BIT
2000, Vol. 40, No. 4, pp. 001–004
0006-3835/00/4004-0001 $15.00
© Swets & Zeitlinger

STEEPEST DESCENT, CG, AND ITERATIVE REGULARIZATION OF ILL-POSED PROBLEMS *

K. P. LEE[1] and J. G. NAGY[2] †

[1] *Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, USA. email: kpalmer@mathcs.emory.edu*

[2] *Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, USA. email: nagy@mathcs.emory.edu*

Abstract.

The state of the art iterative method for solving large linear systems is the conjugate gradient (CG) algorithm. Theoretical convergence analysis suggests that CG converges more rapidly than steepest descent. This paper argues that steepest descent may be preferable to CG when solving linear systems arising from the discretization of ill-posed problems. Specifically, it is shown that, for ill-posed problems, steepest descent has a more stable convergence behavior than CG, which may be explained by the fact that the so called "filter factors" for steepest descent behave much less erratically than those for CG. Moreover, it is shown that, with proper preconditioning, the convergence rate of steepest descent is competitive with that of CG.

AMS subject classification: 65F20, 65F30.

Key words: ill-posed problem, regularization, conjugate gradient, steepest descent.

1 Introduction

We consider linear systems that arise from discretization of ill-posed problems,

$$\mathbf{b} = A\mathbf{x} + \mathbf{n}\,, \tag{1.1}$$

where A is an ill-conditioned $n \times n$ matrix, whose singular values decay to zero. The vector $\mathbf{b}$ and matrix A are assumed known, and the vector $\mathbf{n}$, which represents perturbations and/or noise in the data, is assumed unknown. The aim is to compute an approximation of the unknown vector, $\mathbf{x}$. Following the terminology of Hansen [7], we call such a linear system a *discrete ill-posed problem*.

Discrete ill-posed problems arise in applications, such as astronomy, medical imaging, geophysical applications, and many other areas [4, 7, 11]. Because A is ill-conditioned, and because there is noise in $\mathbf{b}$, it is very difficult to compute accurate approximations of $\mathbf{x}$. To see why this is the case, suppose we naively assume the noise is small enough to ignore, and compute $\mathbf{x}_{\text{naive}} = A^{-1}\mathbf{b}$. Let $A = U\Sigma V^T$ be the singular value decomposition (SVD) of A, where U and V

*Received October 2002. Revised ??? Communicated by ???

†This work was partially supported by the National Science Foundation under Grant DMS 00-75239.

are $n \times n$ orthogonal matrices, and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. If we denote the columns of U and V as $\mathbf{u}_i$ and $\mathbf{v}_i$, respectively, then

$$\mathbf{x}_{\text{naive}} = V\Sigma^{-1}U^T\mathbf{b} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{n} \left(\xi_i + \frac{\eta_i}{\sigma_i}\right)\mathbf{v}_i\,, \tag{1.2}$$

where $\mathbf{x} = \sum_i \xi_i \mathbf{v}_i$ and $\mathbf{n} = \sum_i \eta_i \mathbf{u}_i$. From (1.2), we can make the following observations. First, discrete ill-posed problems have the property that the singular values decay to, and cluster at 0. Thus, division by small singular values, σ_i, magnifies the corresponding noise components, η_i. In addition, most discrete ill-posed problems have the additional property that the singular vectors, $\mathbf{v}_i$, tend to oscillate more for smaller singular values. Thus, the naive solution is horribly corrupted by large oscillating components. These properties of ill-posed problems, as well as the above explanation using the SVD, are well known; see, for example, [4, 7, 11] for more details.

A general approach to compute an accurate approximation of $\mathbf{x}$ can be formulated as a modification of the naive solution [7]; specifically,

$$\mathbf{x}_{\text{filt}} = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i\,, \tag{1.3}$$

where the *filter factors*, ϕ_i, satisfy $\phi_i \approx 1$ for large σ_i, and $\phi_i \approx 0$ for small σ_i. That is, the large singular value components of the solution are reconstructed, while the components corresponding to the small singular values are filtered out.

