Gradient Descent Methods for Large-Scale Linear Inverse Problems

By

Chen (Cool) Cheng

James Nagy

Department of Mathematics and Computer Science

Gradient Descent Methods for Large-Scale Linear Inverse Problems

by

Chen (Cool) Cheng

Advisor : James Nagy

Abstract of
A thesis submitted to the Faculty of Emory College
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2013

Abstract

Gradient Descent Methods for Large-Scale Linear Inverse Problems
By Chen (Cool) Cheng

Iterative gradient descent methods are frequently used for ill-posed inverse problems because they are suitable for large models and they are cheap to work with. In this thesis, we explore three different types of gradient descent methods: the Landweber method, method of steepest descent, and the Barzilai-Borwein method. Specifically, we also compare the efficiency of these methods to the conjugate gradient method. The thesis begins with an introduction to the history and application of the gradient descent methods and to the methods tested, and follows with convergence analysis and numerical experience on real images. Ways to accelerate and smooth the Barzilai-Borwein method are also included.

Gradient Descent Methods for Large-Scale Linear Inverse Problems

by

Chen (Cool) Cheng

Advisor : James Nagy

A thesis submitted to the Faculty of Emory College
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelors of Science with Honors

Department of Mathematics and Computer Science

2013

Contents

List of Figures

Chapter 1 Introduction

We are interested in the numerical solution of large-scale linear inverse problems. These problems occur frequently in many scientific and engineering fields, such as medical and seismic tomography [4] [19], astronomy [1] [17], and many others. A linear inverse problem can usually be solved by either direct methods or iterative methods. Even though direct methods deliver the exact solution of the given linear system, they are not suitable for real-world large-scale problems due to several reasons, which we discuss in the following section. Iterative methods, such as gradient descent methods, are often the choice for computing such problems with large number of variables. In this chapter, we present the background of gradient descent methods, and then show the mathematical model that is popular to many scientific applications.

1.1 Background

Inverse problems [8] are usually hard to define but easy to recognize. They always come paired with direct problems, which are easier to describe and work with. A simple direct problem of multiplication is shown as follows: given two numbers we find their product. It is obvious that a corresponding inverse problem is, given the product, find two factors of that given value. In general, we can describe direct problems as follows,

$$input \rightarrow process \rightarrow output,$$

where the input represents what is provided such as the two given numbers, the process represents what action needs to be taken on the input such as multiplication, and the output is the resulting solution such as the product. On the other hand, inverse problems can be described by the following,

$$input \leftarrow inverse\, process \leftarrow output,$$

where the input is what we are interested in finding. Although this is a simple example of what we are studying, many other complicated problems often occur in numerous scientific applications.

Image deblurring is one of the most important and frequently used examples of large-scale inverse problems. The goal of image deblurring is to recover the original and clear image, given the blurred image and the mathematical model of the blurring process [11]. One particular example of image deblurring arises in the technique of positron emission tomography (PET) for brain imaging [14]. The blurred image is caused by head movements during long scan times, and the goal is to reconstruct the desired image by monitoring the head movements, which essentially provides information about the blurring process. Another important example is in the area of astronomy [1] [17]. For instance, a telescope takes a blurred satellite image, due to distortion of the optical wavefront caused by atmospheric turbulence. In order to reconstruct a clearer picture, solving a large-scale inverse problem is required.

In recent years, the focus on solving these inverse problems in a more accurate and efficient way has arisen consistently. As we mentioned earlier, iterative methods are preferred to direct methods. One advantage of using iterative methods is that it becomes more efficient for large and sparse problems compared to using direct methods, because the computational cost and storage capacity cause the direct methods to slow down enormously with a large amount of variables. Another reason that we want to use iterative methods rather than direct methods is that computation can often be done more easily in parallel implementations [16]. In this thesis, we mainly focus on one specific type of iterative method, which is based on gradient descent, and will be discuss in detail in the next chapter.

1.2 Mathematical Model

In order to relate real world problems to the mathematical point-of-view, we need an appropriate mathematical model. We are interested in solving linear systems that arise from large-scale inverse problems. Such linear systems can typically be written as

$$b = Ax_{true} + \epsilon, \tag{1.1}$$

where b is known, such as a blurred image, A is an $n \times n$ ill-conditioned matrix representing the data collection operation, such as the blurring process, with singular values that decay to, and cluster at zero, x_{true} represents the true solution, and vector ϵ is random noise affecting the problem.

With this mathematical model in mind, the next step is to find a method that solves this equation. At first sight, it seems simple to find an inverse solution to (1.1), particular if A^{-1} exists and the noise ϵ is small enough to ignore, i.e., $\parallel \epsilon \parallel$ is small. In this case, we can simply assume that $x_{inv} = A^{-1}b \approx x_{true}$. However, the naive solution,

$$\begin{aligned} x_{inv} &= A^{-1}b \\ &= A^{-1}(Ax_{true} + \epsilon) \\ &= x_{true} + A^{-1}\epsilon, \end{aligned} \tag{1.2}$$

is usually incorrect. Even the naive approach gives a solution that contains the true solution x_{true}, there is an extra term contains the inverted noise $A^{-1}\epsilon$. If A is ill-conditioned and very large, which is typically the case, then $A^{-1}\epsilon$ can be very large. Therefore, the naive approach may not produce an accurate solution due to these characteristics such as ill-posedness.

We can see this challenge arises when attempting to solve the naive solution by using the singular value decomposition (SVD). Specifically, matrix A can be factored into a product of three matrices

$$A = U\Sigma V^T \tag{1.3}$$

where U and V are $n \times n$ orthogonal matrices ($U^TU = I$ and $V^TV = I$) and $\Sigma = diag(\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n)$ is a diagonal matrix containing the singular values of A. Using the singular value decomposition, we can easily see what effect the error term ϵ has on the inverse solution:

$$\begin{aligned} x_{inv} &= x_{true} + A^{-1}\epsilon \\ &= x_{true} + V\Sigma^{-1}U^T\epsilon \\ &= x_{true} + \sum_{i=1}^{n} \frac{u_i^T\epsilon}{\sigma_i} v_i. \end{aligned} \tag{1.4}$$

As we mentioned earlier, the matrix A is ill-conditioned (i.e., σ_1/σ_n is large), and, morever, the singular values, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, decay to 0. Thus, it follows that the term $(u_i^T\epsilon)/\sigma_i$ can be highly magnified by division of small singular values. The inverted noise overwhelms the desired x_{true} portion of the solution, so in general $x_{inv} = x_{true} + A^{-1}\epsilon$ is a very poor approximation of the true solution, x_{true}. In addition, the singular vectors, v_i, become increasingly oscillatory when i is large (corresponding to small singular values). These properties of ill-posed problems using SVD analysis are well know; see, for example, [10] [19] for more details.

1.3 Iterative Regulizarization

Finding an alternative method to compute an approximation of x_{true} is the next step. We must modify the solution process so that the desired approximation of x_{true} is not horribly corrupted by noise. This process is called regularization [10] [19]. One way to do this, as is discussed later, is to apply an iterative method to the least-squares problem:

$$\min_x \| b - Ax \|_2^2, \tag{1.5}$$

and stop the iteration before division of small singular values magnifies the noise. Iterative regularization approaches are usually preferred when solving large-scaled linear systems, but a serious drawback is that they suffer from semi-convergence behavior. That is, the relative error, $\| x_k - x_{true} \| / \| x_{true} \|$, where x_k is the approximate solution at the kth iteration, begins to decrease, but rise after some optimal iteration. We want to stop the iteration when we reach the bottom of the error curve, so a regularized approximation of the solution is reached. This phenomenon is shown because iterative regularization tends to first calculate the parts of the solution with large singular values, and then calculate the parts of the solution with undesired small singular values, which was discussed in the previous section. A clear representation of the principle of semi-convergence will be illustrated in Chapter 4 for conjugate gradient method.

In Chapter 2, we present the gradient descent methods, including the classic steepest descent method, the Landweber method, and the Barzilai-Borwein method, and we also briefly introduce the conjugate gradient method. Chapter 3 will discuss the convergence analysis of the methods described in Chapter 2 along with a preconditioner used on the Barzilai-Borwein method. Convergence comparisons of the various methods are also included. Finally, in Chapter 4, we apply our methods to three image restoration problems to see the effects on real-world problems. Also, a modification of the Barzilai-Borwein method is introduced.

Chapter 2
Methods

This section provides a detailed introduction to the four methods that are implemented for the experiments in this thesis. The first three methods are based on gradient descent methods; specifically, they share the same negative gradient step direction, but with different step sizes. The Landweber (LW) method uses a constant step length at each iteration; method of steepest descent (SD) uses a step length that changes at each iteration; the Barzilai-Borwein (BB) method uses a modified steepest descent step length. We expect that each method results in a different convergence rate. Subsequently, we present an alternative method, the conjugate gradient method (CG), for comparison purpose. This method has a different way to choose the step direction, which depends on the previous selection. In the rest of this chapter, we give a basic introduction to each method and provide a simple algorithm for each method.

2.1 Gradient Descent Methods

Gradient descent methods are among the most widely used methods for function optimization. As we mentioned in the last chapter, we want to minimize the least squares problem such as in equation (1.5). To give a simple definition of the gradient descent methods, picture that we are on top of a valley and we want to walk down to the bottom. In order to reach our goal, we must choose a direction along with the distance we want to walk. The gradient descent methods choose the direction to be the negative of the gradient of the function at each step. This makes perfect sense because this direction is where the function decreases most quickly. Therefore, we can see the reason why it is called the gradient descent method. Then, it is left to determine the step length. Different step lengths deliver different gradient descent methods. In this thesis, we focus on three different types, the Landweber method with constant step length, method of steepest descent and the Barzilai-Borwein method both with changing step lengths. An illustration of how general descent methods work for a simple function is shown in Figure 2.1.

