CONSTRAINED OPTIMIZATION TECHNIQUES
FOR IMAGE RECONSTRUCTION

by
Johnathan Matheas Bardsley

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in
Mathematics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2002
APPROVAL

of a dissertation submitted by

Johnathan Matheas Bardsley

This dissertation has been read by each member of the dissertation committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Curtis R. Vogel

(Signature) Date

Approved for the Department of Mathematics

Kenneth L. Bowers

(Signature) Date

Approved for the College of Graduate Studies

Bruce McLeod

(Signature) Date
STATEMENT OF PERMISSION TO USE

In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. I further agree that copying of this dissertation is allowable only for scholarly purposes, consistent with “fair use” as prescribed in the U. S. Copyright Law. Requests for extensive copying or reproduction of this dissertation should be referred to Bell & Howell Information and Learning, 300 North Zeeb Road, Ann Arbor, Michigan 48106, to whom I have granted “the exclusive right to reproduce and distribute my dissertation in and from microform along with the non-exclusive right to reproduce and distribute my abstract in any format in whole or in part.”

Signature

Date
This thesis is dedicated
to the memory of my father Scott Leavitt Bardsley, 1950-1999,
to my son Alex and daughter Ellie,
and to my wife Jennifer.
ACKNOWLEDGEMENTS

I would first like to thank Dr. Curtis R. Vogel, who has been an excellent thesis advisor in every conceivable way. Without his substantial commitment of time and energy, this thesis would not have been possible. I would also like to thank all of the other teachers, professors, and mentors who have made a difference in my life along the way.

I would like to thank my family and friends. In particular, I would like to thank my mom and dad for their endless encouragement and belief in me, my children, with whom I have found countless hours of play and unconditional love, and, most of all, my wife, whose love, support and belief in me has been a great source of strength, and whose investment of energy towards the successful completion of my PhD has been at least as great as my own.

Finally, I would like to thank James Jacklitch, who provided the LaTeX template for this thesis.
# TABLE OF CONTENTS

## LIST OF FIGURES

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vi</td>
</tr>
</tbody>
</table>

## 1. INTRODUCTION

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Dimensional Image Formation</td>
<td>1</td>
</tr>
<tr>
<td>The Test Problem with Unknown Phase</td>
<td>6</td>
</tr>
<tr>
<td>Outline of Thesis</td>
<td>8</td>
</tr>
</tbody>
</table>

## 2. MATHEMATICAL PRELIMINARIES

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Notation and Definitions</td>
<td>12</td>
</tr>
<tr>
<td>Necessary and Sufficient Conditions for a Local Minimizer</td>
<td>20</td>
</tr>
<tr>
<td>The Gradient Projection Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>Two Stage Algorithms</td>
<td>33</td>
</tr>
<tr>
<td>Unconstrained Minimization</td>
<td>36</td>
</tr>
<tr>
<td>The Conjugate Gradient Algorithm</td>
<td>37</td>
</tr>
<tr>
<td>The Newton-CG-Trust Region Method</td>
<td>40</td>
</tr>
<tr>
<td>The BFGS Method</td>
<td>44</td>
</tr>
</tbody>
</table>

## 3. LOWER BOUND CONSTRAINED QUADRATIC PROGRAMMING

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Gradient Projection-Conjugate Gradient Algorithm</td>
<td>49</td>
</tr>
<tr>
<td>The GPCG-FM Algorithm</td>
<td>54</td>
</tr>
<tr>
<td>Other Possibilities</td>
<td>59</td>
</tr>
<tr>
<td>Numerical Results</td>
<td>60</td>
</tr>
</tbody>
</table>

## 4. LOWER BOUND CONSTRAINED CONVEX PROGRAMMING

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex Gradient Projection-Conjugate Gradient</td>
<td>73</td>
</tr>
<tr>
<td>Numerical Results</td>
<td>77</td>
</tr>
</tbody>
</table>

## 5. CONCLUSION

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>88</td>
</tr>
</tbody>
</table>

## REFERENCES CITED

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>89</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A Conventional Phase Diversity Imaging Setup</td>
<td>7</td>
</tr>
<tr>
<td>2.</td>
<td>Phase and Corresponding PSF. To the left is a mesh plot of the discrete phase $\phi_{\text{true}}$. To the right is a mesh plot of the center $32 \times 32$ pixels of the corresponding PSF.</td>
<td>62</td>
</tr>
<tr>
<td>3.</td>
<td>Binary Star Object and Image Data. To the left is the upper left hand $64 \times 64$ pixels of the true image. To the right is the corresponding data with one percent noise.</td>
<td>63</td>
</tr>
<tr>
<td>4.</td>
<td>Satellite Object and Image Data. To the left is the upper left hand $64 \times 64$ pixels of the true image. To the right is the corresponding data with one percent noise.</td>
<td>63</td>
</tr>
<tr>
<td>5.</td>
<td>Binary Star Data Reconstructions. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.</td>
<td>65</td>
</tr>
<tr>
<td>6.</td>
<td>Satellite Data Reconstructions. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.</td>
<td>65</td>
</tr>
<tr>
<td>7.</td>
<td>Solution Error Plots for Binary Star Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise.</td>
<td>66</td>
</tr>
</tbody>
</table>
8. Regularized Solution Error Plots for Binary Star Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise .................................................. 67

9. Solution Error Plots for Satellite Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise .................................................. 67

10. Regularized Solution Error Plots for Satellite Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise .................................................. 68

11. Distribution of Gradient Components for Quadratic Cost Function. The horizontal axis shows the log_{10} of the size of the gradient components corresponding to the active indices. Note that the data has been binned. The vertical axis shows the number of occurrences for each size. The left hand plot corresponds to the binary star data. The right hand plot corresponds to the satellite data. The solid line corresponds to the data with one percent noise. The dashed line corresponds to the data with ten percent noise .................................................. 70

12. Binary Star Data Reconstructions with Negative Laplacian Regularization. On the left is a mesh plot of the upper left 64 x 64 pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left 64 x 64 pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images .................................................. 79

13. Satellite Data Reconstructions with Negative Laplacian Regularization. On the left is a mesh plot of the upper left 64 x 64 pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left 64 x 64 pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images .................................................. 80
14. Solution Error for Convex Cost Function with Identity Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO ........................................ 82

15. Regularized Solution Error for Convex Cost Function with Identity Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO ...................... 82

16. Solution Error for Convex Cost Function with Negative Laplacian Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO .............................. 83

17. Regularized Solution Error for Convex Cost Function with Negative Laplacian Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO .................... 83

18. Solution Error for Convex Cost Function with Identity Regularization with Satellite Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B .............................................. 84
19. Solution Error for Convex Cost Function with Identity Regularization with Satellite Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dash-dot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B .......................................................... 84

20. Distribution of Gradient Components for Convex Cost Function with Identity Regularization. The horizontal axis shows the $\log_{10}$ of the size of the gradient components corresponding to the active indices. Note that the data has been binned. The vertical axis shows the number of occurrences for each size. The left hand plot corresponds to the binary star data. The right hand plot corresponds to the satellite data. The solid line corresponds to the data with one percent noise. The dashed line corresponds to the data with ten percent noise .................................................. 86
ABSTRACT

The goal of this thesis is the solution of constrained minimization problems that originate in applications from image reconstruction. These applications include microscopy, medical imaging and, our particular application, astronomical imaging. Constraints arise from the fact that the objects we are imaging are photon densities, or light intensities, and are therefore nonnegative. These minimization problems are ill-conditioned, and, since high resolution images are desired, are large-scale. Consequently, efficient numerical techniques are required. We minimize both quadratic and strictly convex functions. Existing algorithms are implemented for the quadratic minimization problem, and ideas from these algorithms are extended to the problem of minimizing the convex function. A detailed analysis and numerical study of these algorithms is presented.
CHAPTER 1

INTRODUCTION

The goal of this thesis is the solution of constrained minimization problems that originate in applications from image reconstruction. These applications include microscopy, medical imaging and, our particular application, atmospheric imaging.

We want to find a vector $\mathbf{x}$ in a constraint set $\Omega \subset \mathbb{R}^N$ that minimizes a function $J$ mapping $\mathbb{R}^N$ to $\mathbb{R}$. We express this problem mathematically as

$$\min_{\mathbf{x} \in \Omega} J(\mathbf{x}).$$  \hspace{1cm} (1.1)

Problem 1.1 is referred to as a constrained minimization problem. We restrict our attention to a class of functions $J$ arising in two-dimensional image reconstruction.

