Accelerating Convergence of Iterative Image Restoration Algorithms

James G. Nagy

Mathematics and Computer Science
Emory University
Atlanta, GA 30322, USA
nagy@mathcs.emory.edu

ABSTRACT

Iterative methods are often used for applications in science and engineering to solve very large scale linear systems. Efficiency of an iterative method depends on the amount of computation needed per iteration, as well as on the number of iterations needed to reconstruct the desired approximate solution. Convergence speed can be accelerated using a technique called *preconditioning*. Although preconditioning is used in many applications, its use in image restoration has been limited. This paper describes a technique for preconditioning iterative image restoration algorithms. A particular conjugate gradient type iterative method with Tikhonov regularization is used to illustrate the effectiveness of the preconditioning scheme. Discussion of MATLAB software implementing the methods is also provided.

1. INTRODUCTION

Image restoration is the process of reconstructing an approximation of an image from blurred and noisy measurements. The image formation process is typically modeled as

$$\mathbf{g} = \mathbf{H}\mathbf{f}_{\text{true}} + \boldsymbol{\eta}\,, \tag{1}$$

where $\mathbf{f}_{\text{true}}$ is a vector representing the true image, $\mathbf{H}$ is an ill-conditioned matrix that models the blurring operation, $\boldsymbol{\eta}$ is a vector representing additive noise, and $\mathbf{g}$ is a vector representing the observed, blurred and noisy image. We assume $\mathbf{H}$ is known, at least implicitly. In addition, we assume that $\boldsymbol{\eta}$ is a combination of background and readout noise, where the background noise is modeled as a Poisson random process with fixed Poisson parameter β, and the readout noise is modeled as a Gaussian random process with mean 0 and fixed variance σ^2 [22, 23]. Thus, given $\mathbf{H}$, $\mathbf{g}$, and possibly statistical information about the noise, the aim is to compute an approximation $\mathbf{f}$ of $\mathbf{f}_{\text{true}}$.

Two aspects of the image restoration problem make it computationally challenging:

- The problem is large scale. If the images contain $m \times n$ pixels, then $\mathbf{f}, \mathbf{g}, \boldsymbol{\eta} \in \Re^N$ and $\mathbf{H} \in \Re^{N\times N}$, where $N = mn$. Values of $m \approx n \approx 10^3$, and hence $N \approx 10^6$, are not unusual. Fortunately the matrix $\mathbf{H}$ can usually be represented by a compact data structure. For example, if the blur is spatially invariant, then the the blur can be represented by a point spread function (PSF) [1, 11]. Approximation techniques for more complicated spatially variant blurs include geometrical coordinate transformations, sectioning, and PSF interpolation; see [18, 19] and the references there in.

- The matrix $\mathbf{H}$ is severely ill-conditioned, with singular values decaying to, and clustering at 0. This means that *regularization* is needed to avoid computing solutions that are corrupted by noise. Regularization can be enforced through well-known techniques such as Wiener filtering and Tikhonov regularization, and/or by incorporating constraints such as nonnegativity [6, 12, 10, 24].

Efficient implementation of an image restoration algorithm is obtained by exploiting structure of the matrix $\mathbf{H}$. For example, if the blur is spatially invariant *and* we assume periodic boundary conditions, then $\mathbf{H}$ is a block circulant matrix with circulant blocks. In this case many image restoration algorithms, such as the Wiener filter, can be implemented in the Fourier domain, using fast Fourier transforms (FFT). If spatial invariance is a poor approximation of the actual blur, or periodic boundary conditions are poor approximations to the actual true image scene, then the quality of reconstructions will be limited. Moreover, it is not possible to incorporate additional constraints, such as nonnegativity, into simple filtering methods.

Iterative image restoration algorithms have many advantages over simple filtering techniques. Iterative methods can be very efficient for spatially invariant as well as spatially variant blurs, they can incorporate a variety of boundary conditions, and can more easily incorporate additional constraints, such as nonnegativity [2, 16]. The cost of an iterative scheme depends on the amount of computation needed per iteration, as well as on the number of iterations needed to reach a good restoration of the image. Much work has been done to optimize cost per iteration, for both serial and parallel implementations. However, very little work has been done to develop robust schemes to accelerate convergence.