The basic iteration for the gradient descent methods usually takes the form:

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)} \tag{2.1}$$

where α_k is the step length and $d^{(k)}$ is the step direction. In the case of gradient descent methods, the step direction associated with the least squares problem

$$\min_x \|b - Ax\|_2^2$$

is the normal equations residual; that is, $d^{(k)} = A^T(b - Ax^{(k)})$. Thus, the basic iteration takes the form:

$$x^{(k+1)} = x^{(k)} + \alpha_k A^T(b - Ax^{(k)}). \tag{2.2}$$

Because the matrix A in our problem is usually not symmetric, we use implementations to the normal equations. Throughout our discussion of solving least squares problems, the normal equations form will be referenced. For example, the steepest

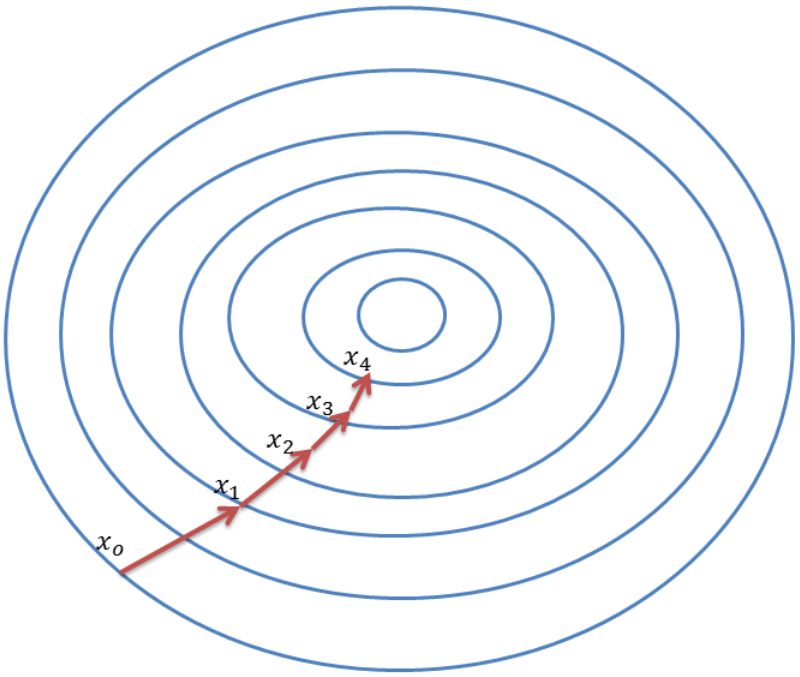


Figure 2.1: Illustration of gradient descent. The blue curves are the contour lines and the graph has a bowl shape. The search starts at x_o and then moves down the negative of the gradient with a certain step length, until it reaches close enough to the desired minimum. Notice that the gradient at each point is orthogonal to the contour line at that point.

descent method is actually the residual norm steepest descent method, and the conjugate gradient method is actually the the conjugate gradient method for least squares problems.

2.1.1 The Landweber Method

The Landweber Method, which is often called the Richardson iteration [10] [19], is one of the simplest gradient descent methods. With the step direction already established, we only need to determine the step size.

The Landweber method has a fixed step size, $\alpha \equiv \alpha_k$, for each iteration. It is not hard to show [3] that the step length must satisfy the condition,

$$0 < \alpha < \frac{2}{\sigma_{max}^2}, \tag{2.3}$$

where σ_{max} is the largest singular value of matrix A, to guarantee convergence. In order to calculate σ_{max}^2, it is easier to find a bound [7] instead of an exact value, such that

$$\sigma_{max}^2 = \| A^T A \|_2 \leq \| A \|_1 \| A \|_\infty \tag{2.4}$$

This is simple to compute because,

$$\| A \|_1 = \text{the maximum absolute column sum of the matrix A},$$

$$\| A \|_\infty = \text{the maximum absolute row sum of the matrix A}.$$

In MATLAB, we can use the built-in function, norm, to compute these matrix norms. An algorithm for the Landweber method can be stated as follows:

The Landweber Method

```
given:      $A$, $b$
choose:     initial guess for $x_o$
compute:
            $r_o = b - Ax_o$
            $d_o = A^T r_o$
            for k = 0,1,2,...
               $\alpha_k = \frac{1}{\|A\|_1 \|A\|_\infty}$
               $x_{k+1} = x_k + \alpha_k d_k$
               $r_{k+1} = b - Ax_{k+1}$
               $d_{k+1} = A^T r_{k+1}$
            end
```

2.1.2 Method of Steepest Descent

The method of steepest descent is another simple gradient descent method, where we assume the matrix A^TA is symmetric positive definite. Here, we try to minimize the error in the direction of the negative gradient, $e_{(k+1)} = x - x_{(k+1)}$, in order to choose the step length α_k, but we allow the step length to change at each iteration. Using the Cholesky factorization of A^TA and some algebra, it is not hard to show that,

$$\alpha_k = \frac{\langle d^{(k)}, d^{(k)} \rangle}{\langle w^{(k)}, w^{(k)} \rangle}, \tag{2.5}$$

where $d^{(k)} = A^T r^{(k)}$ and $w^{(k)} = Ad^{(k)}$. Before we derive a complete algorithm for this method, let us take a close look at the computational cost. To compute α_k, we need one matrix-vector multiplication per iteration. We will need another matrix-vector multiplication when we update the residual because $r_{(k+1)} = b - Ax_{(k+1)}$. Matrix-vector multiplications are generally the most expensive computation at each iteration, and we want to reduce them as much as possible. Therefore, a simple modification can reduce it to only once per iteration. That is,

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} \\ &= b - A(x_k + \alpha_k r_k) \\ &= b - Ax_k - A\alpha_k r_k \\ &= r_k - A\alpha_k r_k. \end{aligned} \tag{2.6}$$

With equation (2.6) in mind, we can write the algorithm for the method of steepest descent as follows:

The Method of Steepest Descent

```
given:      $A$, $b$
choose:     initial guess for $x_o$
compute:
            $r_o = b - Ax_o$
            $d_o = A^T r_o$
            for k = 0,1,2,...
               $w_k = Ad_k$
               $\alpha = \frac{\langle d_k, d_k \rangle}{\langle w_k, w_k \rangle}$
               $x_{k+1} = x_k + \alpha d_k$
               $r_{k+1} = r_k - \alpha w_k$
               $d_{k+1} = A^T r_{k+1}$
            end
```

An interesting fact appears when we apply the steepest descent method. That is, it approaches the minimum in a zig-zag path, because each gradient is orthogonal to the previous gradient. This particular feature is shown in Figure 2.2.

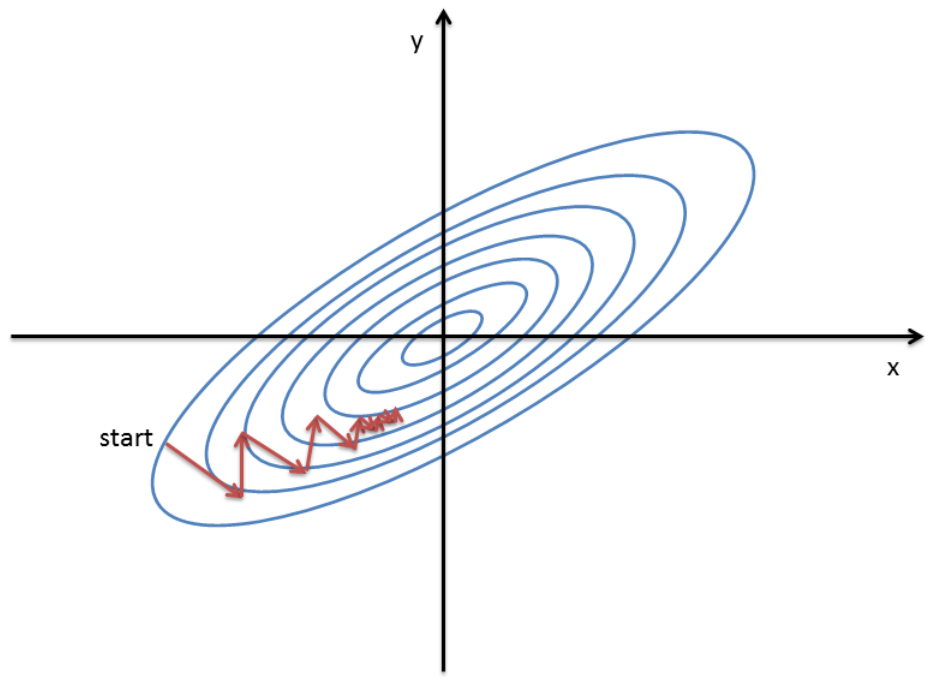


Figure 2.2: Illustration of zig-zag manner for the method of steepest descent. Beside the zig-zag fact, it can also be seen that the step length becomes smaller and smaller as the iteration proceeds.

As seen from the algorithm, this method is easy and simple to apply and it will converge whenever $A^T A$ is nonsingular [16]. That is, the method is guaranteed to converge to the minimum. This may not be efficient if we need many iterations to reach convergence, which is often the case when A is ill-conditioned, because the step length becomes extremely small. Although increasing the step length may help, the ultimate convergence behavior cannot be guaranteed. Most attempts to modify the method of steepest descent have led to the introduction of the conjugate gradient method, which we will discuss in Section 2.2. Another modified steepest descent method will be introduced in the next subsection, which seems to be comparable in practical efficiency to conjugate gradient methods [6].