Different choices of filter factors lead to different methods. For example, the truncated SVD (TSVD) solution is given by (1.3), where

$$\phi_i = \begin{cases} 1 & \text{if } i \leq k \\ 0 & \text{if } i > k \end{cases}\,.$$

If $\Sigma^\dagger = \text{diag}(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}, 0, \ldots, 0)$, then the TSVD solution can be written as

$$\mathbf{x}_{\text{tsvd}} = U\Sigma^\dagger V^T\,. \tag{1.4}$$

The spectral filter solution (1.3) is also referred to as a *regularization* method. The truncation index, k, is called a *regularization parameter*, whose specific value is problem dependent. It is nontrivial to choose an "optimal" regularization parameter, though good values can be estimated using methods such as generalized cross validation, the L-curve, or some other scheme [4, 7, 11].

There are many other regularization methods besides TSVD. In this paper we focus on iterative regularization methods, which may be appropriate for large scale problems. We describe the basic idea behind iterative regularization in Section 2, and, in particular, discuss the conjugate gradient (CG) and steepest descent methods. We compare the filter factors for these methods, and investigate how their behavior is related to the convergence history of the algorithms. In Section 3 we see how this behavior carries over to preconditioned versions

of the algorithms. We find that, with proper preconditioning, the convergence rate of steepest descent is competitive with that of CG, and, moreover, that steepest descent has a more stable convergence behavior than CG. In Section 4 we compare the two preconditioned algorithms on a realistic large scale example from image restoration. Concluding remarks are given in Section 5.

2 Iterative Regularization

Iterative methods are used to solve large linear systems, $A\mathbf{x} = \mathbf{b}$. Ultimately, the iteration converges to $\mathbf{x}_{\text{naive}} = A^{-1}\mathbf{b}$, which, for ill-posed problems, is a poor approximation of the true solution, $\mathbf{x}$. Fortunately, schemes such as CG and steepest descent belong to a class of *iterative regularization* methods [4, 11]. Such iterative algorithms exhibit a *semi-convergence* behavior with respect to the relative error, $||\mathbf{x}_k - \mathbf{x}||/||\mathbf{x}||$, where $\mathbf{x}_k$ is the approximate solution at the kth iteration. That is, the relative error begins to decrease and, after some "optimal" iteration, begins to rise. By stopping the iteration when the error is low, we obtain a good (regularized) approximation of the solution. The iteration index plays the role of the regularization parameter. As with other regularization methods, it is a nontrivial matter to choose an "optimal" stopping iteration.

Because it converges more rapidly, CG is usually considered superior to steepest descent. For ill-posed problems, though, one should not only consider convergence speed, but also regularization properties of the methods. In this section we compare the filter factors and convergence behavior of CG and steepest descent. Because our problems are usually not symmetric, we use implementations applied (implicitly) to the normal equations, $A^TA\mathbf{x} = A^T\mathbf{b}$. In particular, we consider the conjugate gradient method for least squares problems (CGLS) [1], and the residual norm steepest descent method (RNSD) [9].

To determine the filter factors for RNSD, we first need to recall that, given an initial guess, $\mathbf{x}_0$, the RNSD iteration is given by $\mathbf{x}_{k+1} = x_k + \alpha_k A^T\mathbf{r}_k$, where $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$. Substituting $\mathbf{r}_k$ into $\mathbf{x}_{k+1}$, we have

$$\mathbf{x}_{k+1} = \alpha A^T\mathbf{b} + (I - \alpha A^TA)\mathbf{x}_k\,. \tag{2.1}$$

Assuming, without loss of generality, $\mathbf{x}_0 = \mathbf{0}$, and using (2.1), we see that

$$\mathbf{x}_{k+1} = P_k(A^TA)A^T\mathbf{b}\,, \tag{2.2}$$

where the polynomial P_k is defined as

$$P_k(\lambda) = P_{k-1}(\lambda) + \alpha_k(1 - A^TAP_{k-1}(\lambda))\,, \quad P_{-1}(\lambda) = 0\,. \tag{2.3}$$

Substituting $A = U\Sigma V^T$ into (2.2), we obtain

$$\mathbf{x}_{k+1} = \sum_{i=1}^{n} \sigma_i^2 P_k(\sigma_i^2)\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i \tag{2.4}$$

Comparing (2.4) with (1.3), we see that the RNSD filter factors are $\sigma_i^2 P_k(\sigma_i^2)$.