Preconditioning is a classical approach used in many areas of scientific computing to accelerate convergence of iterative methods. However, if not done carefully, preconditioning can lead to erratic convergence behavior that results in fast convergence to a poor approximate solution. In this paper we show how to overcome these difficulties for image restoration. Specifically, we describe a robust preconditioning scheme, where the preconditioner is constructed from the PSF and noise properties. To avoid erratic convergence behavior, regularization is naturally incorporated into the construction of the preconditioner. Although there is an additional cost when using preconditioning, we show that for typical iterative methods, such as conjugate gradients, the number of iterations can be reduced dramatically, resulting in a substantial reduction in overall cost of the iterative scheme.

This paper is outlined as follows. In Section 2 we describe a preconditioning approach that can be used for iterative image restoration algorithms, and in Section 3 we describe a particular conjugate gradient type iterative method that solves a Tikhonov regularization problem. Throughout the paper we also describe a set MATLAB tools we have developed that allow for easy experimentation of iterative methods and our preconditioning approach. The paper concludes with some numerical experiments in Section 4 illustrating how to use our MATLAB tools, and how our iterative methods and preconditioning schemes perform an some sample imaging data.

2. ITERATIVE METHODS AND PRECONDITIONING

In this paper we consider iterative image restoration methods that have the following general form:

$\mathbf{f}_0$ = initial estimate of $\mathbf{f}$
`for` $k = 0, 1, 2, \ldots$
- $\mathbf{f}_{k+1}$ = computations involving $\mathbf{f}_k$, $\mathbf{H}$ and other intermediate quantities
- determine if stopping criteria are satisfied

`end`

Most well-known iterative methods have this basic form, including conjugate gradient type schemes, the expectation-maximization method (sometimes referred to as the RichardsonLucy method), and many others. Regularization can be enforced in a variety of ways, including Tikhonov, iteration truncation, as well as mixed approaches [6, 10, 24]. Since any one iterative method is not optimal for all image restoration problems, the study of iterative methods is an important and active area of research. The specific computational operations that are required to update $\mathbf{f}_{k+1}$ at each iteration depend on the particular iterative scheme being used, but the most intensive part of these computations usually involves matrix vector products with $\mathbf{H}$ (and with $\mathbf{H}^T$ for unsymmetric matrices).

2.1. MATLAB Software for Iterative Image Restoration

Implementation of even the most basic iterative method for image restoration is not trivial, especially if we want to have the flexibility to use a variety of blurring operators and boundary conditions. We have developed a MATLAB toolbox to simplify the process of using iterative methods for image restoration. The software can be found at `http://www.mathcs.emory.edu/~nagy/RestoreTools`, with the original implementation described in [16]. The software uses an object oriented approach to hide the difficult implementation details from the user. This paper includes some information on the important aspects of the software in case readers wish to test our iterative methods and preconditioning techniques on their own data. We do not provide any implementation details, but rather just an overview of the more important tools in the software, and examples on how to use them.

For iterative methods, probably the most important object is the `psfMatrix`, which defines the matrix $\mathbf{H}$ implicitly using a compact data structure. Given an array containing a PSF, the simple MATLAB statement

```
>> H = psfMatrix(PSF);
```

constructs an object containing information about the blurring operator. Some preprocessing is done to prepare `H` for efficient matrix-vector multiplications, and the $*$ operator is overloaded so that an operation such as $\mathbf{g} = \mathbf{Hf}$ can be computed with the simple statement:

```
>> g = H*f;
```

The above example assumes compatibility between $\mathbf{H}$ and $\mathbf{f}$, `f` is an $m \times n$ array containing an image, and the result after multiplication is another $m \times n$ image `g`. We remark that the constructor routine `psfMatrix` can accept additional input parameters, including a boundary condition. The default boundary condition is "reflexive", which is typically much better (than zero and periodic boundary conditions) at reducing ringing effects if significant details are located near the edges of the observed image.

An advantage of using this object oriented approach, with operator overloading, is that iterative methods developed in the scientific computing community typically use matrix-vector notation. With our `psfMatrix`, these iterative methods can be easily used for image restoration problems, in general, without reordering data, or without the need to write "translation" wrapper routines. Moreover, the constructor function `psfMatrix` can also be used with some spatially variant blurs; for more details see [16].

2.2. Classical Approach to Preconditioning

Speed of convergence of iterative methods is typically dictated by certain spectral properties of the matrix $\mathbf{H}$. *Preconditioning* refers to a process of modifying the spectral properties of the matrix to accelerate convergence. Preconditioning is often presented in the context of solving linear systems $\mathbf{Hf} = \mathbf{g}$. The standard approach to preconditioning is to construct a matrix, $\mathbf{P}$, that satisfies the following properties:

- It should be relatively inexpensive to construct $\mathbf{P}$.
- It should be relatively inexpensive to solve linear systems of the form $\mathbf{Pz} = \mathbf{w}$.
- The preconditioned system should satisfy $\mathbf{P}^{-1}\mathbf{H} \approx \mathbf{I}$.