2.1.3 The Barzilai-Borwein Method

In 1988, Barzilai and Borwein [2] introduced a new gradient descent method, the BB method, with a modified step length. It has been shown that the BB method is significantly better than the method of steepest descent [15], though the difference in choosing the step length is relatively simple. Barzilai and Borwein presented the step length in two alternative formulae:

$$\alpha_k = \frac{\langle h^{(k-1)}, h^{(k-1)} \rangle}{\langle h^{(k-1)}, y^{(k-1)} \rangle}, \tag{2.7}$$

and

$$\alpha_k = \frac{\langle y^{(k-1)}, h^{(k-1)} \rangle}{\langle y^{(k-1)}, y^{(k-1)} \rangle}, \tag{2.8}$$

where $y^{(k-1)} = r^{(k)} - r^{(k-1)}$, and $h^{(k)} = \alpha_k r^{(k)}$. With some simple algebra, we can express it in the following form:

$$\alpha_k = \frac{\langle d^{(k-1)}, d^{(k-1)} \rangle}{\langle w^{(k-1)}, w^{(k-1)} \rangle}. \tag{2.9}$$

At first glance, the step size for the BB method and the steepest descent method are very similar. In the BB method, we update the step length from the previous gradient vector. This is in contrast to the method of steepest descent, in which the step length is calculated at the current gradient vector. Therefore, there will be a slight difference in the algorithm, which we will show as follows:

The Barzilai-Borwein Method

```
given:      $A$, $b$
choose:     initial guess for $x_o$
compute:
            $r_o = b - Ax_o$
            $d_o = A^T r_o$
            for k = 0,1,2,...
               $w_k = Ad_k$
               if $k = 0$
                  $\alpha = \frac{\langle d_k, d_k \rangle}{\langle w_k, w_k \rangle}$
               else
                  $\alpha = \frac{\langle d_{k-1}, d_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle}$
               $d_{k-1} = d_k$
               $x_{k+1} = x_k + \alpha d_k$
               $r_{k+1} = r_k - \alpha w_k$
               $d_{k+1} = A^T r_{k+1}$
               $w_{k-1} = w_k$
            end
```

Regardless of how similar the BB method is to the classic steepest descent method, the difference in convergence behavior is surprisingly large, as we will see later. That is, BB delivers a much faster convergence rate than SD does.

2.2 The Conjugate Gradient Method

We have looked at iterative methods with the search direction to be the negative of the gradient. Now, we briefly explore an alternative iterative method, the conjugate gradient method, with a more complicated choice of the step direction. It was initially proposed by Magnus Hestenes and Eduard Stiefel [12] in 1952. Since then, many researchers have worked on this algorithm. Today, it is still the standard algorithm for solving symmetric and positive definite (SPD) linear systems involving large and sparse matrices, because it handles large and ill-conditioned linear systems in a more efficient way than other iterative methods such as the gradient descent based methods.

In this section, we will simply present the result of the conjugate gradient method. If you want to learn how and why the conjugate gradient method is a rich and elegant algorithm, you can read Shewchuk's paper [18] on this method, and any numbers of standard textbooks on numerical analysis, such as [16]. He presented this brilliant method in a way that almost anyone can understand. In the conjugate gradient method, search directions are constructed by conjugation of the residuals, and step length is chosen such that the error is minimized. Here is a simple algorithm:

The Conjugate Gradient Method

```
given:      $A$, $b$
choose:     initial guess for $x_o$
compute:
            $d_o = b$
            $r_o = A^T b$
            $p_o = r_o$
            $t_o = A p_o$
            for k = 0,1,2,...
              $\alpha_k = \| r_{k-1} \|^2 / \| t_{k-1} \|^2$
              $x_k = x_{k-1} + \alpha_k p_{k-1}$
              $d_k = d_{k-1} - \alpha_k t_{k-1}$
              $r_k = A^T d_k$
              $\beta_k = \| r_k \|^2 / \| r_{k-1} \|^2$
              $p_k = r_k + \beta_k p_{k-1}$
              $t_k = A p_k$
            end
```

Chapter 3

Convergence Analysis

We mentioned in the discussion of the last chapter that the convergence rate depends on different factors such as the condition of the matrix studied, the step length and step direction. It is important for us to understand the convergence for each method in order to compare the efficiency of each method. In this section, we look at the convergence rate for all the methods studied. In order to test for convergence, a mathematically rigorous convergence analysis is often performed. Here, we will not discuss in detail how each proof is done. Rather, we will introduce some background first and then simply state the result. Subsequently, we will try to accelerate the convergence rate of the BB method using some preconditioning techniques.

3.1 Original Methods

In this section, we first present some essential mathematical background material needed to understand the convergence analysis. Then we review what is known about the convergence of each method of interest. A preconditioned BB method is also reviewed in the end of this section.

3.1.1 Some Background

We have established the mathematical model in equation (1.1), $b = Ax_{true} + \epsilon$, such that the matrix A is severely ill-conditioned, with singular values decaying to, and clustering at 0. Recall that the singular value decomposition from Section 1.2, $A = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix containing all the singular values of A. In order to compute an accurate approximation of x_{true}, regularization is needed. One class of regularization methods, called filtering, can be formulated to compute a more accurate solution to the inverse problem. From equation (1.4), we know that the poor approximation on the inverse solution is caused by small singular values, thus a good solution would limit or filter out the unwanted portion of the solution. Specifically, this process can be written as

$$x_{filt} = \sum_{i=1}^{N} \phi_i \frac{u_i^T b}{\sigma_i} v_i, \tag{3.1}$$

where the filter factors, ϕ_i, are chosen such that $\phi_i \approx 1$ for large singular values and $\phi_i \approx 0$ for small singular values. That is, the small singular value components of the solution are filtered out, while the large singular value components of the solution are reconstructed. One popular filtering method is called truncated SVD (TSVD); we discuss its application to the BB method in Section 3.2.

Another important background on iterative methods we should stress here is preconditioning. It is often used to accelerate the rate of convergence by reducing the number of iterations needed to compute the optimal solution. To develop any preconditioned method solving $Ax = b$, we usually construct a matrix P, such that the singular values of $P^{-1}A$ are clustered around a point away from zero. Faster convergence is often caused by more singular values clustering around one point. A detailed preconditioned BB method is introduced in Section 3.2.

3.1.2 Gradient Descent Methods

Let us recall from Section 2.1.1 that the Landweber method is a basic iterative method with a gradient descent direction and a constant step size. The convergence analysis of the Landweber method can be done [10] using an SVD filtering method as described in equation (3.1). If the initial guess for x is equal to zero, then the filter factors, ϕ_i, are given by

$$\phi_i^{(k)} = 1 - (1 - \alpha\sigma_i^2)^k, \quad i = 1, 2, \ldots, n, \tag{3.2}$$

where k represents the iteration number, and ϕ_i is the i-th largest singular value of A. In order to see how SVD filtering helps us understand convergence properties, let us assume $\phi_1 = \alpha = 1$. With equation (3.2) in mind, it is not hard to show that

$$0 \approx \phi_n^{(k)} \leq \phi_{n-1}^{(k)} \leq \cdots \leq \phi_1^{(k)} \approx 1,$$

which is what we expected from the properties of the SVD filtering method such that $\phi_1 \approx 1$ for large singular values and $\phi_n \approx 0$ for small singular values. However, slow convergence is still expected. This happens when we need to reconstruct a solution corresponding to intermediate singular values, which may take many iterations. For example, an intermediate singular value $\phi_i = 0.1$ makes the filter factors $\phi_i^{(k)} = 1 - (0.99)^k$. In order to obtain $\phi_i^{(k)} \approx 0$, we need k to be very large. From this analysis, we expect the Landweber method to have slow convergence. However, this method can be accelerated with preconditioning, and some preconditioned Landweber methods are described in [3].

For the method of steepest descent, as in the case of the Landweber method, we take a step direction that is the negative gradient of the function, but instead of fixed step length we use variable step length. As we successfully used the SVD filtering method to analyze the convergence rate for the Landweber method, why don't we try it for steepest descent. However, it is not easy to provide a theoretical analysis of the filtering properties of steepest descent because the step length changes at each iteration. However, we can investigate the convergence behavior of steepest descent. If A is SPD, it has proven [16] that convergence is guaranteed for any initial guess x_o. This result is a consequence of the Kantorovich inequality, which is presented in details in [16]. Although this method is guaranteed to find the minimum after at least an infinite number of iterations, the iteration number may be very large when A is ill-conditioned, which is exactly the case for our problems. This is because the step size is extremely small towards the minimum. Therefore, steepest descent tends to converge slowly as well, just like in the Landweber method. It can also be shown that the filtering properties of steepest descent are similar to Landweber; see [13].

The previous two gradient descent methods both have slow convergence rates. Barzilai and Borwein [2] presented a new choice of step length, which requires less computation and considerably speeds up the convergence rate of the gradient descent method. In their paper, Barzilai and Borwein [2] have proven that the rate of convergence is R-superlinear when it is a 2-dimensional problem. Also, a global convergence of the BB method has been shown in [15] for SPD linear systems, and the R-linear convergence has been shown by Dai and Liao [5]. The convergence analysis of this method is difficult and non-standard, so we will not present the proof here. It can be found in both [5] and [15], and the proof is done by contradiction. Here, we make several remarks about the convergence of the BB method. First, the non-monotonicity property of this method is observed, which can be seen in Figure 4.1, and the extent of such property is closely related to the size of the condition number of matrix A, which is explained in [6]. Also, for the convergence performance, the BB method shows significant improvement over the Landweber method and the method of steepest descent. An example from [6] can clearly show the difference: the BB method converges in around 1000 steps while the steepest descent method does not even have one significant figure improvement on the gradient norm after 2000 steps (that's when it is terminated manually). Lastly, we compare the convergence to that of the conjugate gradient method, which is discussed in the next subsection. Although it has been shown [6] that the BB method is comparable in practical efficiency to CG method, we must accept the fact that the CG method is necessarily superior in convergence rate. In conclusion, Barzilai and Borwein [2] successfully speed up the efficiency of the gradient methods. However, it leaves us to explore that if this method can be accelerated, and we review a preconditioned BB method in section 3.2.

All the gradient descent methods share something in common. That is, they do not perform well when the problem is poorly conditioned. Both the Landweber method and the steepest descent method show relatively slow convergence. Although the classic BB method gains a much faster convergence rate comparing to other gradient descent methods and is compatible with CG method in some cases, it is necessarily inferior to CG method and the property of non-monotonicity is not desired. On the other hand, bound constraints on x are easier to implement with gradient descent methods than they are with CG.