Two Dimensional Image Formation

Our particular imaging application arises in atmospheric optics. As light propagates through the atmosphere, light rays are bent due to random variations in the index of diffraction caused by atmospheric turbulence. This causes blurring of images of an astronomical object captured with a ground-based telescope [38].

A model for data obtained from a linear, translation-invariant optical imaging system is

$$d_{i,j} = \int \int s(x_i - x, y_j - y)f_{\text{true}}(x, y)dxdy + \eta_{i,j},$$  \hspace{1cm} (1.2)
\( i = 1, \ldots, n_x, \ j = 1, \ldots, n_y \). Here \( f_{\text{true}} \) denotes the true image, or object. \( f_{\text{true}} \) is an energy density, or photon density, and hence, is nonnegative. \( s \) is called the point spread function (PSF) and is the response of the imaging system to a point light source. In our application, \( s \) quantifies the blurring effects of the atmosphere and is nonnegative. \( d \) is called the (discrete, noisy, blurred) image. \( \eta \) represents noise in the collection of the image \( d \).

For computational purposes, we discretize the integral in (1.2) to obtain

\[
d_{i,j} = \sum_{\ell} \sum_{j'} s_{i+\ell,j-j'} [f_{\text{true}}]_{\ell,j'} + \eta_{i,j}.
\]

(1.3)

With lexicographical ordering [42, p. 81] of index pairs \((i,j)\), this yields the model

\[
d = Sf_{\text{true}} + \eta,
\]

(1.4)

where \( d, f_{\text{true}}, \) and \( \eta \) are \((n_x \cdot n_y) \times 1\) vectors, and \( S \) is the \((n_x \cdot n_y) \times (n_x \cdot n_y)\) blurring matrix. Our goal is accurate, efficient estimation of \( f_{\text{true}} \).

Since \( s \) is nonnegative,

\[
Sf \geq 0 \quad \text{for all} \quad f \geq 0,
\]

(1.5)

where \( f \geq 0 \) means \( f_i \geq 0 \) for all \( i \). \( S \) is block Toeplitz with Toeplitz blocks (BTTB) [42, Chapter 5]. In most imaging applications, \( S \) is highly ill-conditioned. Furthermore, for high resolution images, \( S \) is large, e.g., a 1024 \times 1024 pixel image gives 1,048,576 unknowns, in which case \( S \) has approximately \( 10^{12} \) entries.

Due to the ill-conditioning of the matrix \( S \), the pseudo inverse [42] estimate \( S^\dagger d \) for \( f_{\text{true}} \) is unstable with respect to perturbations in the data vector \( d \). We therefore
use a stabilization technique known as Tikhonov regularization [42]. Computing the
regularized estimate for $f_{\text{true}}$ in (1.4) requires the minimization of a function of the
form

$$J_\alpha(f) = \ell(Sf; d) + \frac{\alpha}{2} f^T L f.$$  \hspace{1cm} (1.6)

We incorporate the knowledge that $f_{\text{true}}$ is nonnegative by minimizing $J_\alpha$ subject
to the constraint $f \geq 0$.

The quadratic $\frac{1}{2} f^T L f$, where $L$ is a symmetric positive semidefinite matrix, is
known as the regularization functional. In addition to having a stabilizing effect on
the minimization of (1.6), the regularization term allows for the incorporation of prior
information about the true image $f_{\text{true}}$. For example, if we know that $f_{\text{true}}$ is smooth,
taking $L$ to be the negative of the discrete Laplacian operator [42, p. 84] will penalize
vectors $f$ with large discretized gradients. Another common choice of $L$ is the identity
matrix $I$, which yields the discretized $L^2$ regularization functional $\frac{1}{2} f^T f$.

The nonnegative real number $\alpha$ is known as the regularization parameter. Typi-
cally, as $\alpha \to 0$, the minimization of $J_\alpha$ becomes increasingly ill-conditioned. On the
other hand, if $\alpha$ becomes too large, the regularization term will dominate in (1.6),
and the minimization of $J_\alpha$, though stable, will result in poor estimations of $f_{\text{true}}$. The
choice of the parameter $\alpha$ is therefore an important step in determining a suitable
$J_\alpha$. See [42, Chapter 7].

$\ell$ is called the fit-to-data function and allows for the incorporation of prior infor-
mation about the statistics of the noise vector $\eta$. In atmospheric imaging applications
[38], each component of $d$ in (1.4) is modeled as a realization of a random variable

$$d_i \sim \text{Poisson} ([Sf_{\text{true}}]_i) + \text{Normal}(0, \sigma^2). \quad (1.7)$$

By this we mean that $d_i$ is the sum of a Poisson random variable with mean and variance $[Sf_{\text{true}}]_i$ and a normally distributed, or Gaussian, random variable with mean 0 and variance $\sigma^2$ [2]. We assume that these two random variables are independent. We also assume that $d_i$ is independent of $d_j$ for all $i \neq j$. In this setting, the components of the noise vector $\eta$ in (1.4) are realizations of random variables with components

$$\eta_i = d_i - [Sf_{\text{true}}]_i.$$

We will consider two choices of the function $\ell$. The first is appropriate if the Gaussian noise term in (1.7) is dominant, and the variance $\sigma^2$ is constant with respect to $i$. In this case we take

$$\ell(Sf; d) = ||Sf - d||^2_2. \quad (1.8)$$

This is proportional to the log likelihood function [42] for the model (1.4) when the components of $d$ are independent and identically distributed Gaussian random variables with mean 0 and variance $\sigma^2$. We will refer to this choice of $\ell$ as the least squares fit-to-data function. The resulting $J_\alpha$ is quadratic.

If, on the other hand, the Poisson noise term in (1.7) dominates, we can incorporate this information by choosing $\ell$ to be the log likelihood function for the model (1.4) when the components of $d$ are independent Poisson random variables with Poisson
parameter $|Sf_{\text{true}}|_i$, which is given by

$$\ell(Sf; d) = \sum_i |Sf|_i - \sum_i d_i \log(|Sf|_i).$$

Unfortunately, this choice of $\ell$ is undefined for $f \geq 0$ such that $|Sf|_i = 0$ for some index $i$. To overcome this difficulty, we replace (1.9) by

$$\ell(Sf; d) = \sum_i (|Sf|_i + \sigma^2) - \sum_i d_i \log(|Sf|_i + \sigma^2),$$

where $\sigma^2$ is the variance of the Gaussian random variable in equation (1.7). We can statistically justify this choice of $\ell$ by noting that it is the log likelihood function for the model (1.4) when each component of $d$ is modeled as a realization of a random variable

$$d_i \sim \text{Poisson} \left(|Sf_{\text{true}}|_i + \sigma^2\right),$$

which has the same variance as the random variable $d_i$ given in equation (1.7). We replace our image vector $d$ in (1.10) by $\max(d, 0)$. This is consistent with (1.11), since any realization of a Poisson random variable is nonnegative, and has two important consequences. First, it guarantees that $\ell$, and hence, the $J_\alpha$ resulting from (1.6), is convex on $\{f \in \mathbb{R}^N \mid f \geq 0\}$. Strict convexity for $J_\alpha$ follows if the null spaces of $L$ and $S$ intersect only at $\{0\}$. Secondly, $J_\alpha$ is Lipschitz continuous on $\{f \in \mathbb{R}^N \mid f \geq 0\}$.