Then instead of applying the iterative method to the linear system $\mathbf{Hf} = \mathbf{g}$, we apply it to the modified system $\mathbf{P}^{-1}\mathbf{Hf} = \mathbf{P}^{-1}\mathbf{g}$. This means that the most intensive part of the computation at each iteration is matrix-vector multiplications with $\widehat{\mathbf{H}} = \mathbf{P}^{-1}\mathbf{H}$, or equivalently, matrix-vector multiplications with $\mathbf{H}$ and linear system solves with $\mathbf{P}$. Thus, the first two of the aforementioned requirements in constructing a preconditioner are related to the additional computational costs of preconditioning; constructing $\mathbf{P}$ is a one time cost, where as linear system solves with $\mathbf{P}$ (and with $\mathbf{P}^T$ for unsymmetric problems) are required at each iteration. The last requirement determines the speed of convergence; better approximations $\mathbf{P}^{-1}\mathbf{H} \approx \mathbf{I}$, usually imply faster convergence.

2.3. Filtering Techniques as Preconditioners

Note that if we design a preconditioner such that the singular values of $\mathbf{P}^{-1}\mathbf{H}$ are clustered around 1, then $\mathbf{P}^{-1}\mathbf{H} \approx \mathbf{I}$. That is, more singular values clustered around one, as well as tighter clusters, usually implies faster convergence. Although this approach works well for well-posed problems, it does not work well for ill-posed problems such as image restoration. Indeed, for ill-posed problems, the large singular values correspond to signal information we want to reconstruct, while small singular values correspond to noise information we do not want to reconstruct. By clustering all singular values around one, the signal and noise information becomes mixed together, and it is impossible for the iterative method to distinguish between signal information and noise information. In this situation we get fast convergence to the (noise corrupted) inverse solution.

An alternative approach for ill-posed problems, proposed in [9], is to construct a preconditioner that clusters only the *large* singular values around one. We first explain the basic idea in the ideal situation where we can compute a singular value decomposition (SVD) of $\mathbf{H}$. The SVD is defined as

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{\Sigma}$ =diag$(\sigma_1, \sigma_2, \ldots, \sigma_N)$ is a diagonal matrix containing the singular values of $\mathbf{H}$. If we define $\mathbf{P}_\tau = \mathbf{U}\mathbf{\Sigma}_\tau\mathbf{V}^T$, where $\mathbf{\Sigma}_\tau$ =diag$(\sigma_1, \ldots, \sigma_t, 1, \ldots 1)$, then

$$\mathbf{P}_\tau^{-1}\mathbf{H} = \mathbf{V}\mathbf{\Delta}V^T,$$

where $\mathbf{\Delta} = \text{diag}(1, \ldots, 1, \sigma_{t+1}, \ldots, \sigma_N)$. That is, the first t "large" singular values (e.g., those corresponding to the signal subspace) of the preconditioned system are clustered at one, and are well separated from the remaining "small" singular values (those corresponding to the noise subspace). Determining how to separate "large" and "small" singular values is related to determining regularization parameters. Various techniques can be used, including the discrete Picard condition, L-curve, and generalized cross validation (GCV); see [9, 8, 16] for more details.

Computing the SVD of $\mathbf{H}$ is typically too expensive for large scale problems, so to use this approach for image restoration we must be able to find an efficient approach to approximate it. A general approach is to choose, *a priori*, orthogonal (or unitary, if we want to consider complex bases) matrices $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$, and determine a diagonal matrix $\widehat{\mathbf{\Sigma}}$ such that

$$\widehat{\mathbf{\Sigma}} = \arg\min_{\mathbf{\Sigma}} \|\widehat{\mathbf{U}}^T\mathbf{H}\widehat{\mathbf{V}} - \mathbf{\Sigma}\|_F$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and where the minimization is done over all diagonal matrices, $\mathbf{\Sigma}$. We then construct $\mathbf{P}_\tau$ using $\widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{V}}^T$.