3.1.3 Conjugate Gradient Method

Conjugate gradient method is not the one we are focusing in this thesis, so we will not go into detail on the convergence analysis. Here, we simply present some important facts about the convergence. A theoretical analysis [18] has shown that, in exact arithmetic, CG is guaranteed to converge in at most n steps, where n is the size of the matrix A. However, in pratice, this is not exciting because n is usually very big.

Therefore, it is not feasible to run the CG algorithm for n iterations. On the other hand, if A has a favorable eigen-structure, then CG tends to converge much faster. For example, if A has k distinct eigenvalues, CG terminates in k steps. Moreover, if eigenvalues of A are clustered around a single point (bounded away from zero), then CG will converge more quickly. Some preconditioned CG methods have been introduced over the years, and can be found in [9] [16] and [18].

3.2 Preconditioning

As we discussed in Section 3.1.1, preconditioning is used to accelerate the rate of convergence while requiering additional work at each iteration with another matrix P. At each iteration of the preconditioned method, we solve a linear system $Pz = w$. If P is a good approximation of the ill-conditioned A, then the matrix P will also be very ill-conditioned, which will cause us to compute an inaccurate solution in the first iteration. In general, further iterations do not improve this situation and it is very difficult to recover the desired solution. Therefore, the standard approach of preconditioned method cannot be successfully implemented for ill-posed inverse problems.

Another approach to construct a slightly different matrix P is first proposed in [9]. Here, matrix P is such that it clusters large singular values at around one, but it does not alter the small singular values. We let $A = U\Sigma V^T$ be the SVD of A, and $P \approx$ SVD of A, such that $P_k = U\Sigma_k V^T$, where $\Sigma = diag(\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n)$, $\Sigma_k = diag(\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_k, 1, \cdots, 1)$, and k is the truncation index for the TSVD solution. Then the preconditioned system has the form $P^{-1}A = V\Sigma V^T$, where $\Sigma = diag(1, \cdots, 1, \sigma_{k+1}, \cdots, \sigma_n)$. We can see from the construction of Σ that the larger singular values are well separated from the small ones, and are perfectly clustered at one. It is not practical to compute exactly the SVD of A due to the size of our problem, but we can try to approximate the SVD in some applications.

Although there are many different approaches to preconditioning, we will only look at this TSVD preconditioner in this paper. In the following section, we apply this preconditioning technique to the BB method to see if it can indeed speed up the convergence rate.

Chapter 4

Experiments

In this chapter, we observe the results from three different real world examples from image deblurring. The three images used are a satellite, a grain and a Gaussian blur image from the image restoration package [3]. Numerical experiments on the convergence rate of each method discussed in this thesis is performed. We compare the convergence behavior of various methods. We observe and state different facts on the rate of the convergence, and attempts to modify the BB method are also presented. Finally, we apply our methods on the test problems in [3] to see the effectiveness.

4.1 Satellite Image

The first test problem, satellite image, was developed at the US Air Force Philips Laboratory, Lasers and Imaging Directorate, Kirtland Air Force Base, New Mexico. The image is simulated by a computer showing a satellite taken from a ground based telescope. The true and blur images are shown in the end of this section in Figure 4.8. In this section, we discuss what we observe from the implementation of various methods on the satellite image.

All the methods discussed in Chapter 2 are written in Matlab. Before we test our methods to deblur the satellite image, we compare the relative error and the iteration number to see the general convergence behavior. First, we perform an analysis on LW, SD and BB methods since they are all in the category of gradient descent method. The results are shown in Figure 4.1. It is clear from the plot that LW and SD converge very slowly compared to BB method. BB method reaches the minimum error at around 120th iteration, and this is when we want to stop the iteration for the best result. After the error reaches minimum, it starts to move up. That is, the semi-convergence property is observed for BB method. On the other hand, LW and SD methods do not show signs of semi-convergence here, and that is because they converge too slowly (much more than 200 iterations) to see this behavior. Another interesting fact is the non-monotonicity behavior of the BB method. Two big jumps are observed at around iteration 30 and 90. We will try to smooth out the undesired jumps later in this section.

From the analysis above, it is clear that BB method is preferred over LW and SD methods. Therefore, we want to compare it to the CG method since CG is the standard algorithm for solving large-scaled linear systems. Again, we plot the relative error versus iteration curve (see Figure 4.2) and make several observations. First, CG converges faster than BB, which is expected from the convergence analysis in Chapter 3. It shows that CG converges in around 55 iterations in this particular example. Comparing to 120 iterations for BB method, it is a lot faster. Also, we observe that CG shows a steeper and smoother convergence behavior. Moreover, the semi-convergence behavior of CG is well-established. Although BB doesn't converge as fast as CG, the relative errors of BB don't bounce back up as quickly as CG, which is a nice property because finding an optimal stopping iteration is difficult, and so it becomes a possibility to use BB as an alternative method to CG in some cases.

In order to make BB a comparable method to the CG, we want to get rid of all the jumps showing in Figure 4.1 and 4.2. Recall from Section 2.1 that SD and BB share

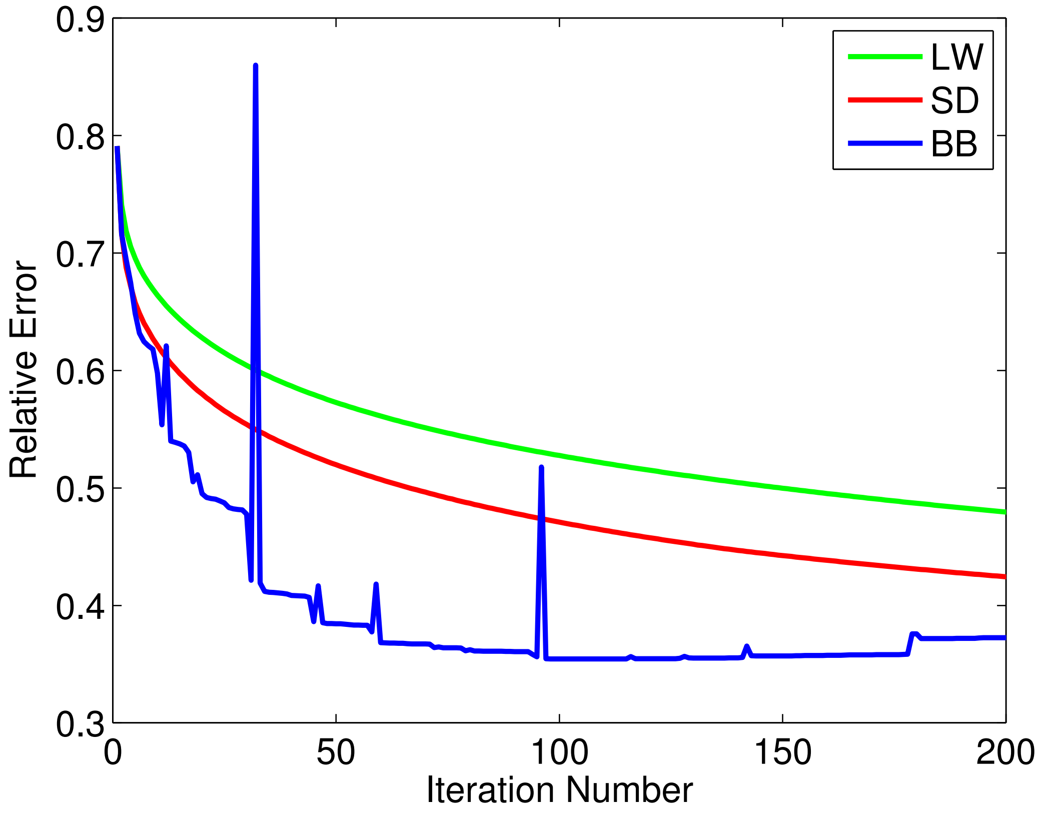


Figure 4.1: Relative error versus iteration plot for LB, SD and BB methods applied to satellite image.

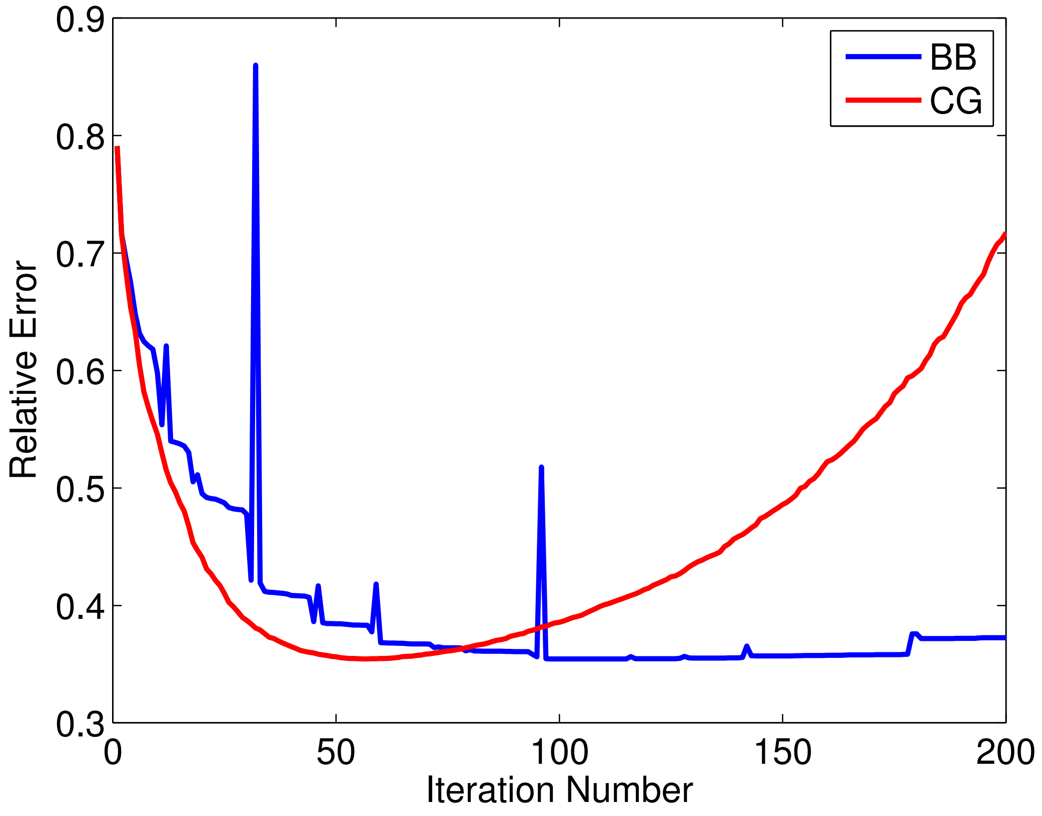


Figure 4.2: Relative error versus iteration plot for BB and CG methodsapplied to satellite image.