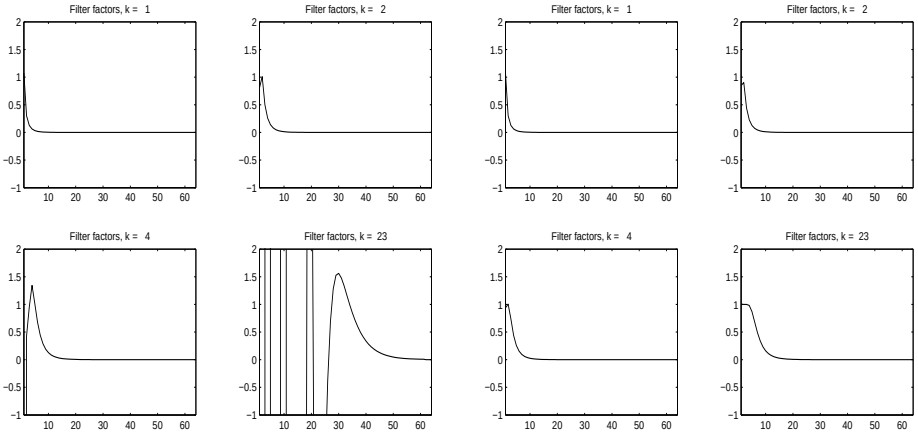
Filter factors for CGLS can be found in a similar way; see [10, 7]. Specifically, they are $\sigma_i^2 R_k(\sigma_i^2)$ where the polynomial R_k is defined as

$$R_k(\lambda) = \left(1 - \alpha_k\lambda + \frac{\alpha_k\beta_{k-1}}{\alpha_{k-1}}\right) R_{k-1}(\lambda) - \frac{\alpha_k\beta_{k-1}}{\alpha_{k-1}} R_{k-2} + \alpha_k\,, \tag{2.5}$$

where $R_{-1}(\lambda) = 0$ and $R_0(\lambda) = \alpha_0$.

Because $P_k(\lambda)$ and $R_k(\lambda)$ are defined recursively, it is difficult to see how the filter factors for RNSD and CGLS compare. We can, however, make such a comparison experimentally for small scale problems. Consider, for example, the test problem `heat` from Hansen's *Regularization Tools* [6]. We used this example to construct $\mathbf{b} = A\mathbf{x}+\mathbf{n}$, where A is a 64×64 matrix and $\mathbf{n}$ is a vector containing random entries, normally distributed with mean 0, variance 1, and scaled so that $\dfrac{||\mathbf{n}||}{||A\mathbf{x}||} = 0.001$. The condition number of A is approximately 7.03×10^{28}.

In Figure 2.1, we show the filter factors for iterations $k = 1, 2, 4$ and 23. Recall that we would like the filter factors to start off at 1, and eventually decay to 0. We observe that at the 4th iteration, the filter factors for CGLS begin to oscillate, and this oscillation increases as the iterations proceed (as can be seen when $k = 23$). This property of the CGLS filter factors was observed by Vogel [10]. In comparison, the filter factors for RNSD behave much more stably.



(a) CGLS filter factors. (b) RNSD filter factors.

Figure 2.1: Filter factors for a few selected iterations of CGLS and RNSD.

The iterative regularization properties of CGLS and RNSD are seen in Figure 2.2. Note that CGLS converges much more rapidly than RNSD, and the semi-convergence behavior of CGLS is much more pronounced than it is for RNSD. This suggests that if we do not have a good stopping criteria for CGLS, then we are at high risk of computing a poor solution. However, RNSD converges too slowly to be considered a practical method for this problem.