The minimization problems we wish to solve are then expressed as

$$\min_{f \geq 0} J_\alpha(f),$$

where $\ell$ in (1.6) is given either by (1.8) or (1.10). This problem is in the form of (1.1) with $J$ replaced by $J_\alpha$, $x$ replaced by $f$, and $\Omega = \{f \in \mathbb{R}^N \mid f \geq 0\}$. 
The Test Problem with Unknown Phase

In certain atmospheric optics applications, the PSF $s$ in (1.2) is unknown, but can be parameterized in terms of a function known as the phase, which we will denote by $\phi$. The phase, or wavefront aberration, quantifies the deviation from planarity of the wavefront of light that has propagated through the atmosphere. We assume that the dependence of the PSF $s$ on $\phi$ is given by

$$s[\phi] = |\mathcal{F}^{-1}\{pe^{i\phi}\}|^2,$$  \hspace{1cm} (1.13)

where $\mathcal{F}$ denotes the 2-D Fourier transform, $i = \sqrt{-1}$, and $p$ is the pupil, or aperture, function. See [38] for conditions which justify this assumption. We further assume

$$p(x, y) = \begin{cases} 1, & (x, y) \in A, \\ 0, & \text{otherwise}, \end{cases}$$

where the region $A$ represents the aperture and is an annulus in the case of imaging with large astronomical telescopes. Discretizing the integral in (1.2) and taking into account that corresponding to each image $d$ there is a (discretized) true phase $\phi_{\text{true}}$, we obtain the matrix-vector equation

$$d = S[\phi_{\text{true}}]f_{\text{true}} + \eta.$$  \hspace{1cm} (1.14)

This reduces to (1.4) for the case when $\phi_{\text{true}}$, and hence, $s$, is known.

When the vectors $\phi_{\text{true}}$ and $f_{\text{true}}$ are both unknown then (1.14) is underdetermined. We overcome this difficulty by applying a technique called phase diversity [22]. A
second image with a known phase perturbation $\theta$ is collected and is modeled by

$$d' = S[\phi_{\text{true}} + \theta]f_{\text{true}} + \eta'.$$

(1.15)

See Figure 1 for an illustration. Using least squares fit-to-data functions (see equation (1.8)), and quadratic regularization terms for both the object and the phase we get

$$J_{a,\gamma}(f, \phi) = \frac{1}{2}||S[\phi]f - d||^2 + \frac{1}{2}||S[\phi + \theta]f - d'||^2 + \frac{\alpha}{2}f^T Lf + \frac{\gamma}{2}\phi^T M \phi.$$  (1.16)

As in (1.6), $L$ is symmetric positive semidefinite and incorporates prior smoothness information about $f$. $M$ is also symmetric positive semidefinite and incorporates prior smoothness information about $\phi$. $\alpha$ and $\gamma$ are positive regularization parameters.

In (1.16), $\phi$ is not a constrained variable. We therefore minimize $J_{a,\gamma}$ over

$$\Omega = \{(f, \phi) \in \mathbb{R}^{2N} \mid f \geq 0\}.$$
The associated minimization problem is then expressed as

$$\min_{(\mathbf{r}, \phi) \in \Omega} J_{\alpha, \gamma}(\mathbf{r}, \phi).$$

(1.17)

This minimization problem is of the form of problem (1.1), with $J$ replaced by $J_{\alpha, \gamma}$, \( \mathbf{x} \) replaced by \((\mathbf{f}, \phi)\), and \( \mathbb{R}^N \) replaced by \( \mathbb{R}^{2N} \). If one ignores the nonnegativity constraint on the vector \( \mathbf{f} \), (1.17) becomes an unconstrained minimization problem. In [43, 19] this problem is explored.

Unfortunately, the dependence of \( S \) on \( \phi \) is nonlinear. This results in a nonconvex function \( J_{\alpha, \gamma} \). Furthermore, there are twice as many unknowns with the addition of the unknown phase. Consequently, problem (1.17), which we will not attempt to solve in this thesis, is much more difficult to solve than is (1.12), and requires more robust algorithms.

Another important consideration in constrained minimization is degeneracy. This property is discussed in Chapter 2. The solutions of the minimization problems which we will consider are not degenerate, but they are very nearly degenerate. This further increases the difficulty of solving these problems.

**Outline of Thesis**

Each of the minimization problems presented above is known as a bound constrained minimization problem. Such problems have been studied extensively and present themselves in a large number of applications [32, 23]. Various techniques have been developed for solving these problems.
The gradient projection method for minimizing a continuously differentiable function $J : \mathbb{R}^N \rightarrow \mathbb{R}$ over a closed subset $\Omega$ of $\mathbb{R}^N$ was introduced in 1964 by Goldstein [20, 21] and independently one year later by Levitin and Polyak [25]. Seminal work was done in the study of the convergence properties of this algorithm by Bertsekas [5]. This work was extended by Calamai and Moré [10]. Unfortunately, the gradient projection algorithm is very slow to converge for ill-conditioned problems.

Active set methods attempt to incorporate Hessian information via the reduced Hessian (see Chapter 2). An active set Newton algorithm for nonlinear bound constrained minimization was developed by Bertsekas [6]. Active set strategies have yielded many successful algorithms for bound constrained quadratic minimization [18]. Unfortunately, these algorithms tend not to perform well for large-scale, ill-conditioned bound constrained problems.

Several of the most effective and robust algorithms for bound constrained minimization can be viewed as two stage algorithms, which combine a constraint identification stage with a subspace minimization stage. Such algorithms include a bound constrained variant of the limited memory BFGS algorithm [32] known as LBFGSB [9, 45] and the bound constrained Newton trust region algorithm TRON [26] for general nonlinear functions, and the gradient projection conjugate gradient (GPCG) algorithm [30] and Friedlander and Martinez’s GPCG modification [14] for quadratic functions.
Contrasting the previous three approaches, interior point algorithms create a sequence of approximations to a solution of a constrained minimization problem which lie in the interior of the constraint region. Examples of interior point methods include Byrd, Hribar, and Nocedal’s KNITRO [8] and MATLAB’s large-scale bound constrained minimizer, which is based on the work of Coleman and Li [11, 12].

In Chapter 2 of this thesis we present mathematical preliminaries. These will include a general theory for lower bound constrained minimization, including the necessary and sufficient conditions for a solution to (1.1). We will also present the gradient projection algorithm of Bertsekas [5] and the corresponding convergence theory. We will present a general two stage algorithm of the form of that found in Calamai and Moré [10] and prove that convergence for this algorithm is guaranteed for a strictly convex function. We finish the chapter with a presentation of three unconstrained minimization algorithms: the conjugate gradient (CG) algorithm for quadratic minimization, and the limited memory BFGS and Newton-CG-trust region algorithms for the minimization of general nonlinear functions.

In Chapter 3 we present in detail two algorithms for lower bound constrained quadratic programming. The first is based on the work of Moré and Toreldo [30], while the second is based on the work of Friedlander and Martinez [14]. We conclude the chapter with a numerical implementation and comparison of these two methods on four different imaging test problems.
In Chapter 4 we extend the results of Chapters 2 and 3 to convex lower bound constrained minimization. We present an algorithm for the solution of this problem. Convergence follows from results is Chapter 2. We conclude the chapter with a numerical implementation and comparison of methods on several imaging test problems.
CHAPTER 2

MATHEMATICAL PRELIMINARIES

In this chapter we introduce notation and develop the mathematical tools that will be used in the sequel.

Preliminary Notation and Definitions

We will denote a vector in $\mathbb{R}^N$ by $\mathbf{x} = (x_1, \ldots, x_N)$. The $i^{th}$ standard unit vector $\mathbf{e}_i \in \mathbb{R}^N$ is defined by $[\mathbf{e}_i]_j = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and is zero otherwise. The Euclidean inner product of $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^N$ is defined by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{N} x_i y_i$, and the associated Euclidean norm of $\mathbf{x}$ is then given by $||\mathbf{x}|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

$\mathbb{R}^{M \times N}$ denotes the set of $M \times N$ matrices with real-valued entries. A matrix $A \in \mathbb{R}^{N \times N}$ is symmetric if $A = A^T$. $A$ is positive definite if for every $\mathbf{p} \neq 0$ in $\mathbb{R}^N$, $\langle A\mathbf{p}, \mathbf{p} \rangle > 0$, while $A$ is positive semidefinite if $\langle A\mathbf{p}, \mathbf{p} \rangle \geq 0$ for all $\mathbf{p}$ in $\mathbb{R}^N$. In the sequel we will use the acronym SPD for symmetric and positive definite. For $A \in \mathbb{R}^{M \times N}$, the spectral norm of $A$ is given by

$$||A|| = \sqrt{\lambda_{\text{max}}(A^T A)},$$

where $\lambda_{\text{max}}(B)$ denotes the absolute value of the largest eigenvalue of the symmetric matrix $B$. This is also known as the $\ell^2$ operator norm (see [24]).
By \( J : \mathbb{R}^N \rightarrow \mathbb{R} \) we mean that \( J \) is a real valued function of \( N \) variables, and, unless otherwise stated, we will assume that \( J \) is smooth, i.e. \( J \) is sufficiently differentiable.

