(\mathbf{f}_{l+1})\mathbf{s}_l = \mathbf{0}$ . In addition, by an argument analogous to that found in (30),

$$\langle D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l, D_{\mathcal{I}}(\mathbf{f}_{l+1})\text{Hess } J(\mathbf{f}_{l+1} + t\mathbf{s}_l)D_{\mathcal{I}}(\mathbf{f}_{l+1})\mathbf{s}_l \rangle > 0.$$

Hence, for  $k \geq m_0 + m$ ,  $\langle \hat{\mathbf{s}}_l, \hat{\mathbf{y}}_l \rangle > 0$  for  $l = N_{total} - m + 1, \dots, N_{total}$  and  $\widehat{M}_k$  is positive definite. We summarize these results in the following theorem.

**Theorem 4.1** *Let  $\{\mathbf{f}_k\}$  be the iterates generated by GPRNCG applied to problem (2), (6), (9). Then, given a value for the limited memory parameter  $m$  in the LBFGS recursion, there exists a positive integer  $K_0$  such that for all  $k \geq K_0$ , the quasi-Newton matrix  $\widehat{M}_k$  defined above is symmetric and positive definite.*

As was discussed above, the active set is often not identified until late in the GPRNCG iteration, and hence, the result of Theorem 4.1 is of little practical use. On the other hand, in our experience, curvature condition (35) is satisfied for a large percentage of the vector pairs in (33) well before the active set has been identified, and this percentage increases as changes in the active set from iteration to iteration lessen.

In practice, however, failure of condition (35) does occur, in which case we advocate not using the corresponding pair in the LBFGS recursion. Then the preconditioner  $\widehat{M}_k$  will always be positive definite.

**Remark:** The preconditioning strategy presented in this section can also be used to speed up CG iterations in the GPCG algorithm. Recall that GPCG is an effective algorithm for solving (2), (6), (7). This strategy can also be modified in a straightforward fashion for use with other active set methods that employ CG iterations, e.g., if CG iterations are used for approximate computation of the Newton step in the projected Newton algorithm of Bertsekas.<sup>16</sup>

## 5. Numerical Results

Our primary goal in this section is to demonstrate that the preconditioning strategy presented in the previous section improves the performance of the GPRNCG algorithm. Also, in order to demonstrate the effectiveness of the overall algorithm, we include in our comparison the bound constrained variant of the limited memory BFGS algorithm<sup>9</sup> known as LBFGS-B.<sup>17,18</sup> LBFGS-B was implemented in MATLAB via a mex interface with FORTRAN77 source code made publicly available by the authors of Refs. 17, 18.

In Figure 6, simulated true images are plotted. Data are generated using statistical model (8), where the Poisson parameter for  $n_0(i)$  is zero for all  $i$ , and the mean and variance of the Gaussian random variable  $g(i)$  are 0 and 9 respectively. The error level is then adjusted by changing the intensity of the true image. In one set of simulations,

we generate data with an error level of one percent. That is,

$$\frac{\|\boldsymbol{\eta}\|}{\|S\mathbf{f}_{\text{true}}\|} = .01,$$

where  $\boldsymbol{\eta}$ ,  $S$ , and  $\mathbf{f}_{\text{true}}$  are from equation (1). This data is plotted in Figure 6. We also perform our comparisons with data generated at an error level of ten percent.

Given a particular data set, we define  $\mathbf{f}_\alpha$  to be the global constrained minimizer of problem (2), (6), (9). The regularization parameter  $\alpha$  is then chosen to approximately minimize  $\|\mathbf{f}_\alpha - \mathbf{f}_{\text{true}}\|$ . For the binary star data,  $\alpha = 4 \times 10^{-12}$  for one percent noise, and  $\alpha = 2 \times 10^{-9}$  for ten percent noise. For the satellite data,  $\alpha = 1 \times 10^{-9}$  for one percent noise, and  $\alpha = 2 \times 10^{-6}$  for ten percent noise. Plots of the reconstructed images are given in Figures 6.