To make this approach efficient, the construction of, and computations with $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ should be inexpensive. Several approaches have been proposed in the literature:

- By choosing $\widehat{\mathbf{U}} = \widehat{\mathbf{V}} = \mathcal{F}$, the discrete Fourier transform matrix, computations with $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ can be done very efficiently with FFTs. The resulting approximation, $\widehat{\mathbf{H}} = \widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{V}}^T$ is a block circulant matrix with circulant blocks, and is the best such approximation; see [4] for more details. The cost of constructing the approximation, and computations with the preconditioner are $O(N \log N)$.

- Instead of using the FFT basis to build $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$, we could use another fast transform, such as the discrete cosine transform (DCT) [4]. As with FFTs, construction of this approximation, and computations with the resulting preconditioner are $O(N \log N)$.

- Another alternative is use a separable approximation of $\mathbf{U}$ and $\mathbf{V}$, so that $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ have the form

$$\widehat{\mathbf{U}} = \mathbf{U}_1 \otimes \mathbf{U}_2 \quad \text{and} \quad \widehat{\mathbf{V}} = \mathbf{V}_1 \otimes \mathbf{V}_2$$

where $\otimes$ denotes Kronecker product. This approximation can be obtained by finding the best separable (i.e., x-y) approximation of the PSF. Construction of this approximation, and computations with it, are $O(N^{3/2})$. This is slightly more than the $O(N \log N)$ computations of the fast transforms, but it is still very efficient, and a separable basis may prove to be a much better approximation than the FFT or DCT for some problems [13, 14, 17].

Note that the cost of matrix vector multiplications with $\mathbf{H}$ can usually be done with FFTs (including the spatially variant case; see [18, 19]), so even if preconditioning is not used, the cost of each iteration is at least $O(N \log N)$. Thus, if convergence of the iterative method is much faster when using preconditioning, there can be a dramatic overall savings in computational cost compared to using no preconditioning.

We refer to the approach described in this subsection as a *filtering preconditioner* because the idea is very similar to applying an approximate pseudo-inverse filter at each iteration. These preconditioners can be constructed for both spatially invariant and spatially variant blurs [16]. It is not possible to say that one approach is better than the others; the optimal approach depends on the PSF as well as on the image data.

2.4. MATLAB Software for Preconditioning

Analogous to the function `psfMatrix`, our software contains a constructor function `psfPrec` that looks at all three approximation techniques described in the previous subsection, and chooses the one that provides the best approximation. The following simple MATLAB statement can be used to construct the preconditioner:

```
>> P = psfPrec(H, g);
```

If no additional input parameters are provided to `psfPrec`, then determination of the separation between large and small singular values is done using GCV. The user can choose an alternative separation by specifying a third input parameter τ, where $\sigma_t \geq \tau > \sigma_{t+1}$. For example, if we want $\tau = 0.001$, then we use the MATLAB statement

```
>> P = psfPrec(H, g, 0.001);
```

3. A CONJUGATE GRADIENT TYPE ITERATIVE METHOD

To focus the discussion in this paper, we describe an iterative scheme called *Hybrid Bidiagonalization Regularization* (HyBR). This is essentially a conjugate gradient type algorithm that solves a Tikhonov regularized least squares problem, but has the advantage that it can refine the choice of the regularization parameter during the iteration process.

As previously mentioned, because image restoration is an ill-posed problem, regularization is needed to avoid computing solutions that are horribly corrupted by noise. Probably the most well known approach is Tikhonov regularization, where the standard least squares problem involving $\mathbf{H}$ and $\mathbf{g}$ is modified as

$$\min_{\mathbf{f}} \left\{ \|\mathbf{H}\mathbf{f} - \mathbf{g}\|_2^2 + \lambda^2 \|\mathbf{L}\mathbf{f}\|_2^2 \right\} \tag{2}$$

The matrix $\mathbf{L}$ is called a regularization operator, and is often chosen to be the identity matrix, or a discrete approximation of a differentiation operator. The parameter, λ, controls the amount of regularization; if it is chosen too small then the computed solution is noisy, and if it is chosen too large, then the computed solution is too smooth. Most iterative methods can be used to solve the regularized least squares problem (2), provided we know a good value for λ. However, λ is problem dependent and methods to estimate it from the given data, such as GCV, can be very expensive to implement.