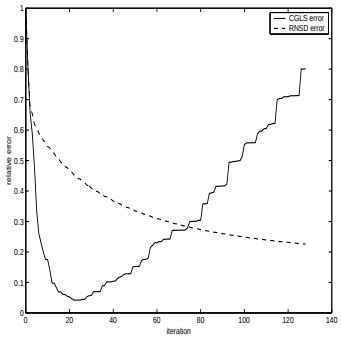


Figure 2.2: Convergence history of CGLS and RNSD.

We have done similar experiments with several other examples, including those from [6], but omit the results due to space restrictions. We remark, however, that we obtained similar results for each problem.

Though one would be tempted to dismiss RNSD as a viable approach, and instead search for good stopping criteria for CGLS, we compare the methods further when preconditioning is used to accelerate convergence.

3 Preconditioning

Preconditioning is used with iterative methods to accelerate the rate of convergence; that is, to reduce the number of iterations needed to compute a good approximation of the solution. The standard approach to preconditioning, when solving $A\mathbf{x} = \mathbf{b}$, is to construct a matrix, P, such that the singular values of $P^{-1}A$ are clustered around 1. More singular values clustered around one, as well as tighter clusters, implies faster convergence.

At each iteration of a preconditioned method, a linear system of the form $P\mathbf{z} = \mathbf{w}$ is solved. If P is a good approximation to a severely ill-conditioned A, then P will also be very ill-conditioned, and thus inaccuracies in the data will be highly amplified when $P\mathbf{z} = \mathbf{w}$ is solved in the first iteration. Further iterations do not improve the situation, and, in general, there is no hope of recovering a good approximation of the solution. Thus, the standard approach to preconditioning cannot be used when solving ill-posed problems.

An approach for preconditioning ill-posed problems, first proposed in [5], is to construct a matrix P that clusters the large singular values around one, but leaves the small singular values alone. We can argue why this approach can be effective by constructing an "ideal" preconditioner as follows. Let $A = U\Sigma V^T$ be the SVD of A, where $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$. Define P_k as

$$P_k = U\Sigma_k V^T, \tag{3.1}$$

where $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k, 1, \ldots, 1)$, and k is the truncation index for the TSVD solution (1.4). Then the preconditioned system has the form $P^{-1}A = V\Delta V^T$,

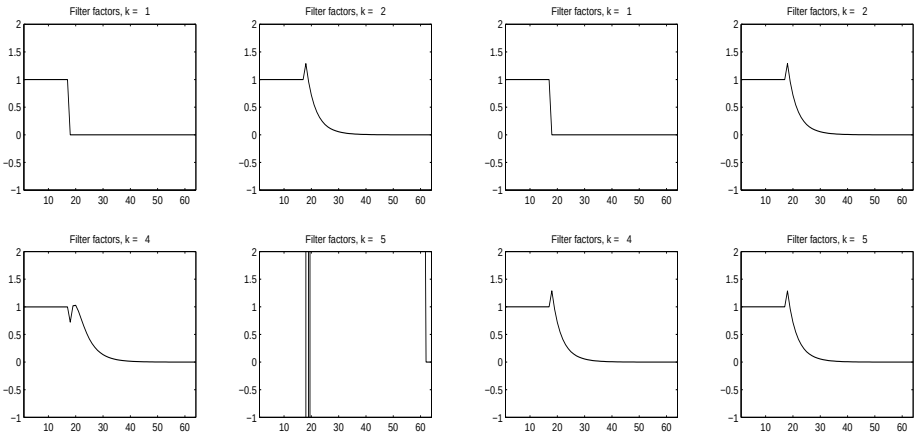
where $\Delta = \text{diag}(1, \ldots, 1, \sigma_{k+1}, \ldots, \sigma_n)$. Thus the large singular values are perfectly clustered at one, and are well separated from the small singular values. Regularization properties of the iterative methods imply that it takes only one iteration to compute a regularized solution.