We define the gradient of \( J \) at \( \mathbf{x} \) by

\[
\text{grad } J(\mathbf{x}) = \left( \frac{\partial J}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial J}{\partial x_N}(\mathbf{x}) \right)^T.
\]

The Hessian of \( J \) at \( \mathbf{x} \), denoted Hess \( J(\mathbf{x}) \), is the \( N \times N \) matrix with entries

\[
[Hess \ J(\mathbf{x})]_{ij} = \frac{\partial^2 J}{\partial x_i \partial x_j}(\mathbf{x}).
\]

**Definition 2.1.** If \( h : \mathbb{R}^N \rightarrow \mathbb{R}^M \) then \( h \) is Lipschitz continuous if there exists a positive real number \( \gamma \) such that

\[
||h(\mathbf{x}) - h(\mathbf{y})|| \leq \gamma ||\mathbf{x} - \mathbf{y}||
\]

for every \( \mathbf{x} \) and \( \mathbf{y} \) in \( \mathbb{R}^N \).

**Definition 2.2.** Let \( J : \mathbb{R}^N \rightarrow \mathbb{R} \). \( J \) is convex if

\[
J(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \leq \lambda J(\mathbf{x}) + (1 - \lambda) J(\mathbf{y})
\]

whenever \( \mathbf{x}, \mathbf{y} \in \mathbb{R}^N \) and \( \lambda \in (0, 1) \). If the inequality in (2.1) is strict whenever \( \mathbf{x} \neq \mathbf{y} \) then \( J \) is said to be strictly convex.

We will now concentrate on the problem of finding a minimizer \( \mathbf{x}^* \) of a function \( J : \mathbb{R}^N \rightarrow \mathbb{R} \) subject to the constraint that \( \mathbf{x}^* \in \Omega \subset \mathbb{R}^N \).
**Definition 2.3.** Let \( \Omega \) be a proper subset of \( \mathbb{R}^N \). \( \Omega \) will be known as the feasible set, and any \( x \in \Omega \) will be known as a feasible point.

**Definition 2.4.** \( x^* \in \Omega \) is a local constrained minimizer for \( J \) provided that there exists a \( \delta > 0 \) such that

\[
J(x^*) \leq J(x) \quad \text{whenever } x \in \Omega \text{ and } ||x - x^*|| < \delta.
\]  (2.2)

If (2.2) holds for \( \delta = \infty \) then \( x^* \) is a global constrained minimizer for \( J \). \( x^* \) is a strict local constrained minimizer for \( J \) if the inequality in (2.2) is strict whenever \( x \neq x^* \).

**Remark 2.5.** We indicate (2.2) by

\[
x^* = \arg \min_{x \in \Omega} J(x).
\]

Such an \( x^* \) is then said to be a solution to the problem

\[
\min_{x \in \Omega} J(x).
\]  (2.3)

We now introduce the notion of a convex set, and characterize smooth, convex functions defined on convex sets.

**Definition 2.6.** \( C \subset \mathbb{R}^N \) is convex if \( \lambda x + (1 - \lambda)y \in C \) whenever \( x, y \in C \) and \( 0 \leq \lambda \leq 1 \).

The following theorem can be found in Luenberger [27].

**Theorem 2.7.** Let \( J \) be a \( C^1 \) function defined on a convex set \( \Omega \). Then \( J \) is convex if and only if for all \( x, y \in \Omega \)

\[
J(y) \geq J(x) + \langle \text{grad } J(x), y - x \rangle.
\]  (2.4)
Proof. First suppose $J$ is convex. Then for all $\alpha \in [0,1]$,

$$J(\alpha y + (1 - \alpha)x) \leq \alpha J(y) + (1 - \alpha)J(x).$$

This implies that

$$J(y) \geq J(x) + \frac{J(x + \alpha(y - x)) - J(x)}{\alpha}.$$

Letting $\alpha \to 0$ yields (2.4).

Now suppose (2.4) holds. Fix $x_1$, $x_2 \in \Omega$ and $\alpha \in [0,1]$. Setting $x = \alpha x_1 + (1 - \alpha)x_2$ and alternatively $y = x_1$ and $y = x_2$, we get

$$J(x_1) \geq J(x) + \langle \text{grad } J(x), x_1 - x \rangle$$

(2.5)

$$J(x_2) \geq J(x) + \langle \text{grad } J(x), x_2 - x \rangle.$$ 

(2.6)

Multiplying (2.5) by $\alpha$ and (2.6) by $(1 - \alpha)$ and adding yields

$$\alpha J(x_1) + (1 - \alpha)J(x_2) \geq J(x) + \langle \text{grad } J(x), \alpha x_1 + (1 - \alpha)x_2 - x \rangle.$$

Substituting $x = \alpha x_1 + (1 - \alpha)x_2$ then gives

$$\alpha J(x_1) + (1 - \alpha)J(x_2) \geq J(\alpha x_1 + (1 - \alpha)x_2),$$

and hence, $J$ is convex. \qed

The notion of coercivity allows us to prove the existence of a global minimizer for a smooth function $J$. Strict convexity then guarantees uniqueness.

**Definition 2.8.** $J : \mathbb{R}^N \to \mathbb{R}$ is coercive if given any $L \in \mathbb{R}$ there exists an $M > 0$ such that if $||x|| > M$ then $J(x) > L$. 

Theorem 2.9. Let $J : \mathbb{R}^N \to \mathbb{R}$ be smooth and coercive. Then there exists a global constrained minimizer $x^*$ of $J$ on a closed set $\Omega$.

Proof. Let $x_0 \in \Omega$. By coercivity, there exists an $M > 0$ such that whenever $\|x\| > M$, $J(x) > J(x_0)$. Let $B = \{x \in \mathbb{R}^N \mid \|x\| \leq M\}$. Then $B \cap \Omega$ is nonempty and is closed and bounded, and hence compact. The continuity of $J$ therefore implies that there exists $x^* \in B \cap \Omega$ at which $J$ attains its minimum on $B \cap \Omega$. Existence follows since $\arg \min_{x \in \Omega} J(x) = \arg \min_{x \in B \cap \Omega} J(x)$.

Theorem 2.10. If $J : \mathbb{R}^N \to \mathbb{R}$ is strictly convex and a global constrained minimizer exists for $J$ on a convex set $\Omega$, it is unique.

Proof. Suppose that $x^*$ is a global constrained minimizer for $J$ on $\Omega$. If $J(\dot{x}) = J(x^*)$ with $\dot{x} \neq x^*$ then $J(\frac{1}{2}x^* + \frac{1}{2}\dot{x}) < \frac{1}{2}J(x^*) + \frac{1}{2}J(\dot{x}) = J(x^*)$, with $\frac{1}{2}x^* + \frac{1}{2}\dot{x} \in \Omega$, which is a contradiction. Hence $x^*$ is unique.

Definition 2.11. The projection of $x \in \mathbb{R}^N$ onto a closed and convex set $C$ is given by

$$P_C(x) := \arg \min_{y \in C} \|y - x\|^2.$$ 

In order to prove that $P_C$ is continuous we will need the following lemma, which will also prove useful in the sequel. We follow the exposition of Zarantonello [44].

Lemma 2.12.

$$\langle x - P_C(x), P_C(x) - y \rangle \geq 0 \quad \text{for all} \quad y \in C. \quad (2.7)$$
\textbf{Proof.} By definition,

\[ ||x - P_C(x)|| \leq ||x - y|| \quad \text{for all} \quad y \in C. \quad (2.8) \]

Consequently,

\[
0 \leq ||x - y||^2 - ||x - P_C(x)||^2
= ||(x - P_C(x)) + (P_C(x) - y)||^2 - ||x - P_C(x)||^2
= 2 \langle x - P_C(x), P_C(x) - y \rangle + ||P_C(x) - y||^2.
\]

Now replace \( y \) by \( y' = ty + (1 - t)P_C(x) \), where \( t \in [0, 1] \). Since \( y' \in C \)

\[
0 \leq 2t \langle x - P_C(x), P_C(x) - y \rangle + t^2||P_C(x) - y||^2
\]

for \( t \in [0,1] \). By first dividing by \( t \) and then letting \( t \to 0 \), (2.7) follows. \( \square \)

\textbf{Theorem 2.13.} If \( C \) is a nonempty, closed and convex set, then \( P_C : \mathbb{R}^N \to \mathbb{R}^N \) is well-defined and continuous.