the same step direction but use different step length. From Figure 4.1, we can clearly see the smoothness of SD, regardless of the slow convergence. Therefore, it is natural to think replacing BB step length to SD step length when the relative error is large. That is, using the current gradient vector instead of the previous one to calculate the step length when the jump occurs. The problem is that for a realistic problem we cannot use the relative error because the true solution is not known. We wanted to see if we could find a different way to recognize when the BB step length was poor. To do this, we plot step length and relative error at each iteration on the same graph to see the relation between the two (see Figure 4.3). It shows an interesting result:

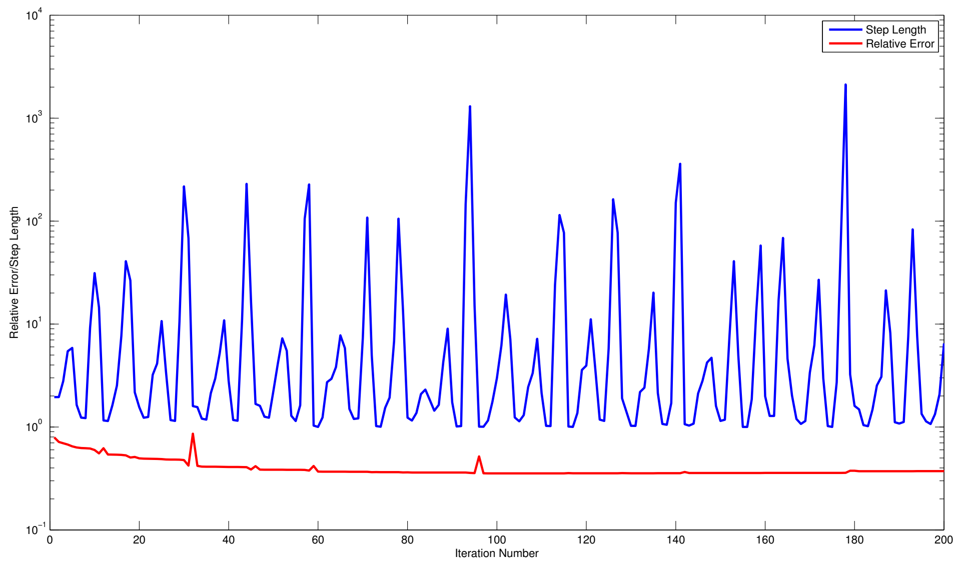


Figure 4.3: Step length and relative error at each iteration for BB method applied to satellite image.

whenever the jumps occur (large relative errors), the previous corresponding step lengths of BB method are large as well. This phenomenal fact allows us to think that changing any large values of the BB step lengths to the SD step lengths may result in a smooth BB convergence. Then we observe that the values of many step lengths are just over 1, with some extremely big values greater than 100. The following plot (see Figure 4.4) shows the comparison between the original BB method and two stabilized BB methods with maximum allowed BB step lengths equal to 2 and 5. We have also tried various values for maximum allowed BB step sizes (not shown), and it occurs that using SD step length when BB step length is greater than 2 shows the best result for this particular case.

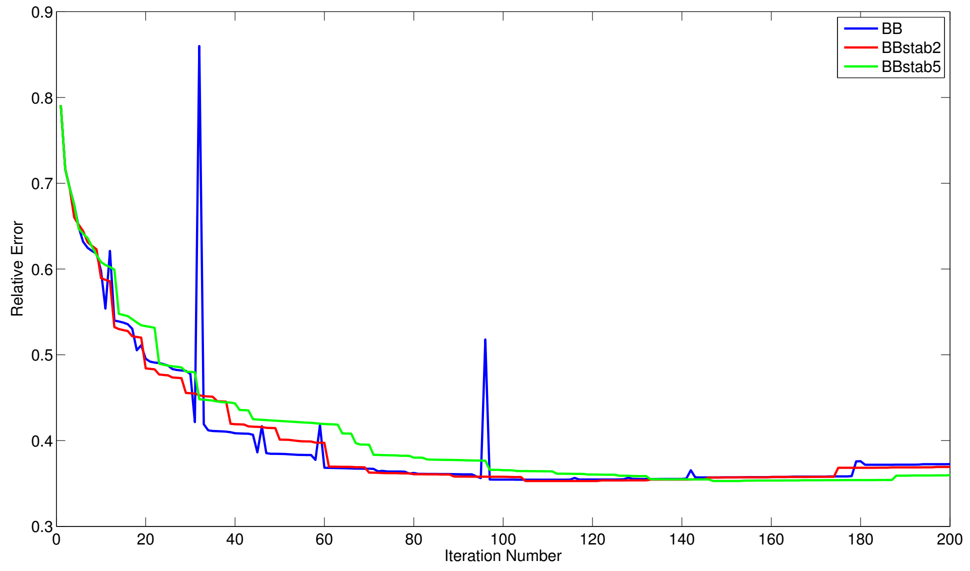


Figure 4.4: Relative error versus iteration plot for BB and two other stabilized BB methods applied to satellite image.

As shown in Figure 4.4, we can clearly see that both of the stabilized methods are much smoother than the original BB method. It even appears that the stabilized method (BBstab2) is doing better, in terms of convergence, than the original BB method at some point, such as between 20th and 30th iteration. Now, let us put BB, stabilized BB and CG together to compare their convergence behavior. The result is shown in Figure 4.5. With no surprise, CG still converges the fastest. However, for the stabilized BB method, the non-monotonicity behavior is gone. Also, the semi-

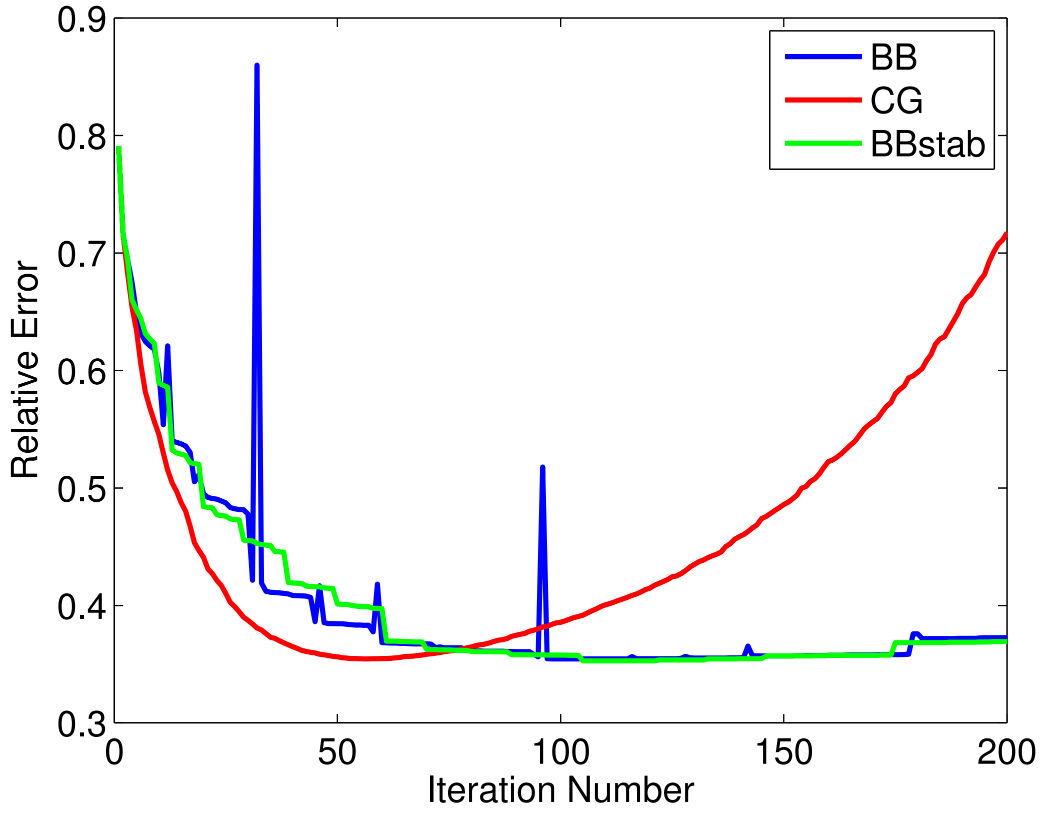


Figure 4.5: Relative error versus iteration plot for BB, stabilized BB and CG applied to satellite image.

convergence property is much better, as the relative error curve does not bounce back as quickly as it does for CG. It leaves us to think whether BB method can ever converge faster than or close to CG. Recall from Section 3.2 that we have developed a preconditioned BB method. Figure 4.6 shows that preconditioned BB method is much more efficient than the original BB method and is comparable with CG method. Although a preconditioned CG method is most likely to perform better than the original CG, we will not discuss it here because we are interested in improving BB method. PBB seems to reach the minimum relative error at around 40th iteration,