To illustrate the effectiveness of this "ideal" preconditioner, and to study the behavior of the filter factors for the preconditioned algorithms PCGLS and PRNSD, we again consider the `heat` example. The preconditioner is given by (3.1), where the truncation index, k, was chosen to

$$\min_k ||V\Sigma_k^\dagger U^T \mathbf{b} - \mathbf{x}||, \tag{3.2}$$

where $\mathbf{x}$ is the true solution. We admit that it is unrealistic to construct such an ideal preconditioner in practice, but the purpose of this section is to investigate the behavior of PCGLS and PRNSD under ideal situations. We provide a more realistic example in the next section.

The filter factors for PCGLS and PRNSD at iterations $k = 1, 2, 4$, and 5 are shown in Figure 3.1. Notice that the filter factors for the early iterations of both methods behave like they should for a spectral filtering scheme, starting at 1, and eventually decaying to 0. But at the 5th iteration, the PCGLS filter factors begin to oscillate, and this oscillation increases as the iteration proceeds. However, the filter factors for PRNSD remain virtually unchanged for iterations 2, 4, and 5, and this situation continues for subsequent iterations.

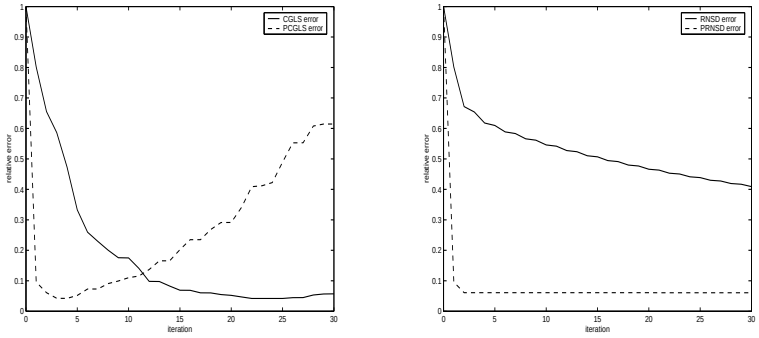


(a) PCGLS filter factors. (b) PRNSD filter factors.

Figure 3.1: Filter factors for a few selected iterations of PCGLS and PRNSD.

In Figure 3.2, we show the convergence history of PCGLS and PRNSD, and observe that the convergence rate of PRNSD is competitive with that of PCGLS. Moreover, the PRNSD relative error does not increase drastically like it does for PCGLS. This is an important point because one of the most difficult issues for iterative regularization methods is determining an optimal stopping point. With

PCGLS, there is a much higher risk of computing a poor solution when using an imperfect stopping rule. As with the unpreconditioned case, we obtained similar results on several other examples.



(a) CGLS/PCGLS convergence. (b) RNSD/PRNSD convergence.

Figure 3.2: Convergence history of PCGLS and PRNSD.

The observations in this section have been made under ideal conditions. For a realistic, large scale problem, it is not possible to compute an SVD to use as an ideal preconditioner. But, as illustrated next, in some applications, it is possible to use an approximate SVD, or spectral decomposition.

4 An Application to Image Restoration

There are many applications where scientists use images to analyze experiments, study structures, and record unique events. It may be difficult, or impossible, to repeat the process by which the image was obtained. Unfortunately, due to environmental effects and/or technological problems, the images are rarely perfect, and in some cases may be substantially distorted by blurring and noise. Image restoration is the process of recovering a good image from the degraded one. In many cases the image restoration problem can be accurately modeled as a discrete ill-posed problem (1.1), where $\mathbf{b}$ represents the observed, blurred and noisy, image, $\mathbf{x}$ represents an ideal, undistorted image, and A models the blurring phenomenon.

The matrices that arise in image restoration are highly structured, involving circulant, Toeplitz and Hankel matrices. For an $n \times n$ image, A is $n^2 \times n^2$. Matrix vector multiplications with A can be done using $O(n^2 \log n)$ arithmetic operations using fast Fourier transforms (FFT). Preconditioning such structured matrices has been thoroughly investigated in the literature; see the survey paper [2] and the references therein. Moreover, many of these approaches have been applied to image restoration [5].

Although a variety of approaches to preconditioning have been proposed, the most popular is to use circulant approximations. The approximation we use is

$$\min ||A - P||_F$$

where, for two dimensional image restoration problems, the minimization is done over all matrices that are block circulant with circulant blocks (BCCB). It is very inexpensive to construct these preconditioners; see [2] for more details.