3.1. The HyBR Method

HyBR is an efficient iterative implementation of Tikhonov regularization, where the regularization parameter λ is estimated from information computed during the iteration procedure. The scheme is based on Lanczos bidiagonalization, and for the rest of this section we assume $\mathbf{L} = \mathbf{I}$ (i.e., standard Tikhonov regularization). Given a matrix $\mathbf{H}$ and vector $\mathbf{g}$, the kth-iteration of Lanczos bidiagonalization ($k = 1, \ldots, N$) computes an $N \times (k+1)$ matrix $\mathbf{W}_k$, an $N \times k$ matrix $\mathbf{Y}_k$, an $N \times 1$ vector $\mathbf{y}_{k+1}$, and a $(k+1) \times k$ bidiagonal matrix $\mathbf{B}_k$ such that

$$\begin{aligned} \mathbf{H}^T\mathbf{W}_k &= \mathbf{Y}_k\mathbf{B}_k^T + \alpha_{k+1}\mathbf{y}_{k+1}\mathbf{e}_{k+1}^T \\ \mathbf{H}\mathbf{Y}_k &= \mathbf{W}_k\mathbf{B}_k. \end{aligned}$$

where $\mathbf{e}_{k+1}$ denotes the $(k+1)$st unit vector. Matrices $\mathbf{W}_k$ and $\mathbf{Y}_k$ have orthonormal columns, and the first column of $\mathbf{W}_k$ is $\mathbf{g}/\|\mathbf{g}\|$.

Given these relations, we can approximate the Tikhonov regularization problem (2) by the *projected* LS problem

$$\begin{aligned} \min_{\mathbf{f} \in R(\mathbf{Y}_k)} \left\{ \|\mathbf{H}\mathbf{f} - \mathbf{g}\|_2^2 + \lambda^2 \|\mathbf{f}\|_2^2 \right\} &= \min_{\mathbf{z}} \left\{ \|\mathbf{B}_k\mathbf{z} - \mathbf{W}_k^T\mathbf{g}\|_2^2 + \lambda^2 \|\mathbf{z}\|_2^2 \right\} \\ &= \min_{\mathbf{z}} \left\{ \|\mathbf{B}_k\mathbf{z} - \gamma\mathbf{e}_1\|_2^2 + \lambda^2 \|\mathbf{z}\|_2^2 \right\} \end{aligned} \tag{3}$$

where $\gamma = \|\mathbf{g}\|$, and choose our approximate solution as $\mathbf{f}_k = \mathbf{Y}_k\mathbf{z}$. Thus we can build an iterative method where at each iteration we solve a regularized least squares problem involving a bidiagonal matrix $\mathbf{B}_k$. Notice that since the dimension of $\mathbf{B}_k$ is very small compared to $\mathbf{H}$, it is much easier to solve for $\mathbf{z}$ in equation (3) than it is to solve for $\mathbf{f}$ in equation (2). More importantly, when solving equation (3) we can use sophisticated parameter choice methods to find λ at each iteration.

This projection based approach was used by Paige and Saunders [21] to develop the LSQR algorithm, which is an implementation of the conjugate gradient method for least squares problems without regularization. The idea of using regularization for the projected problem was proposed independently by O'Leary and Simmons [20] and Björck [3]. A variety of authors have considered computational issues [3, 15, 7]. Recent work has also been done on robust methods for choosing regularization parameters and reliable stopping iterations [5].

3.2. MATLAB Software for HyBR

Implementation of HyBR is nontrivial, especially if one would like to include a robust scheme for choosing the regularization parameter, and a scheme for determining when the solution has converged. We have recently added an implementation of HyBR to our MATLAB image restoration software. It can be used with the very simple MATLAB statement (no preconditioning)

```
>> f = HyBR(H, g);
```

or (with preconditioning)

```
>> f = HyBR(H, g, P);
```

There is also another function `HyBRset` that can be used to change many default settings. Discussion of `HyBRset` would be tedious in this paper, so we refer to the software documentation for further details (e.g., using the MATLAB statement `help HyBRset`).

Additional information, with examples using HyBR on other types of ill-posed problems, can be found at `www.mathcs.emory.edu/~nagy/WGCV`.

4. NUMERICAL EXAMPLES AND RESULTS

In this section we illustrate how preconditioning can be used to effectively accelerate convergence of the HyBR method. For our tests we use data that was developed at the U.S. Air Force Phillips Laboratory, Laser 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 image, PSF, and blurred image are shown in Fig. 1. Using the statistical model for noise discussed in Section 1, noise was generated with the sky background parameter $\beta = 10$ and readout noise with standard deviation $\sigma = 5$. The noise is scaled to obtain a signal to noise ratio (SNR) of approximately 100, which corresponds to a noise power that is 1% of the signal power.