In Figures 6 and 6, we compare the performance of GPRNCG, preconditioned GPRNCG, and LBFBS-B. We save five BFGS vectors for the preconditioning matrix in preconditioned GPRNCG, and we save ten BFGS vectors for LBFBS-B. These choices balance computational cost with rapid convergence. We set  $CG_{max} = 10$  and  $GP_{max} = 1$  in both GPRNCG and preconditioned GPRNCG for each data set, with the exception of the binary star with one percent noise, in which  $CG_{max} = 20$ . In order to compare the performance of the algorithms, we plot the relative solution error, which we define by

$$\frac{\|\mathbf{f}_k - \mathbf{f}_\alpha\|}{\|\mathbf{f}_\alpha\|}$$

on the vertical axis, versus total fast fourier transforms (FFTs) on the horizontal axis. The cost of computing FFTs dominates the cost of the implementation of the algorithms under comparison.

With the exception of the solution error plots for the satellite data with ten percent noise, in which GPRNCG without preconditioning minimizes the solution error with the least amount of cost, preconditioned GPRNCG is the superior algorithm. In our experience, as the difficulty of the problem increases, preconditioned GPRNCG is more effective. One possible explanation for this is that for the more difficult problems, the active set at the solution is identified, or nearly so, long before the algorithm reaches convergence, and it is in this scenario that our preconditioning strategy works best. This is motivated in part by Theorem 4.1. For the problem with satellite data with ten percent noise, each of the algorithms converges rapidly, and the active set at the solution is identified within a couple of iterations of convergence. Hence, the preconditioning strategies are not given a chance to work. In such cases, though, it could be argued that preconditioning is not necessary.

Comments in the previous paragraph suggest a strategy for further improving the algorithm. One can begin preconditioning only after the active set has stabilized. In Bardsley, et al. such a strategy is implemented for a convex minimization problem very similar to that of this paper.<sup>5</sup> Here, a sparse preconditioner is used that is very expensive in early iterations, and a decrease in CPU time of approximately 20% is obtained. Although the preconditioner presented in this paper is not expensive to implement, it is inefficient in early iterations when the active set is changing dramatically. This is supported by the convergence plots in Figures 6 and 6, where GPRNCG without preconditioning performs better than preconditioned GPRNCG in early iterations. Consequently, waiting to implement the preconditioner until the active set has stabilized is likely to yield a minor improvement in computational efficiency, but

we do not pursue such a strategy in this paper.

## 6. Conclusions

We present a limited memory, quasi-Newton preconditioner for use with the GPRNCG algorithm applied to nonnegatively constrained image reconstruction. The preconditioner is defined implicitly via the LBFGS recursion, and is sufficiently general so that it can be implemented in conjunction with other constrained algorithms.

We compare the performance of GPRNCG both with and without preconditioning. In all but one case, the convergence properties of the GPRNCG algorithm improve when the preconditioner is used. In the case where preconditioning does not improve the convergence properties of GPRNCG, convergence of the GPRNCG algorithm was rapid, and hence, it can be argued that preconditioning was not necessary. In addition, we compare the performance of these algorithms with that of the LBFGS-B algorithm and find, in all but one case, that GPRNCG with preconditioning is the most efficient of the three. In the other case, GPRNCG without preconditioning was most effective.

Please send all correspondence to `bardsleyj@mso.umt.edu`.

## References

1. D. L. Snyder, A. M. Hammoud, and R. L. White, “Image recovery from data acquired with a charge-coupled-device camera”, *J. Opt. Soc. Am. A* **10**, 1014-1023 (1993).

2. L. B. Lucy, “An Iterative Technique for the Rectification of Observed Distributions”, *The Astronomical Journal* **79**, 745-754 (1974).
3. W. H. Richardson, “Bayesian-Based Iterative Method of Image Restoration”, *J. Opt. Soc. Am.* **62**, 55-59 (1972).
4. J. Nagy and Z. Strakos, “Enforcing nonnegativity in image reconstruction algorithms”, in *Mathematical Modeling, Estimation, and Imaging*, David C. Wilson, et.al., eds., *Proc. SPIE* **4121**, 182-190 (2000).
5. J. M. Bardsley and C. R. Vogel, “A Nonnegatively Constrained Convex Programming Method for Image Reconstruction”, *SIAM J. Scientific Comput.* (to be published).
6. J. J. Moré and G. Toraldo, “On the Solution of Large Quadratic Programming Problems with Bound Constraints”, *SIAM J. Optimization* **1**, 93-113 (1991).
7. D. P. Bertsekas, “On the Goldstein-Levitin-Poljak Gradient Projection Method”, *IEEE Transactions on Automatic Control* **21**, 174–184 (1976).
8. C. T. Kelley, “Iterative Methods for Optimization”, SIAM, Philadelphia, 1999.
9. J. Nocedal and S. J. Wright, “Numerical Optimization”, Springer-Verlag, New York, 1999.
10. Jesse L. Barlow and Geraldo Toraldo, “The Effect of Diagonal Scaling on Projected Gradient Methods for Bound Constrained Quadratic Programming Problems”, *Optimization Methods and Software* **5**, 235-245 (1995).
11. J. W. Goodman, “Introduction to Fourier Optics, 2nd Edition”, McGraw-Hill, New York 1996.