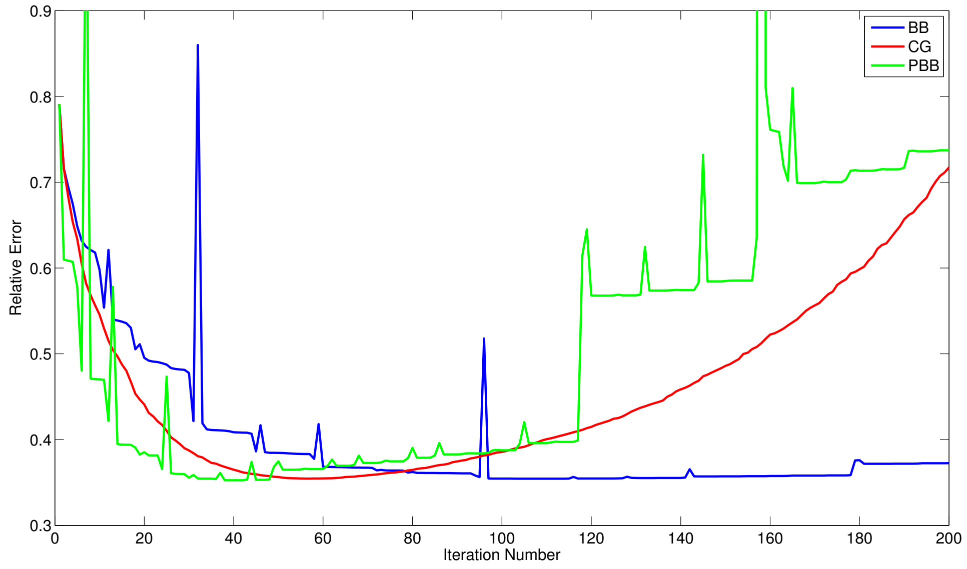


Figure 4.6: Relative error versus iteration plot for BB, preconditioned BB and CG applied to satellite image.

comparing to a much later convergence for BB, around 120th iteration. However, the non-monotonicity behavior does not disappear. Instead, it seems to have more jumps than the original BB method.

Therefore, it is natural for us to stabilize PBB, just like the way we did for BB. Then, we plot relative error versus iteration for CG, stabilized BB and stabilized

PBB. The result is shown in Figure 4.7. It is clear that both stabilized methods in-

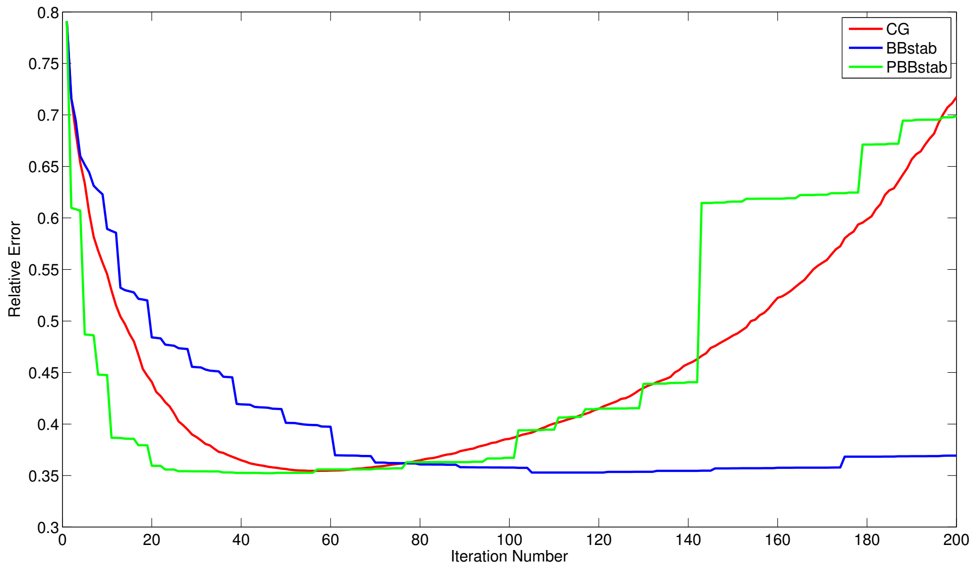
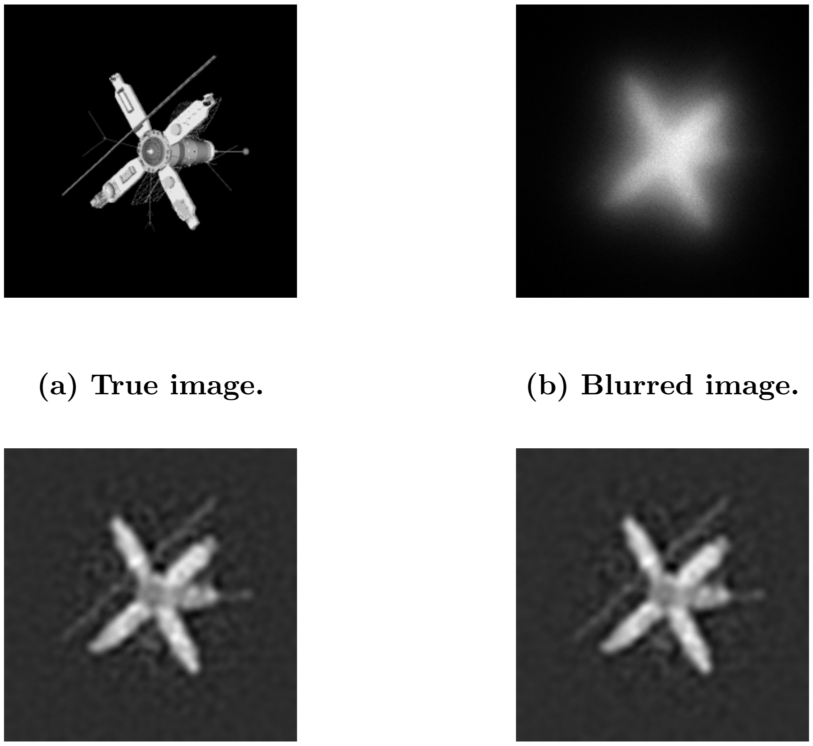


Figure 4.7: Relative error versus iteration plot for CG, stabilized BB and stabilized PBB applied to satellite image.

deed removed the undesired jumps, as shown in Figure 4.6. Moreover, the stabilized BB method does not destroy the property of fast convergence that PBB has. From PBB, we observed that the minimum error occurs at around the 40th iteration. Similarly, the stabilized PBB method converges in about the same time, around the 40th iteration. Again, the convergence behavior for stabilized PBB is better than that of that of BB method. This is what we desired.

Having developed a general idea on the convergence analysis for all the methods, we will test the effectiveness of the methods on simulated problems applying to image deblurring. The true and blurred images have 256 $\times$ 256 pixels, and are shown in the top of Figure 4.8. The bottom of Figure 4.8 compares the computed stabilized BB and CG restorations at the iteration where each method shows optimal solution (i.e., 40 for stabilized BB and 55 for CG). Visually, the two restorations are virtually indistinguishable, but semi-convergence behavior for stabilized BB is not as severe as it is for CG. Therefore, stabilized BB method might be a good alternative choice in some cases. We remark that preconditioning can be used for CG, but the semi-convergence behavior is very sensitive to preconditioning. That is, although the error may decrease more quickly to the optimal solution, further iterations may result in a rapid increase in relative error.



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

(c) Stabilized BB restoration **(d) CG restoration**

Figure 4.8: Images considered in the first example (satellite): **(a)** true image; **(b)** blurred image; **(c)** solution obtained at the 40th iteration of stabilized BB method; **(d)** solution obtained at the 55th iteration of CG method.

4.2 Grain Image

In this section, we repeat our experiment using the grain image in [3]. The true and blurred images are shown in the end of this section in Figure 4.11. First, relative error versus iteration plot is graphed for LW, SD and BB. The result is shown in Figure 4.9, and it is similar to the satellite case. LW and SD methods converge too slowly compared to BB method, and the undesired jumps still occurs for BB, such as at iteration 20. We also observed an interesting drop at around iteration 120 for SD. Then, we want to apply the stabilized BB method to this test problem to see if we

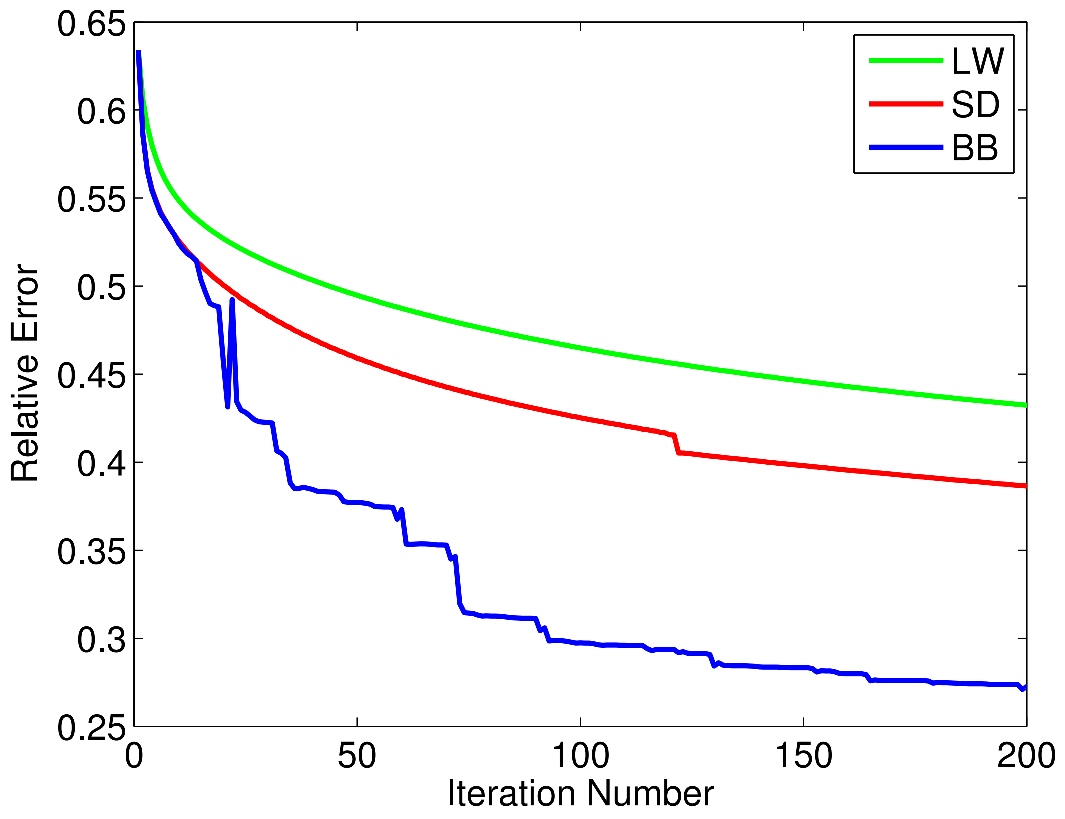


Figure 4.9: Relative error versus iteration plot for LW, SD and BB applied to grain image