\textbf{Proof.} Let \( x_0 \in \mathbb{R}^N \), and define \( J : \mathbb{R}^N \to \mathbb{R} \) by \( J(y) = ||y - x_0||^2 \). In order to prove that \( P_C \) is well-defined, it suffices to show that there exists a unique \( y^* = \arg\min_{y \in C} J(y) \).

Fix \( y_0 \in C \) not equal to \( x_0 \). Then \( \inf_{y \in C} J(y) = \inf_{y \in B} J(y) \), where \( B = C \cap \{ y \in \mathbb{R}^N \mid ||y - x_0|| < ||y_0 - x_0|| \} \). \( B \) is nonempty, closed, and bounded and is therefore compact. The continuity of \( J \) therefore implies that there exists \( y^* = \arg\min_{y \in C} J(y) \).

To prove uniqueness, suppose \( \hat{y} = \arg\min_{y \in C} J(y) \). Let \( ||x_0 - y^*|| = \eta = ||x_0 - \hat{y}|| \). By the parallelogram inequality \( ||y^* - \hat{y}||^2 = ||(y^* - x_0) - (\hat{y} - x_0)||^2 = 2||y^* - x_0||^2 + \)
2\|\hat{y} - x_0\|^2 - \|(y^* - x_0) + (\hat{y} - x_0)\|^2 = 4\eta^2 - 4\|\frac{1}{2}(y^* + \hat{y}) - x_0\|^2.\] But \(\frac{1}{2}(y^* + \hat{y}) \in C\), so \(\|\frac{1}{2}(y^* + \hat{y}) - x_0\|^2 \geq \eta^2\). Hence \(\|y^* - \hat{y}\| \leq 0\), implying \(\|y^* - \hat{y}\| = 0\), so \(y^* = \hat{y}\).

We now prove that \(P_C\) is continuous. Choose \(x\) and \(x'\) in \(\mathbb{R}^N\). Replacing \(y\) in (2.7) by \(P_C(x')\) we get
\[
\langle x - P_C(x), P_C(x) - P_C(x') \rangle \geq 0. \tag{2.9}
\]
Interchanging \(x\) and \(x'\) in inequality (2.9) gives
\[
\langle x' - P_C(x'), P_C(x') - P_C(x) \rangle \geq 0. \tag{2.10}
\]
Adding (2.9) and (2.10) gives
\[
\langle x - P_C(x) - (x' - P_C(x')), P_C(x) - P_C(x') \rangle \geq 0, \tag{2.11}
\]
or, equivalently,
\[
\|P_C(x) - P_C(x')\|^2 \leq \langle x - x', P_C(x) - P_C(x') \rangle. \tag{2.12}
\]
The Cauchy-Schwartz inequality then gives us that \(P_C\) is continuous.

We now to problem (2.3).

**Definition 2.14.** (2.3) is known as a bound constrained minimization problem if
\[
\Omega = \{x \in \mathbb{R}^N \mid L_i \leq x_i \leq U_i \text{ for each } i = 1, \ldots, N\}, \tag{2.13}
\]
where \(L_i, U_i \in \mathbb{R}\) satisfy \(-\infty \leq L_i < U_i \leq \infty\) for each \(i\).
**Definition 2.15.** (2.3) is known as a lower bound constrained minimization problem if

$$\Omega = \{ \mathbf{x} \in \mathbb{R}^N \mid L_i \leq x_i \text{ for each } i = 1, \ldots, N \}, \quad (2.14)$$

where $L_i \in \mathbb{R} \cup \{-\infty\}$.

**Remark 2.16.** The inequality $L_i \leq x_i$ in (2.14) is called the $i^{th}$ constraint.

**Remark 2.17.** The feasible set $\Omega$ defined by (2.14) or (2.13) is closed and convex, and, therefore, $\mathcal{P}_\Omega$ is well-defined and continuous.

**Remark 2.18.** In the sequel, unless otherwise stated, $\Omega$ will be defined by equation (2.14).

**Definition 2.19.** Given any feasible point $\mathbf{x}$, the $i^{th}$ constraint is active if $x_i = L_i$ and is inactive if $x_i > L_i$. The active set of indices at $\mathbf{x}$ is given by

$$\mathcal{A}(\mathbf{x}) = \{ i \mid x_i = L_i \},$$

and the inactive set by

$$\mathcal{I}(\mathbf{x}) = \{ i \mid x_i > L_i \}.$$
Theorem 2.20. (First-Order Necessary Conditions) Let $J \in C^1(\mathbb{R}^N, \mathbb{R})$. If $\mathbf{x}^*$ is a solution of (2.3) then

$$
\frac{\partial J}{\partial x_i}(\mathbf{x}^*) \geq 0 \text{ if } i \in \mathcal{A}(\mathbf{x}^*), \quad \text{and}
$$

$$
\frac{\partial J}{\partial x_i}(\mathbf{x}^*) = 0 \text{ if } i \in \mathcal{I}(\mathbf{x}^*). \tag{2.15}
$$

PROOF. First, consider $\phi \in C^1(\mathbb{R}, \mathbb{R})$. From elementary calculus, if $0 = \arg \min_{t \geq 0} \phi(t)$, then $\phi'(0) \geq 0$, while if $0 = \arg \min_{t \geq \delta} \phi(t)$ for some $\delta > 0$, then $\phi'(0) = 0$. Now define $\phi(t) = J(\mathbf{x}^* + t\mathbf{e}_i)$, where $\mathbf{e}_i$ is the $i^{th}$ standard unit vector. By the chain rule, $\phi'(0) = \langle \mathbf{e}_i, \text{grad } J(\mathbf{x}^*) \rangle = \frac{\partial J}{\partial x_i}(\mathbf{x}^*)$. If $i \in \mathcal{A}(\mathbf{x}^*)$ then $0 = \arg \min_{t \geq 0} \phi(t)$ and (2.15) holds. On the other hand, if $i \in \mathcal{I}(\mathbf{x}^*)$ then $0 = \arg \min_{t \geq \delta} \phi(t)$ for some $\delta > 0$ and (2.16) holds. \qed

The first-order necessary conditions (2.15) and (2.16) could also have been derived from the Karush-Kuhn-Tucker (KKT) conditions [32, Theorem 12.1]. The KKT conditions provide the first-order necessary conditions for a broad class of constrained optimization problems, of which (2.3) is a special case. The following Corollary gives an alternate formulation of the first-order necessary conditions.

Corollary 2.21. Let $J \in C^1(\mathbb{R}^N, \mathbb{R})$. If $\mathbf{x}^*$ is a solution of (2.3) then

$$
\langle \text{grad } J(\mathbf{x}^*), \mathbf{x}^* - \mathbf{y} \rangle \leq 0 \tag{2.17}
$$

for all $\mathbf{y} \in \Omega$. 
Theorem 2.22. Let \( J \in C^1(\mathbb{R}^N, \mathbb{R}) \) be strictly convex. Then \( x \) is a (local) solution of (2.3) if and only if \( x \) is the unique global constrained minimizer for \( J \) on \( \Omega \).

PROOF. Suppose that \( x \) is a solution of (2.3). Then, by Corollary 2.21, (2.17) holds. Theorem 2.7 then tells us that \( J(y) - J(x) \geq 0 \) for all \( y \in \Omega \). So \( x \) is a global constrained minimizer of \( J \) on \( \Omega \). Uniqueness follows from Theorem 2.10. The reverse implication is obvious. \( \square \)

Definition 2.23. If \( x \in \Omega \) satisfies equations (2.15) and (2.16) then \( x \) is a stationary point for problem (2.3). If, in addition, the inequality in (2.15) is strict for each \( i \in \mathcal{A}(x) \) then \( x \) is a nondegenerate stationary point for problem (2.3), while if \( \frac{\partial J}{\partial x_i}(x) = 0 \) for some \( i \in \mathcal{A}(x) \) then \( x \) is a degenerate stationary point for problem (2.3).