12. J. W. Goodman, "Statistical Optics", John Wiley and Sons, New York, 1985.
13. C. R. Vogel, "Computational Methods for Inverse Problems", SIAM, Philadelphia, 2002.
14. W. Feller, "An Introduction to Probability Theory and Its Applications", Wiley, New York, 1971.
15. J. Nocedal and J. L. Morales, "Automatic Preconditioning by Limited Memory Quasi-Newton Updating", *SIAM J. Optimization* **10**, 1079-1096 (2000).
16. D. P. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints", *SIAM J. Control and Optimization*, **20**, 221-246 (1982).
17. R. H. Byrd, P. Lu and J. Nocedal, "A Limited Memory Algorithm for Bound Constrained Optimization", *SIAM J. Scientif. Comput.* **16**, 1190-1208 (1995).
18. C. Zhu, R. H. Byrd and J. Nocedal, "L-BFGS-B: FORTRAN subroutines for large scale bound constrained optimization", *ACM Transactions on Mathematical Software* **23**, 550-560 (1997).

**Caption 1.**

Mesh plots of the upper left  $64 \times 64$  pixels of the binary star and satellite true images.

**Caption 2.**

Mesh plots of the upper left  $64 \times 64$  pixels of the binary star and satellite data with 1% noise.

**Caption 3.**

Reconstructions. On the top is a mesh plot of the upper left  $64 \times 64$  pixels of the reconstruction corresponding to the binary star data with one percent noise. On the bottom is a mesh plot of the upper left  $64 \times 64$  pixels of the reconstruction corresponding to the satellite data with one percent noise.

**Caption 4.**

Relative Solution Error versus FFTs for Binary Star Data. The horizontal axis denotes cumulative FFTs. The vertical axis shows the relative solution error on a logarithmic scale. The plot on the top shows convergence results for one percent noise. The plot on the bottom shows convergence results for 10 percent noise. The solid line denotes GPRNCG with preconditioning. The dash-dot line denotes GPRNCG without preconditioning. The dashed line denotes LBFGS-B.

**Caption 5.**

Relative Solution Error versus FFTs for Satellite Data. The horizontal axis denotes cumulative FFTs. The vertical axis shows the relative solution error on a logarithmic scale. The plot on the top shows convergence results for one percent noise. The plot on the bottom shows convergence results for 10 percent noise. The solid line denotes GPRNCG with preconditioning. The dash-dot line denotes GPRNCG without

preconditioning. The dashed line denotes LBFGS-B.

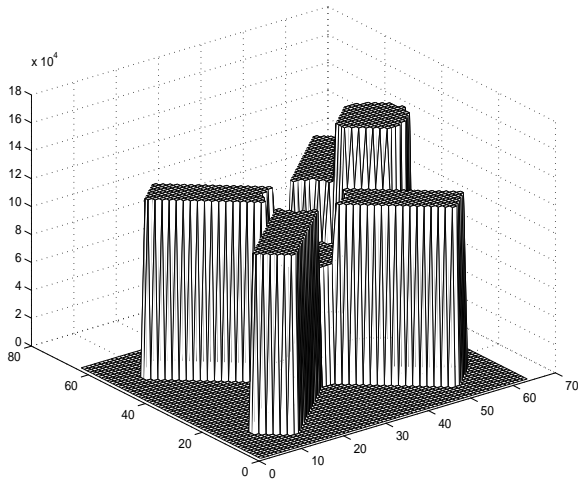
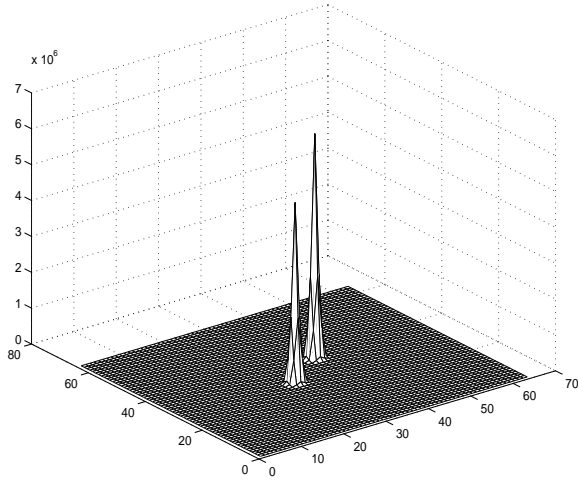