can still smooth out the jumps. We compare BB, stabilized BB and CG methods to see their convergence rates. With no surprise, the non-monotonicity behavior is again removed. However, there are several interesting facts that we observed from Figure 4.10. First, BB and stabilized BB methods do not reach the minimum error as fast as it does for CG. CG converges in less than 100 iterations; BB and stabilized BB both reach the minimum errors after 150 iterations. Second, the expected semi-convergence behavior is observed for CG before 150th iteration, but the relative error starts to decrease again. This did not occur for the satellite image. The last interesting fact is that the stabilized BB method seems to perform worse than the original BB method before 130th iteration, but it looks like to reach the minimum error faster than original BB between 130th and 200th iteration. Preconditioned BB method is not studied in this example.

In the end of this section, we will again test our methods on deblurring the grain image. The true and blurred images have 256 $\times$ 256 pixels, and are shown in the top of Figure 4.11. The bottom of Figure 4.11 compares the computed stabilized BB and CG restorations at the iteration where each method reached the optimal solution (i.e., 170 for stabilized BB and 95 for CG). The stabilized BB restoration and CG restoration do not show a big difference again. However, in this example, it takes much more iterations for stabilized BB method to compute the optimal solution comparing to CG method, which only takes about half of what stabilized BB takes.

4.3 Gaussian Blur Image

In this section, we use a Gaussian blur image as our final test problem. The true and blurred images are shown in the end of this section in Figure 4.14. First, we compare the convergence behavior of LW, SD and BB. A plot of relative error versus iteration

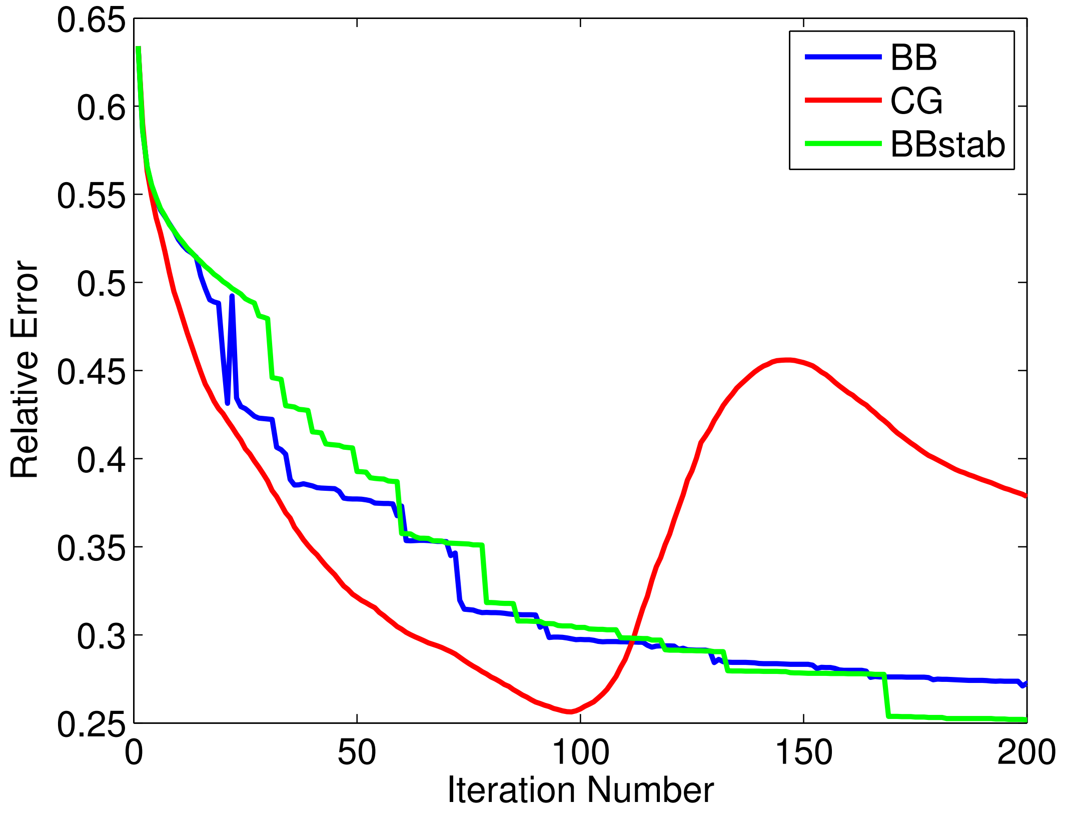
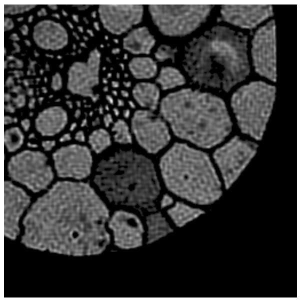
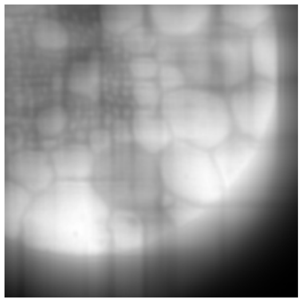
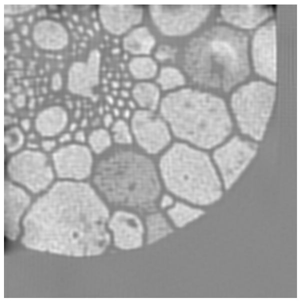


Figure 4.10: Relative error versus iteration plot for BB, stabilized BB and CG applied to grain image


(a) True image.


(b) Blurred image.


(c) Stabilized BB restoration

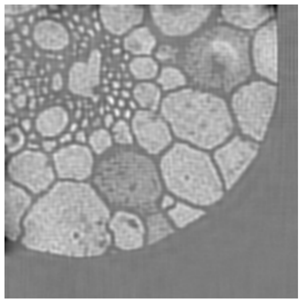

(d) CG restoration

Figure 4.11: Images considered in the second example (grain): **(a)** true image; **(b)** blurred image; **(c)** solution obtained at the 170th iteration of stabilized BB method; **(d)** solution obtained at the 95th iteration of CG method.

is shown in Figure 4.12. Even though BB still performs the best among these three methods, LW and SD are not far from BB. In other words, there is not a big difference in convergence rate between them, unlike the previous two cases. Non-monotonicity behavior still occurs, at iteration 100 for example, and semi-convergence property is also observed for BB.

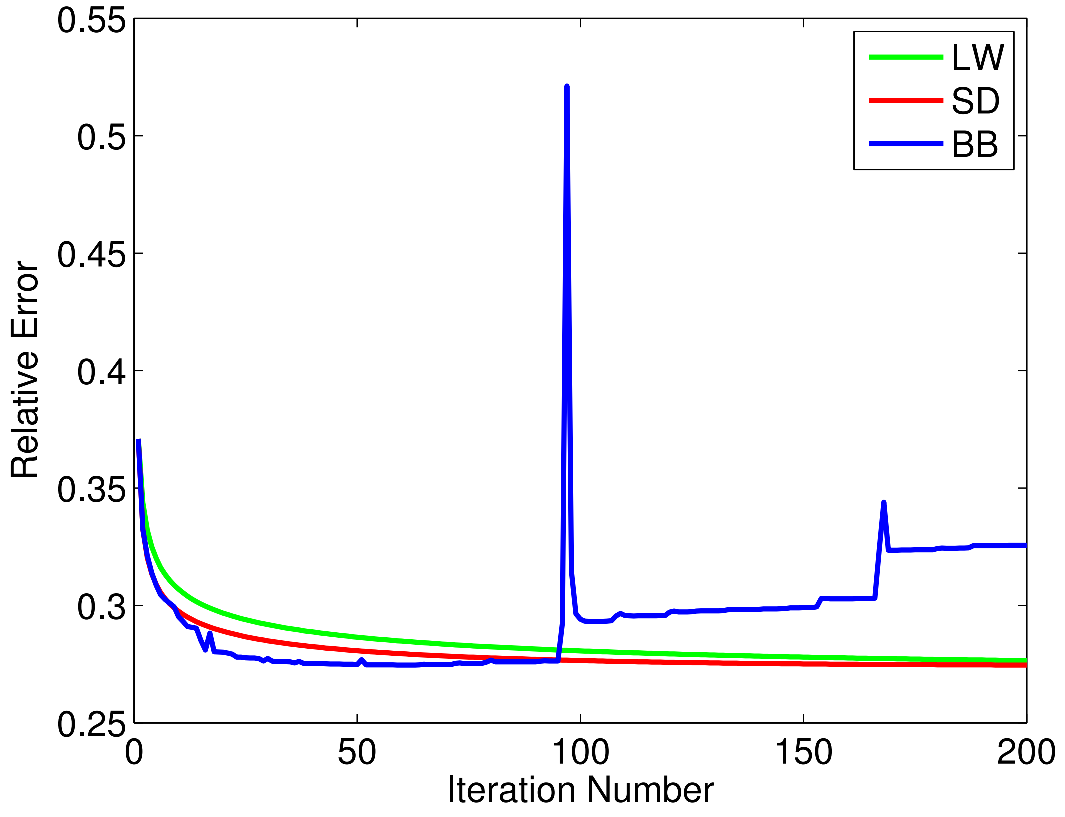


Figure 4.12: Relative error versus iteration plot for LW, SD and BB applied to Gaussian blur image