An important and useful property of BCCB matrices is that they can be diagonalized efficiently using FFTs. That is, every BCCB matrix can be written as $P = \mathcal{F}^* \Lambda \mathcal{F}$, where Λ is a diagonal matrix containing the eigenvalues of P, $\mathcal{F}$ is the unitary (two dimensional) discrete Fourier transform matrix, and $\mathcal{F}^*$ is the complex conjugate transpose of $\mathcal{F}$ [3]. Thus, a BCCB approximation of A provides an approximate spectral decomposition, which can be used in an analogous manner as outlined in Section 3 for the ideal SVD preconditioner. That is, we determine a reasonable truncation index, and construct the preconditioner

$$P = \mathcal{F}^* \Lambda_k \mathcal{F} .$$

In a realistic problem, we cannot use (3.2) to determine the truncation index. Instead, we must use a *parameter-choice* method, such as the discrepancy principal, generalized cross validation (GCV), L-curve, or some other method [4, 7, 11]. We use GCV for the results reported in this section.

We remark that for computational purposes, the matrix $\mathcal{F}$ does not need to be constructed explicitly, since multiplying a vector by $\mathcal{F}$ is equivalent to applying an FFT to that vector, and multiplication by $\mathcal{F}^*$ is equivalent to applying an inverse FFT. Moreover, the eigenvalues of the BCCB matrix can be obtained by computing an FFT of its first column [3]. For an $n \times n$ image (thus an $n^2 \times n^2$ matrix A), it requires only $O(n^2 \log n)$ arithmetic operations to compute an FFT.

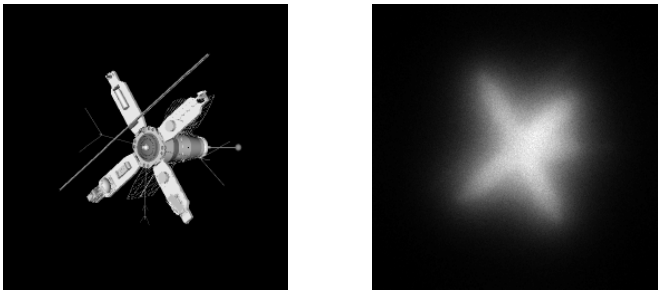
It is not the intention of this section to propose a new preconditioner, but instead to see how PCGLS and PRNSD perform on a realistic problem, using a well established approach for preconditioning. For large scale problems, we cannot compute the filter factors, since it requires knowledge of the singular values of the preconditioned system, $P^{-1}A$. However, if the true solution is known, we can monitor and compare the convergence history of the methods.

The image restoration test problem[1] we use was developed at the US Air Force Phillips Laboratory, Lasers and Imaging Directorate, Kirtland Air Force Base, New Mexico. The image is a computer simulation of a field experiment showing a satellite as taken from a ground based telescope. The true and blurred images have 256×256 pixels, and are shown in Figure 4.1. The matrix A is an ill-conditioned $65,536 \times 65,536$ block Toeplitz matrix with Toeplitz blocks.

This data has been widely used in the literature for testing image restoration algorithms. As with the small scale examples in [6], such as `heat`, this data has become a standard example on which to test algorithms. Our numerical tests were done on a Sun Ultra 80, using Matlab 6.1. Efficient implementations were done using *RestoreTools*, a Matlab package for image restoration [8].