Proposition 2.24. Let \( x \in \mathbb{R}^N \). Then \( P_\Omega(x) \) has components

\[
[P_\Omega(x)]_i = \max(x_i, L_i) \text{ for } i = 1, \ldots, N.
\]

PROOF. Let \( x \in \mathbb{R}^N \) and \( \hat{y} = P_\Omega(x) \). Then \( \hat{y} = \arg \min_{y \in \Omega} J(y) \), where \( J(y) = ||y - x||^2 \).

Furthermore,

\[
\frac{\partial J}{\partial y_i}(\hat{y}) = 2(\hat{y}_i - x_i). \tag{2.18}
\]

If \( i \in \mathcal{A}(\hat{y}) \), then \( \hat{y}_i = L_i \). But by (2.15) and (2.18), \( \hat{y}_i \geq x_i \). Similarly, if \( i \in \mathcal{I}(\hat{y}) \), then \( \hat{y}_i > L_i \). But by (2.16) and (2.18), \( \hat{y}_i = x_i \). In either case, \( \hat{y}_i = \max(x_i, L_i) \). \( \square \)
Definition 2.25. The projected gradient of $J$ at $x \in \Omega$ is given by

$$[\text{grad}_P J(x)]_i = \begin{cases} 0, & i \in \mathcal{A}(x) \text{ and } \frac{\partial J}{\partial x_i}(x) \geq 0 \\ \frac{\partial J}{\partial x_i}(x), & \text{otherwise.} \end{cases} \quad (2.19)$$

We can now restate the first-order necessary conditions in terms of the projected gradient.

Theorem 2.26. $x^*$ is a stationary point for problem (2.3) if and only if

$$\text{grad}_P J(x^*) = 0.$$ 

Proof. This follows immediately from (2.15), (2.16), and (2.19). \qed

The next theorem uses the projected gradient to characterize the global minimizer of a class of convex functions, and follows immediately from Theorems 2.22 and 2.26.

Theorem 2.27. Let $J : \mathbb{R}^N \to \mathbb{R}$ be a smooth, strictly convex function. Then

$$\text{grad}_P J(x) = 0 \text{ if and only if } x \text{ is the unique global constrained minimizer of } J \text{ on } \Omega.$$ 

In order to determine second-order necessary conditions we will first consider the case in which $\mathcal{A}(x^*)$ is empty.

Lemma 2.28. Let $J \in C^2(\mathbb{R}^N)$ and suppose $x^*$ is a solution of (2.3) at which no constraints are active. Then $\text{Hess} J(x^*)$ is positive semidefinite.

Proof. Suppose $\text{Hess} J(x^*)$ is not positive semidefinite. Then there exists a $p$ with $||p|| = 1$ such that $p^T \text{Hess} J(x^*) p < 0$. Since at $x^*$ no constraints are active, there exists a neighborhood of $x^*$ which is contained inside of $\Omega$. Therefore, since $J$ is twice continuously differentiable at $x^*$, by Taylor’s Theorem $J(x^* + \alpha p) = J(x^*) +$
\[ \alpha \langle \mathbf{p}, \text{grad } J(x^*) \rangle + \frac{\alpha^2}{2} \langle \mathbf{p}, \text{Hess } J(x^*) \mathbf{p} \rangle + O(\alpha^3) \text{ as } \alpha \to 0. \] By (2.16), \( \text{grad } J(x^*) = 0. \) Thus \[ \frac{\alpha^2}{2} \langle \mathbf{p}, \text{Hess } J(x^*) \mathbf{p} \rangle + O(\alpha^3) = J(x^* + \alpha \mathbf{p}) - J(x^*). \] Since \( \langle \mathbf{p}, \text{Hess } J(x^*) \mathbf{p} \rangle < 0, \) there exists a \( \alpha_0 > 0 \) such that \( x^* + \alpha_0 \mathbf{p} \in \Omega \) and \( \frac{\alpha^2}{2} \langle \mathbf{p}, \text{Hess } J(x^*) \mathbf{p} \rangle + O(\alpha_0^3) < 0. \) Thus \( J(x^* + \alpha_0 \mathbf{p}) - J(x^*) < 0 \) and \( x^* \) is not a solution of (2.3). \( \Box \)

We now consider the case where \( \mathcal{A}(x^*) \) is not empty. Let \( x = (\xi, \eta), \) where, after possibly reordering indices, \( \xi \) corresponds to those indices that are inactive at \( x^* \) and \( \eta \) to those that are active. Let \( x^* = (\xi^*, \eta^*) \) and define \( \hat{J}(\xi) = J(\xi, \eta^*). \) Then \( \xi^* \) is an unconstrained local minimizer of \( \hat{J}, \) and hence, by Lemma 2.28, \( \text{Hess } \hat{J}(\xi^*) \) is positive semidefinite. We can make no conclusions about second-order derivative information with respect to the active indices. We therefore restrict our attention to the submatrix of \( \text{Hess } J(x^*) \) corresponding to \( \text{Hess } \hat{J}(\xi^*). \) In order to make this precise, we introduce the reduced Hessian.

**Definition 2.29.** The reduced Hessian of \( J \) is the \( N \times N \) matrix with entries

\[
\text{Hess}_R J(x)_{ij} = \begin{cases} 
\text{Hess } J(x)_{ij}, & \text{if } i \in \mathcal{I}(x) \text{ or } j \in \mathcal{I}(x) \\
\delta_{ij}, & \text{otherwise.}
\end{cases}
\] (2.20)

We can now state the second order necessary conditions for a solution to problem (2.3).

**Theorem 2.30.** (Second-Order Necessary Conditions) Let \( J \in C^2(\mathbb{R}^N) \) and suppose \( x^* \) is a solution of problem (2.3). Then \( \text{Hess}_R J(x) \) is positive semidefinite.

**Proof.** If \( x^* \) has no active indices, \( \text{Hess}_R J(x^*) = \text{Hess } J(x^*) \) is positive semidefinite by Lemma 2.28. Otherwise, \( x^* \) has \( M < N \) active indices. Define \( \hat{J} \) as in the
discussion preceding Definition 2.29. Then Hess $\tilde{J}(x^*)$ is a positive semidefinite $(N - M) \times (N - M)$ matrix. Consequently, the $N \times N$ matrix

$$
\begin{bmatrix}
\text{Hess} \tilde{J}(x^*) & 0 \\
0 & I_{M \times M}
\end{bmatrix}
$$

is positive semidefinite. Furthermore, it is equivalent to Hess$_R J(x)$ with respect to a reordering of indices, which implies that Hess$_R J(x)$ is also positive semidefinite.

As in the case of the first-order necessary conditions, the second-order necessary conditions could also have been derived in terms of the KKT conditions [32, Theorem 12.5]. The same can be said for the second-order sufficient conditions [32, Theorem 12.6], which we will present next.

**Theorem 2.31. (Second-Order Sufficient Conditions)** Let $x^*$ be a nondegenerate stationary point for problem (2.3), and suppose that Hess$_R J(x^*)$ is positive definite. Then $x^*$ is a strict local constrained minimizer for $J$ on $\Omega$.

**Proof.** Choose any nonzero $p \in \mathbb{R}^N$ such that $x^* + tp \in \Omega$ for $t$ small and positive. Then $p_i \geq 0$ whenever $i \in A(x^*)$. Define $\phi(t) = J(x^* + tp)$ for all $t$ such that $x^* + tp \in \Omega$. It suffices to show that $\phi$ has a local minimum at $t = 0$. By Taylor’s theorem, $\phi(t) = J(x^*) + tp^T \text{grad} J(x^*) + \frac{t^2}{2} p^T \text{Hess} J(x^*) p + O(t^3)$ for small $t$. First, suppose there exists $\delta > 0$ such that $x^* + tp \in \Omega$ for all $t \in (-\delta, \delta)$. Then $p_i = 0$ whenever $i \in A(x^*)$. On the other hand, since $x^*$ is a stationary point of (2.3), whenever $i \in I(x^*) = \frac{\partial J}{\partial x_i}(x^*) = 0$. Hence $\phi'(0) = p^T \text{grad} J(x^*) = 0$. Furthermore, $\phi''(0) = p^T \text{Hess} J(x^*) p = p^T \text{Hess}_R J(x^*) p > 0$, since Hess$_R J(x^*)$ is positive definite. Thus $\phi$ has a local minimum at $t = 0$. In the other case, $p_i > 0$ for some $i \in A(x^*)$, }
and, since $x^*$ is a nondegenerate stationary point of (2.3), $\frac{\partial J}{\partial x_i}(x^*) > 0$ whenever $i \in A(x^*)$, while $\frac{\partial J}{\partial x_i}(x^*) = 0$ whenever $i \in I(x^*)$. Hence $\phi'(0) = p^T \text{grad } J(x^*) = \sum_{i \in A(x^*)} p_i \frac{\partial J}{\partial x_i}(x^*) > 0$. Since $x + tp \notin \Omega$ for $t < 0$, this implies that $\phi$ has a local minimum at $t = 0$.