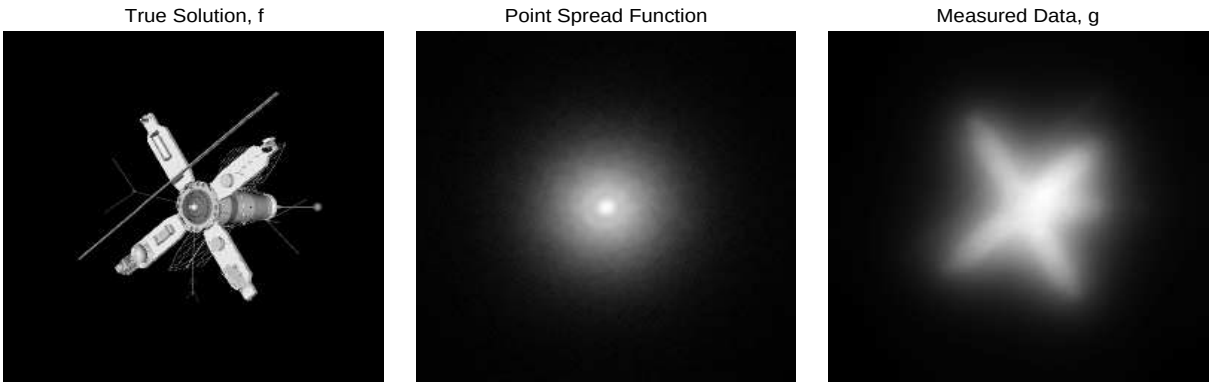

Figure 1. Data used in the numerical experiments. The true image is shown on the left, the PSF is in the middle, and measured image is shown on the right.

4.1. Results with Default Parameters

Recall that construction of the preconditioner requires choosing a truncation parameter. In addition, the iterative method HyBR must choose Tikhonov regularization parameters, and it must determine an appropriate stopping iteration. As discussed in the sections on MATLAB software, we have developed implementations that choose default values for each of these parameters. In this subsection we illustrate how the software performs when using these default values on the test problem shown in Fig. 1. Given an array containing the PSF and an array containing the blurred noisy image $\mathbf{g}$, we use the following MATLAB statements to compute a reconstruction using HyBR with no preconditioning, and a reconstruction using HyBR with preconditioning:

```
>> H = psfMatrix(PSF);
>> f = HyBR(H, g);       % Reconstruction computed without preconditioning.

>> P = psfPrec(H, g);
>> f = HyBR(H, g, P);    % Reconstruction computed with preconditioning.
```

Without preconditioning, HyBR detects convergence after 45 iterations, and *with* preconditioning HyBR detects convergence after 9 iterations. Fig. 2 shows a plot of the relative errors at each iteration:

$$\frac{\|\mathbf{f}_{\text{true}} - \mathbf{f}_k\|_2}{\|\mathbf{f}_{\text{true}}\|_2}.$$

Of course in a realistic problem we do not know $\mathbf{f}_{\text{true}}$, but the convergence history plot does illustrate the good performance of both the iterative method HyBR and the preconditioning scheme. In Fig. 3, we compare the computed reconstructions. The left image is the reconstruction after the 9th iteration of the

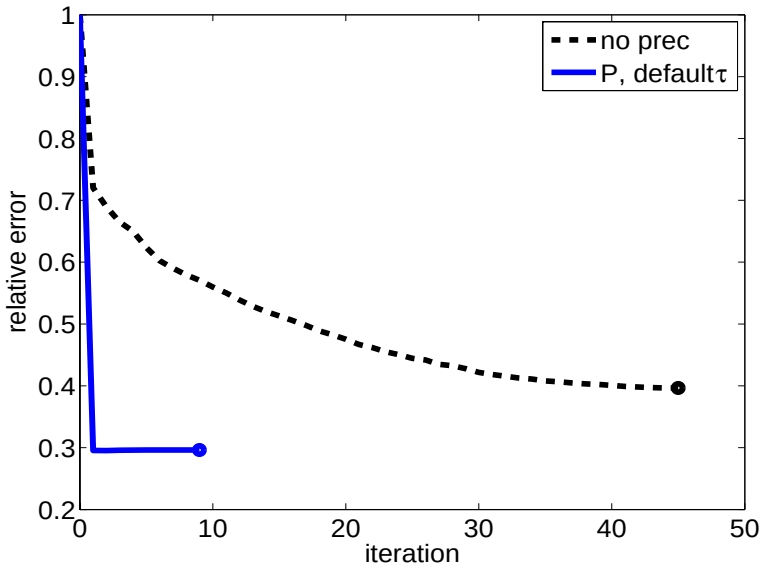

Figure 2. This plot shows the relative error, $\frac{\|\mathbf{f}_{\text{true}} - \mathbf{f}_k\|_2}{\|\mathbf{f}_{\text{true}}\|_2}$, at each iteration of the HyBR method (dashed curve) and the preconditioned HyBR method (solid curve). The solid dots at the end of each curve denote the iteration at which HyBR detected convergence.

preconditioned method, and the other images show computed reconstructions when no preconditioning is used after the 45th and (for comparison purposes) the 9th iteration. These results clearly illustrate that preconditioning can be very effective in computing a very good reconstruction much more quickly than without preconditioning.