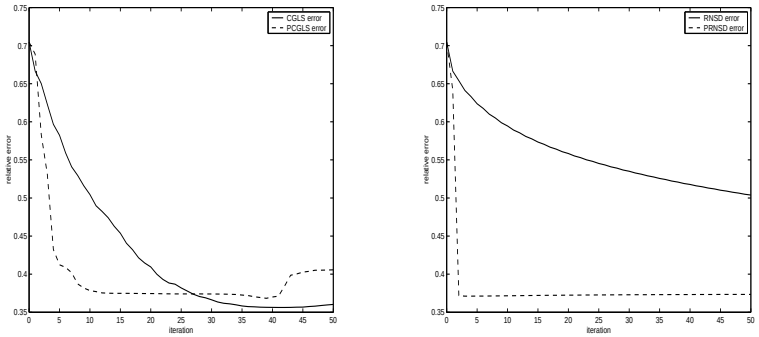
Figure 4.2 shows the relative errors for each of the algorithms, with and without preconditioning. We emphasize that the preconditioner was constructed as

[1] This data is included in the image restoration package, *RestoreTools*, which can be obtained from http://www.mathcs.emory.edu/~nagy/RestoreTools. Examples showing how to use the



(a) True image. (b) Blurred image.

Figure 4.1: Satellite image data.



(a) CGLS/PCGLS convergence. (b) RNSD/PRNSD convergence.

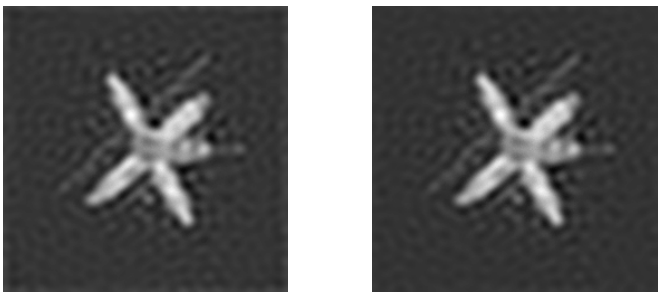
Figure 4.2: Convergence history of the methods, using the satellite image data.

if we did not know the true solution, using the GCV method to choose the truncation index (for an explanation on how this is done, see [8, Example 4.1]), thus simulating how the algorithms perform on a large scale, realistic problem. We see that the convergence behavior is similar to what was observed in Section 3. Figure 4.3 shows the PCGLS and PRNSD restorations at the 10th iteration.

5 Concluding Remarks

Without preconditioning, the very slow convergence rate of steepest descent makes it impractical to use for large scale linear systems. However, we have shown that with proper preconditioning, steepest descent may be preferable to the conjugate gradient method when solving discrete ill-posed problems. Not only is the convergence rate (to a good solution) competitive, but the semi-convergence behavior (i.e., increase in relative error) of steepest descent may be preferable for applications where it is difficult to determine adequate stopping criteria. Although we have shown that the semi-convergence behavior of the

data in iterative methods may be found in an associated manuscript [8].



(a) PCGLS restoration. (b) PRNSD restoration.

Figure 4.3: Restorations computed using PCGLS and PRNSD at the 10th iteration.

algorithms seems to be related to the filter factors, more work is needed to establish a precise relationship. This is a subject of our continuing research.

REFERENCES

1. Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, PA, 1996.
2. R. H. Chan and M. K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Review*, 38:427–482, 1996.
3. P. J. Davis. *Circulant Matrices.* Wiley, New York, 1979.
4. H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems.* Kluwer Academic Publishers, Dordrecht, 2000.
5. M. Hanke, J. G. Nagy, and R. J. Plemmons. Preconditioned iterative regularization. In L. Reichel, A. Ruttan, and R. S. Varga, editors, *Numerical Linear Algebra*, pages 141–163. de Gruyter, Berlin, 1993.
6. P. C. Hansen. Regularization tools: A Matlab package for the analysis and solution of discrete ill-posed problems. *Numerical Algorithms*, 6:1–35, 1994.
7. P. C. Hansen. *Rank-deficient and discrete ill-posed problems.* SIAM, Philadelphia, PA, 1997.
8. K. P. Lee, J. G. Nagy, and L. Perrone. Iterative methods for image restoration: A Matlab object oriented approach. *http://www.mathcs.emory.edu/∼nagy/RestoreTools*, 2002.
9. Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, 1996.
10. C. R. Vogel. Solving ill-conditioned linear systems using the conjugate gradient method. Technical report, Department of Mathematical Sciences, Montana State University, 1987.
11. C. R. Vogel. *Computational Methods for InverseProblems.* SIAM, Philadelphia, PA, 2002.