The second-order sufficient conditions are not necessary. To see this, consider $J(x) = x^4$ on $\Omega = [-1, \infty)$. Then $x^* = 0$ is a global minimizer of $J$ on $\Omega$. The second-order necessary conditions for $J$ are satisfied at $x^*$, while, since $\text{Hess } J(x) = 12x^2$ is zero at $x^*$, the second-order sufficient conditions are not satisfied.

The Gradient Projection Algorithm

The gradient projection algorithm can be viewed as a generalization of the method of steepest descent [32] for unconstrained optimization. In this section we follow the exposition of Bertsekas [?]. The gradient projection algorithm generates a sequence $\{x_k\}$ as follows:

**Algorithm 1.** (Gradient Projection Algorithm)

1. Select initial guess $x_0$, and set $k = 0$.

2. Compute $p_k = -\text{grad } J(x_k)$.

3. Compute $\lambda_k = \arg \min_{\lambda > 0} J(\mathcal{P}_\Omega(x_k + \lambda p_k))$.

4. Set $x_{k+1} = \mathcal{P}_\Omega(x_k + \lambda_k p_k)$.

5. If termination criteria are met, STOP.

6. Otherwise, update $k := k + 1$ and return to (1).
Remark 2.32. Motivated by Theorem 2.26, our objective is to create a sequence \( \{x_k\} \) such that \( \nabla \phi \rightarrow 0 \). We can take our termination criterion in Step (4) of Algorithm 1 to be

\[
\|\nabla \phi (x_k)\| < \tau,
\]

where \( \tau > 0 \) is a given stopping tolerance.

Remark 2.33. Step (2) above is known as a projected line search. Typically it cannot be solved exactly, but, as we will see, this is not necessary. We begin by defining the function \( \phi_k : \mathbb{R} \rightarrow \mathbb{R} \) by

\[
\phi_k(\lambda) = J(x_k(\lambda)),
\]

(2.21)

where

\[
x_k(\lambda) := \mathcal{P}_\Omega(x_k + \lambda p_k)
\]

and \( p_k = -\nabla J(x_k) \). The goal of our algorithm is, given an approximate solution \( x_k \) of problem (2.3) and a search direction \( p_k \), to generate a \( \lambda_k \) which satisfies the sufficient decrease condition

\[
\phi_k(\lambda_k) \leq \phi_k(0) - \frac{\mu}{\lambda_k} \|x_k - x_k(\lambda_k)\|^2,
\]

(2.22)

where \( \mu \in (0, \frac{1}{2}) \). Typically, \( \mu \) is set to \( 10^{-4} \) [23].

For the remainder of this chapter, we will assume that the Hess \( J(x) \) is positive definite for every \( x \in \Omega \), and that there exists \( \lambda_{\text{min}} \) and \( \lambda_{\text{max}} \) such that

\[
0 < \lambda_{\text{min}} \leq \lambda(x) \leq \lambda_{\text{max}}
\]

(2.23)
for every eigenvalue $\lambda(x)$ of Hess $J(x)$ and for every $x \in \Omega$.

We now present an iterative (approximate) line search algorithm that will be used in the sequel. It is based on the line search algorithm found in Moré and Toraldo [30]. First, we choose our initial approximation $\lambda^0_k$ of $\lambda_k$ to be the minimizer of the quadratic Taylor approximation of $\phi_k$,

$$q_k(\lambda) = \phi_k(0) + \lambda \phi_k'(0) + \frac{\lambda^2}{2} \phi_k''(0).$$

In other words, take

$$\lambda^0_k = \frac{-\langle \text{grad} J(x_k), \text{grad}_p J(x_k) \rangle}{\langle \text{Hess} J(x_k) \text{grad}_p J(x_k), \text{grad}_p J(x_k) \rangle}, \quad (2.24)$$

Then, given a $\lambda^i_k$ that does not satisfy (2.22), we compute $\lambda^{i+1}_k$ by first defining $\hat{\lambda}^{i+1}_k$ to be the minimizer of the quadratic function $q_k$ that satisfies $q_k(0) = \phi_k(0)$, $q_k'(0) = \phi_k'(0)$, and $q_k(\lambda^i_k) = \phi_k(\lambda^i_k)$. Then

$$\lambda^{i+1}_k \triangleq \text{median} \left[ \frac{1}{100} \lambda^i_k, \hat{\lambda}^{i+1}_k, \frac{1}{2} \lambda^i_k \right]. \quad (2.25)$$

The line search algorithm is then given follows:

**Algorithm 2.** (Line Search Algorithm)

1. Compute $\lambda^0_k$ as in (2.4), and set $i = 0$.
2. If $\lambda^i_k$ satisfies (2.22), set $\lambda_k = \lambda^i_k$ and STOP.
3. Otherwise, compute $\lambda^{i+1}_k$ by equation (2.25).
4. Update $i := i + 1$ and return to (1).
A natural question arises. Will the above algorithm identify a suitable \( \lambda_k \) in a finite number of iterations? The answer is yes, but to prove this fact we will first prove two lemmas.

**Lemma 2.34.** Define \( x(\lambda) = P_\Omega(x - \lambda \text{grad } J(x)) \). Then for every \( x, y \in \Omega \) and \( \lambda \geq 0 \) we have

\[
\lambda \langle \text{grad } J(x), y - x(\lambda) \rangle \geq \langle x - x(\lambda), y - x(\lambda) \rangle.
\]  

(2.26)

**Proof.** By Lemma 2.12, with \( x \) replaced by \( x - \lambda \text{grad } J(x) \) and \( C = \Omega \),

\[
\langle x - \lambda \text{grad } J(x) - x(\lambda), y - x(\lambda) \rangle \leq 0,
\]

for all \( x \) and \( y \) in \( \Omega \), which is equivalent to (2.26). \( \square \)

**Lemma 2.35.** Assume that \( \text{grad } J \) is Lipschitz continuous with Lipschitz constant \( L \). Then, for any \( x \in \Omega \), the sufficient decrease condition (2.22) holds for all \( \lambda \) such that

\[
0 \leq \lambda \leq \frac{2(1 - \mu)}{L}.
\]  

(2.27)
Proof. Let $x(\lambda)$ be as in the last lemma, and let $\phi(\lambda)$ be defined by equation (2.21).

Then, for any $\lambda \geq 0$ and $x \in \Omega$, since $x(\lambda)$ is continuous and piecewise differentiable,

$$
\phi(1) - \phi(0) = J(x(\lambda)) - J(x)
$$

$$
= \langle \text{grad } J(x), x(\lambda) - x \rangle - \int_0^1 \langle \text{grad } J[x - t(x - x(\lambda))] - \text{grad } J(x), x(\lambda) - x \rangle dt
$$

$$
\leq \frac{1}{\lambda} ||x - x(\lambda)||^2 + \int_0^1 L ||t(x - x(\lambda))|| \cdot ||x - x(\lambda)|| dt
$$

$$
= \left( \frac{1}{\lambda} + \frac{L}{2} \right) ||x - x(\lambda)||^2
$$

$$
= \left\{ \frac{\lambda L - 2}{2} \right\} \frac{1}{\lambda} ||x - x(\lambda)||^2.
$$

Inequality (2.22) then follows immediately from (2.27). \hfill \Box

We can now prove the theorem that tells us our line search algorithm will eventually generate an acceptable step length $\lambda_k$.

**Theorem 2.36.** Suppose that $\text{grad } J$ is Lipschitz continuous. Then, in Algorithm 2, there exists an $N$ such that $\lambda_k^N$ satisfies the sufficient decrease condition (2.22).