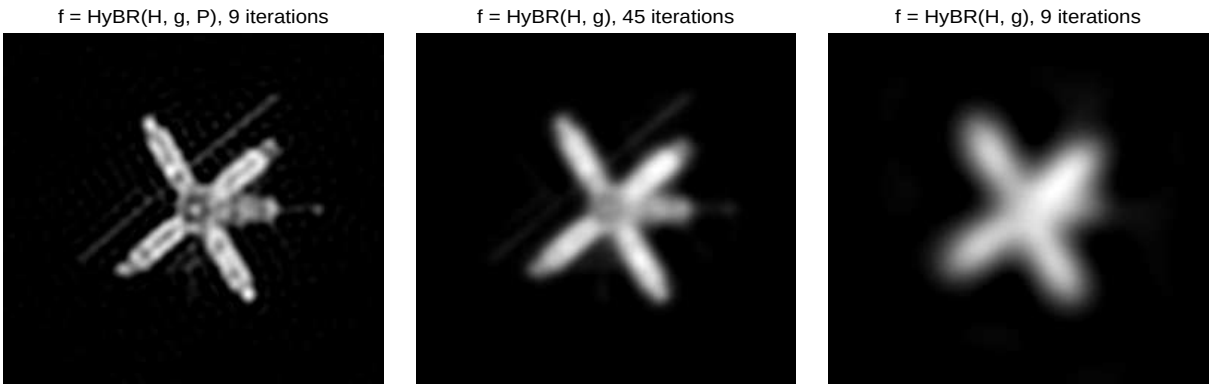

Figure 3. The images in this figure compare solutions computed by HyBR and preconditioned HyBR. The left image is the reconstruction after the 9th iteration of preconditioned HyBR, and the other images show computed reconstructions when no preconditioning is used after the 45th and the 9th iteration.

4.2. Preconditioner Dependence on Truncation

The default truncation parameter used for constructing the preconditioner is chosen using a GCV method. Instead of using the default value, we can specify a third input value to `psfPrec` (see Section 2.4):

```
>> P = psfPrec(H, g, tau);
```

Choosing τ too large means the preconditioned system will have very few singular values clustered around one, and thus we may observe very little increase in speed of convergence. However, if `tau` is chosen too small, then the preconditioned system will have many singular values clustered around one, resulting in a mixing of signal and noise information, and we will not be able to compute a very good reconstruction. These properties are illustrated in Fig. 4, where we show the convergence history when using no preconditioning, and when using preconditioning with truncation tolerances corresponding to the default, larger than the default, and smaller than the default value.

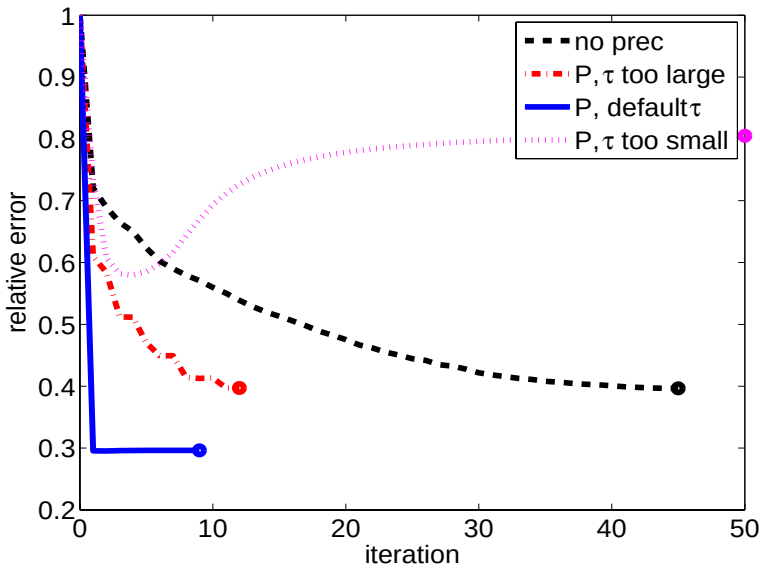

Figure 4. This plot shows the relative error, $\frac{\|\mathbf{f}_{\text{true}} - \mathbf{f}_k\|_2}{\|\mathbf{f}_{\text{true}}\|_2}$, at each iteration of the HyBR method and the preconditioned HyBR method using various truncation tolerances in constructing the preconditioner.