Figure 1, A9086

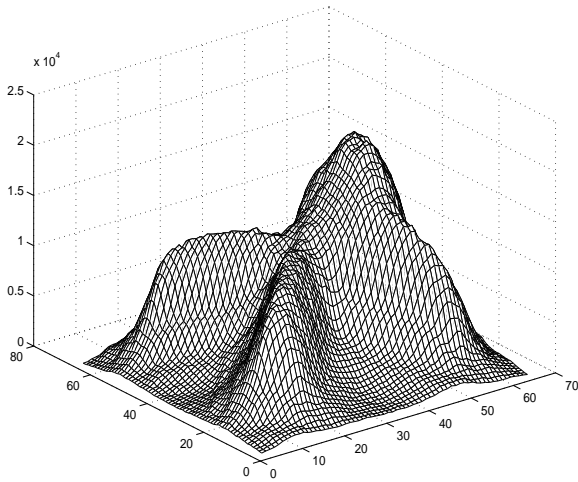
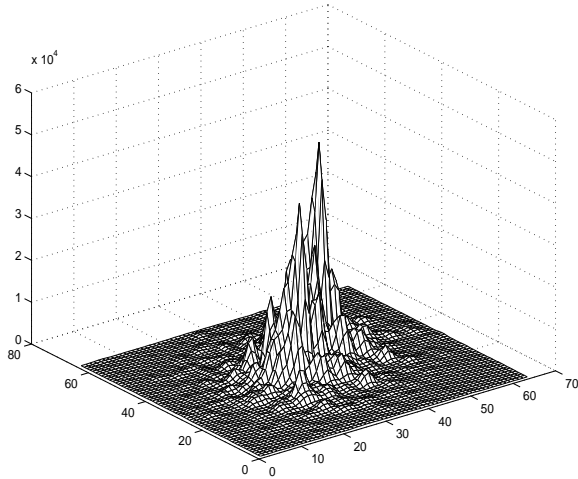


Figure 2, A9086

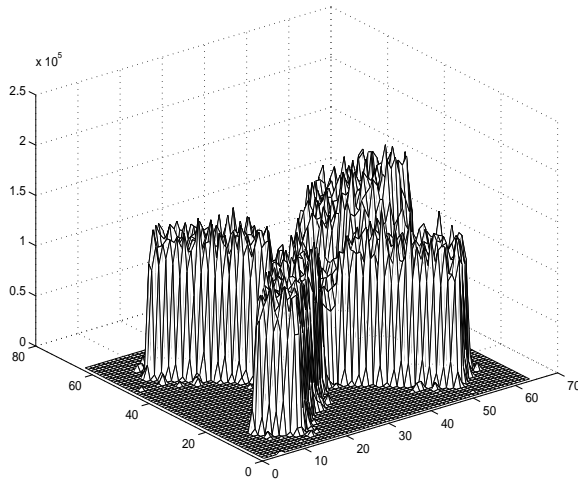
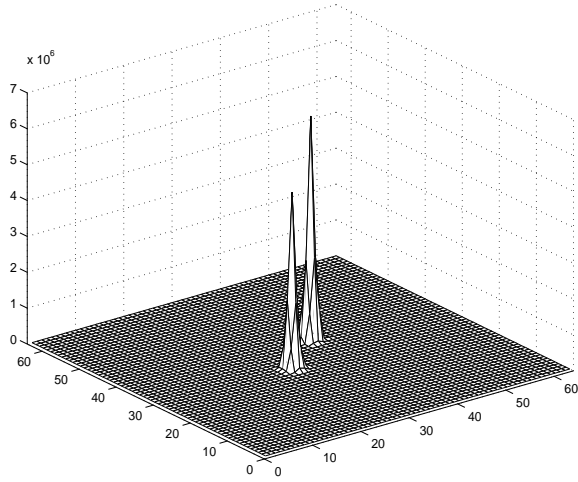