Then, we bring in CG and stabilized BB for comparison purpose. A plot of relative error versus iteration is shown in Figure 4.13. It is clear that stabilized BB method smoothed out the jumps that original BB had, without slowing down its convergence rate. Unlike the two previous cases, all three methods reach the minimum really

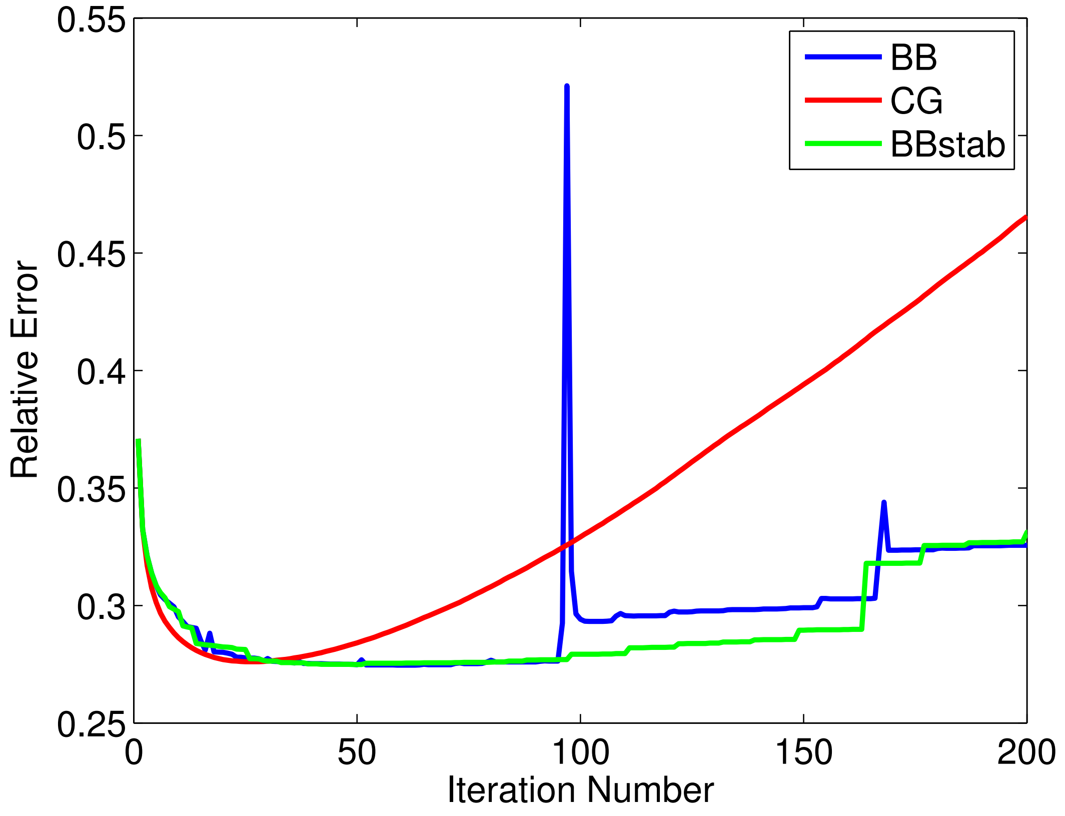
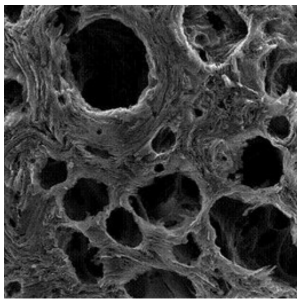


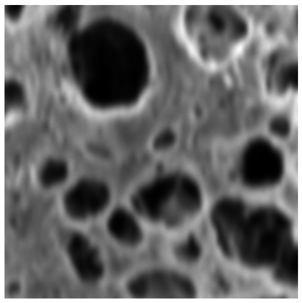
Figure 4.13: Relative error versus iteration plot for BB, stabilized BB and CG applied to Gaussian blur image

fast this time, less than 50 iterations for all of them. The expected semi-convergence behavior is observed for all the methods here, and is most obvious for CG. After CG reaches its minimum, the relative error bounces back more slowly than it reaches its minimum. However, the relative error curve continues to bounce back and it even passes the original relative error, which is about 0.4. Since stabilized BB does not bounce back that quickly, it might be a good alternative choice for solving this kind of problem. Again, preconditioned BB method is not included in this example.

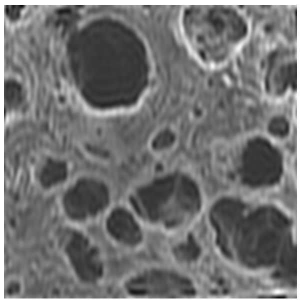
Figure 4.14 shows four different Gaussian blur images. The top two are the true and blurred image, and the left one on the bottom is the restoration from stabilized BB method while the right one is the restoration from CG. All the images again have 256×256 pixels. Note that the two restoration images show not much difference again, and both of them converge in less than 50 steps.



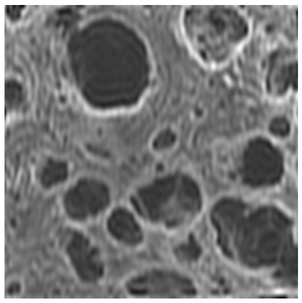
(a) True image.



(b) Blurred image.



(c) Stabilized BB restoration



(d) CG restoration

Figure 4.14: Images considered in the third example (Gaussian blur): **(a)** true image; **(b)** blurred image; **(c)** solution obtained at the 50th iteration of stabilized BB method; **(d)** solution obtained at the 30th iteration of CG method.

Chapter 5

Conclusion and Discussion

In this paper, we reviewed several iterative methods to solve large-scale linear inverse problems, such as the Landweber, steepest descent, Barzilai-Borwein and conjugate gradient method. LW and SD methods are usually not applicable for us since they always show slow convergence for such large problems. BB method, on the other hand, is comparable with the well-known CG method in some cases. However, the non-monotonicity behavior for BB method is not desired and becomes a disadvantage comparing to CG. Therefore, we explored what happened at the jumps in the relative error versus iteration plot and found out that it is closely related to the BB step lengths. Since SD method does not have such undesired property, we figured that using SD step length when BB step length is large might remove this non-monotonicity behavior. It turned out that the unfavored jumps were indeed removed, and it did not slow down the convergence process. We also applied the stabilized BB method and CG method to three examples of image deblurring. The results were consistent. That is, the stabilized BB method successfully smoothed out the original BB method for all three examples and we expect that they will work well for other problems too. It is a fact that CG reaches a minimum relative error faster than BB and stabilized BB, but the severe semi-convergence property of CG is a major drawback, as the relative error gets larger and larger after it reaches the minimum. On the other hand, stabilized BB shows a much slower bounce-back in semi-convergence, which makes it a possible alternative method of CG. The restoration images between stabilized BB and CG also confirm our suggestion, because visually we can not distinguish the difference between these two.

Future work on this project could include the following areas. First, finding the best value for the maximum allowed BB step length needs to be studied more. We picked a value that we think to work the best for our examples, however, we do not know if there is a better choice. Developing a formula to calculate the maximum allowed BB step size depending on each problem is what we want to explore next. Second, we did not compare the stabilized PBB with pre-condtioned CG method. This comparison might give us a deeper understanding on the application of the stabilized PBB method. Lastly, instead of changing large BB step lengths to SD step lengths, we can try to change to the BB step sizes to SD step sizes when the relative error is larger than the previous one. This will lead to less modification since not every large step length corresponds to a jump in relative error, which will reduce the computational cost.

Bibliography

[1] J. M. Bardsley and J. G. Nagy. Covariance-preconditioned iterative methods for non-negatively constrained astronomical imaging. *SIAM Journal on Matrix Analysis Applications*, 27:1184–1197, 2006.

[2] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

[3] S. Berisha and J. G. Nagy. Iterative methods for image restoration. In H. J. Trussell, editor, *e-Reference on Signal Processing*. Elsevier, to appear.

[4] M. Bertero and P. Boccacci. *Introduction to Inverse problems in Imaging*. Institute of Physics Publishing, Bristol, UK, 1998.

[5] Y. H. Dai and L. Z. Liao. R-linear convergence of the barzilai and borwein gradient method. *IMA Journal of Numerical Analysis*, 22:1–10, 2002.

[6] R. Fletcher. On the barzilai-borwein method. *Optimization and Control with Applications*, 96:235–256, 2005.

[7] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[8] C. W. Groetsch. *Inverse Problems.* The Mathematical Association of America, Washington, DC, 1999.

[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.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[11] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2006.

[12] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[13] J. G. Nagy and K. M. Palmer. Steepest descent, CG, and iterative regularization of ill-posed problems. *BIT*, 43:1003–1017, 2003.

[14] S. Suryanarayanan N. Raghunath, T. L. Faber, and J. R. Votaw. Motion correction of pet brain images through deconvolution: Ii. practical implementations and algorithm optimization. *Physics in Medicine and Biology*, 54.3:813–829, 2009.

[15] M. Raydan. On the barzilai and borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13:321–326, 1993.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[17] T. J. Schulz. Multiframe blind deconvolution of astronomical images. *The Journal of the Optical Society of America A*, 10:1064–1073, 1993.

[18] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

[19] C. R. Vogel. *Computational Methods for Inverse Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.