The plot in Fig. 4 illustrates that if we are not sure about how to choose the truncation tolerance, then it is better to choose it too large. We do not claim that the default value for the truncation parameter will always produce such optimal results for the preconditioner; there is a dependence on the data. However, we have observed that the default choice for the truncation parameter works very well for a variety of problems, including a variety of types of blurring operators (including spatially variant), noise, and objects. Furthermore, the preconditioning technique described in this paper can be used with other iterative methods, including ones that enforce nonnegativity constraints [2, 16].

5. ACKNOWLEDGMENTS

This work was supported by the United States National Science Foundation under grant DMS-05-11454.

REFERENCES

1. H. Andrews and B. Hunt. *Digital Image Restoration*. Prentice-Hall, Englewood Cliffs, NJ, 1977.
2. J. M. Bardsley and J. G. Nagy. Covariance-preconditioned iterative methods for nonnegatively constrained astronomical imaging. *SIAM J. Matrix Anal. Appl.*, 27:1184–1197, 2006.
3. Å. Björck. A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations. *BIT*, 28:659–670, 1988.
4. R. H. Chan and M. K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Review*, 38:427–482, 1996.
5. J. Chung, J. G. Nagy, and D. P. O'Leary. A weighted GCV method for Lanczos hybrid regularization. Technical Report TR-2007-004, Mathematics and Computer Science Department, Emory University, 2007.
6. H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht, 2000.
7. M. Hanke. On lanczos based methods for the regularization of discrete ill-posed problems. *BIT*, 41:1008–1018, 2001.
8. M. Hanke and J. G. Nagy. Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques. *Inverse Problems*, 12:157–173, 1996.
9. M. Hanke, J. G. Nagy, and R. J. Plemmons. Preconditioned iterative regularization for ill-posed problems. In L. Reichel, A. Ruttan, and R. S. Varga, editors, *Numerical Linear Algebra*, pages 141–163. de Gruyter, Berlin, 1993.
10. P. C. Hansen. *Rank-deficient and discrete ill-posed problems*. SIAM, Philadelphia, PA, 1997.
11. P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images: Matrices, Spectra and Filtering*. SIAM, Philadelphia, PA, 2006.
12. A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
13. J. Kamm and J. G. Nagy. Kronecker product and SVD approximations in image restoration. *Linear Algebra Appl.*, 284:177–192, 1998.
14. J. Kamm and J. G. Nagy. Optimal Kronecker product approximation of block Toeplitz matrices. *SIAM J. Matrix Anal. Appl.*, 22:155–172, 2000.
15. M. E. Kilmer and D. P. O'Leary. Choosing regularization parameters in iterative methods for ill-posed problems. *SIAM J. Matrix Anal. Appl.*, 22:1204–1221, 2001.
16. J. Nagy, K. Palmer, and L. Perrone. Iterative methods for image deblurring: A matlab object oriented approach. *Numerical Algorithms*, 36:73–93, 2004.
17. J. G. Nagy, M. K. Ng, and L. Perrone. Kronecker product approximation for image restoration with reflexive boundary conditions. *SIAM J. Matrix Anal. Appl.*, 25:829–841, 2004.
18. J. G. Nagy and D. P. O'Leary. Fast iterative image restoration with a spatially varying PSF. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations VII*, volume 3162, pages 388–399. SPIE, 1997.
19. J. G. Nagy and D. P. O'Leary. Restoring images degraded by spatially-variant blur. *SIAM J. Sci. Comput.*, 19:1063–1082, 1998.
20. D. P. O'Leary and J. A. Simmons. A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems. *SIAM J. Sci. Stat. Comp.*, 2:474–489, 1981.
21. C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft.*, 8:43–71, 1982.
22. D. L. Snyder, C. W. 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.
23. D. L. Snyder, C. W. Helstrom, and A. D. Lanterman. Compensation for readout noise in CCD images. *J. Opt. Soc. Am. A*, 12:272–283, 1994.
24. C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, PA, 2002.