Figure 3, A9086

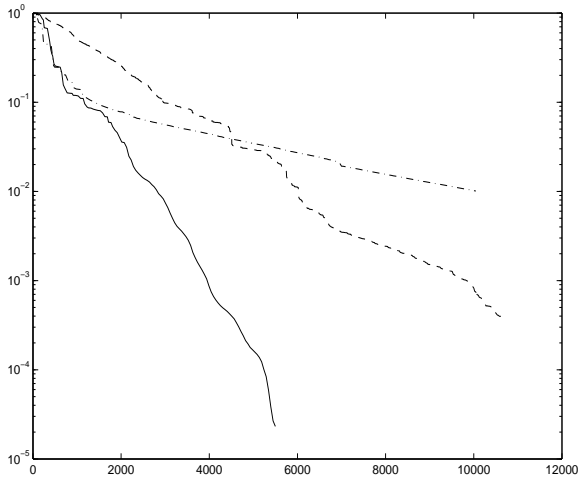
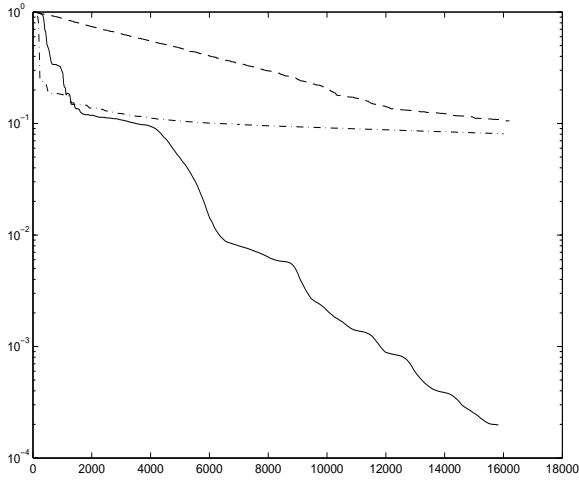


Figure 4, A9086

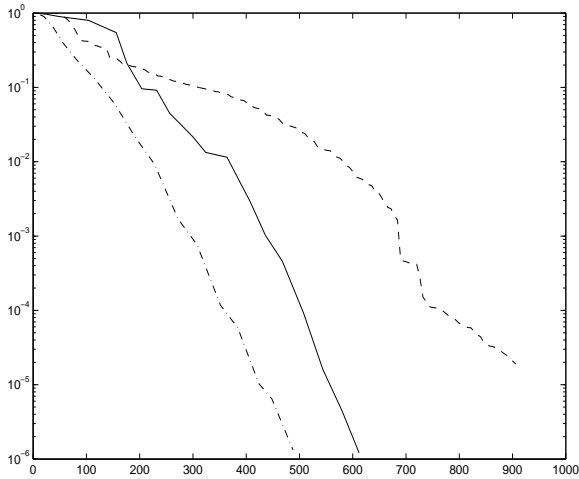
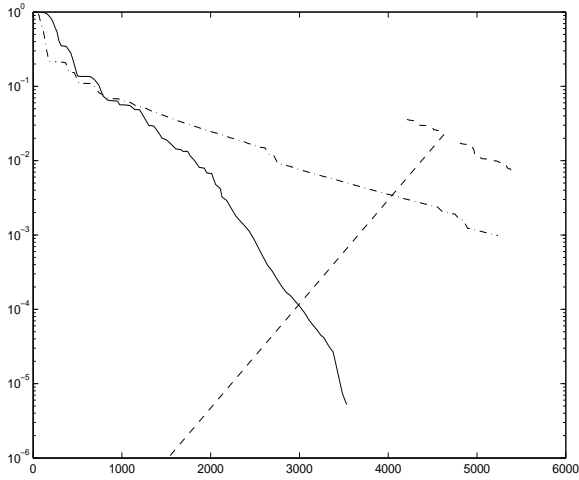


Figure 5, A9086