Proof. Let $L$ be the Lipschitz constant of $\text{grad } J$. There exists an $N$ such that $\lambda_k^0/2^N < \frac{2(1-\mu)}{L}$. Furthermore, (2) of Algorithm 2 tells us that $\lambda_k^N \leq \frac{\lambda_k^0}{2^N}$. The result then follows from Lemma 2.35. \hfill \Box

We can now prove the main result of this section, which gives us a necessary condition for the convergence of Algorithm 1. But first, we will need to prove a lemma.
Lemma 2.37. Let $\lambda_k = \lambda^N_k$ be the output from Algorithm 2 at iteration $k$ of Algorithm 1. Then
\[
\min \left\{ \frac{1}{\lambda_{\max}}, (1 - \mu)/(50 \cdot L) \right\} \leq \lambda_k \leq \frac{1}{\lambda_{\min}} \tag{2.28}
\]
for every $k$.

PROOF. Equation (2.24) can be written
\[
\lambda_k^0 = \frac{1}{\langle \text{Hess} J(x_k) d_k, d_k \rangle},
\]
where $d_k = \text{grad}_P J(x_k)/||\text{grad}_P J(x_k)||$. Then
\[
\frac{1}{\lambda_{\max}} \leq \lambda_k^0 \leq \frac{1}{\lambda_{\min}} \tag{2.29}
\]
which gives us the upper bound in (2.28). The lower bound in (2.29) together with Lemma 2.35 and (2) of Algorithm 2 gives us the lower bound in (2.28).

\[ \square \]

Theorem 2.38. Assume that $J$ is Lipschitz continuous with Lipschitz constant $L$. Let $\{x_k\}$ be a sequence generated by the gradient projection algorithm using an inexact line search satisfying (2.22). Then every limit point of $\{x_k\}$ is a stationary point for problem (2.3). In particular, if $x_k \to \hat{x}$ then $\hat{x}$ is a stationary point of (2.3).

PROOF. Since our line search algorithm is guaranteed to generate an acceptable $\lambda_k$, Algorithm 1 is well defined. If $\{x_k\}$ generated by Algorithm 1 stops at a stationary point we are done. Otherwise, suppose $\{x_k\}_{k \in K}$ is a subsequence which converges to a vector $\bar{x}$. Since $\{J(x_k)\}$ is monotone decreasing, $J(x_k) \to J(\bar{x})$. By (2.22), $||x_k - x_{k+1}||^2 \leq \frac{\lambda_k}{\mu} |J(x_k) - J(x_{k+1})|$. By Lemma 2.37, the $\lambda_k$’s are bounded above.
Hence,
\[ ||x_k - x_{k+1}|| \to 0. \] (2.30)

Now, using Lemma 2.34, for any \( k \in \mathcal{K} \) and \( y \in \Omega \),
\[
\langle \nabla J(x_k), x_k - y \rangle = \langle \nabla J(x_k), x_{k+1} - y \rangle + \langle \nabla J(x_k), x_k - x_{k+1} \rangle \\
\leq \frac{1}{\lambda_k} \langle x_k - x_{k+1}, x_{k+1} - y \rangle + \langle \nabla J(x_k), x_k - x_{k+1} \rangle \\
\leq ||x_k - x_{k+1}|| \cdot \left| \begin{array}{c} \frac{x_{k+1} - y}{\lambda_k} + \nabla J(x_k) \end{array} \right|. 
\]

Lemma 2.37 tells us that the \( \lambda_k \)'s are bounded below. Also, \( \nabla J(x_k) \) is bounded for \( k \in \mathcal{K} \). Hence, taking limits as \( k \to \infty \) and using (2.30), we have \( \langle \nabla J(\bar{x}), \bar{x} - y \rangle \leq 0 \), for all \( y \in \Omega \). Thus \( \bar{x} \) is a stationary point (see Corollary 2.21). \( \square \)

**Remark 2.39.** Equation (2.26) suggests an alternate choice of sufficient decrease condition. Replacing (2.22) by
\[
\phi_k(\lambda_k) \leq \phi_k(0) - \mu \langle \nabla J(x_k), x_k - x_k(\lambda_k) \rangle 
\] (2.31)
results in a stronger condition, since if \( \lambda_k \) satisfies (2.31) then by (2.26) it also satisfies (2.22). Given our assumption regarding the eigenvalues of Hess \( J(x) \) found in Remark 2.33, Theorem 2.38 holds with the sufficient decrease condition (2.22) replaced by (2.31) as long as the \( \lambda_k \)'s are bounded away from zero, or a certain curvature condition is satisfied for each \( \lambda_k \) (see [10]).

The gradient projection algorithm also has a very powerful identification property, which is given in the corollary following the next theorem.
Theorem 2.40. Let \( J \in C^1(\mathbb{R}^N, \mathbb{R}) \), and let \( \{ \mathbf{x}_k \} \) be an arbitrary sequence in \( \Omega \) which converges to a nondegenerate stationary point \( \mathbf{x}^* \) for problem (2.3). If \( \nabla J(\mathbf{x}_k) \rightarrow 0 \) then \( \mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}^*) \) for all \( k \) sufficiently large. The active set \( \mathcal{A} \) is defined in Definition 2.19.

Proof. We begin by proving that \( \mathcal{A}(\mathbf{x}_k) \subseteq \mathcal{A}(\mathbf{x}^*) \) for \( k \) large enough. Let \( \delta = \min_{i \in I(\mathbf{x}^*)} |L_i - x_i^*| \). Then for every \( \mathbf{x} \in \Omega \) such that \( ||\mathbf{x} - \mathbf{x}^*|| < \delta \), \( \mathcal{A}(\mathbf{x}) \subseteq \mathcal{A}(\mathbf{x}^*) \).

There exists \( N_1 \) such that for all \( k \geq N_1 \), \( ||\mathbf{x}_k - \mathbf{x}^*|| < \delta \). Hence, for all \( k \geq N_1 \), \( \mathcal{A}(\mathbf{x}_k) \subseteq \mathcal{A}(\mathbf{x}^*) \).

To show that \( \mathcal{A}(\mathbf{x}^*) \subseteq \mathcal{A}(\mathbf{x}_k) \) for \( k \) large enough, we begin by choosing \( i \in \mathcal{A}(\mathbf{x}^*) \).

Since \( \mathbf{x}^* \) is nondegenerate, \( \frac{\partial J}{\partial x_i}(\mathbf{x}^*) > 0 \). On the other hand, by Cauchy Schwartz, \( \langle \nabla J(\mathbf{x}_k), e_i \rangle \leq ||\nabla J(\mathbf{x}_k)|| \rightarrow 0 \). Thus there exists an \( N_{2,i} \) such that for all \( k \geq N_{2,i} \), \( \frac{\partial J}{\partial x_i}(\mathbf{x}_k) > \frac{1}{2} \frac{\partial J}{\partial x_i}(\mathbf{x}^*) > 0 \) and \( ||\nabla J(\mathbf{x}_k)|| < \frac{1}{2} \frac{\partial J}{\partial x_i}(\mathbf{x}^*) \).

It follows immediately that \( [\nabla J(\mathbf{x}_k)]_{i} = 0 \) for all \( k \geq N_{2,i} \). Hence, \( i \in \mathcal{A}(\mathbf{x}_k) \) for all \( k \geq N_{2,i} \) (see Definition 2.25). Doing this for each \( i \in \mathcal{A}(\mathbf{x}^*) \) and setting \( N_2 = \max_{i \in \mathcal{A}(\mathbf{x}^*)} \{ N_{2,i} \} \), we get that for all \( k \geq N_2 \), \( \mathcal{A}(\mathbf{x}^*) \subseteq \mathcal{A}(\mathbf{x}_k) \).

Finally, let \( N = \max\{N_1, N_2\} \). Then for all \( k \geq N \), \( \mathcal{A}(\mathbf{x}_k) = \mathcal{A}(\mathbf{x}^*) \), which completes the proof.

The previous theorem does not specify how \( \{ \mathbf{x}_k \} \) is generated. The identification property of the gradient projection algorithm given in the following corollary is therefore independent of which sufficient decrease condition, (2.22) or (2.31), is used.
**Corollary 2.41.** Let $J \in C^1(\mathbb{R}^N, \mathbb{R})$ and suppose that the iterates $\{x_k\}$ generated by the gradient projection algorithm using an inexact line search satisfying (2.22) or (2.31) converge to a nondegenerate local minimizer $x^*$. Then $A(x_k) = A(x^*)$ for all $k$ sufficiently large.

**Proof.** The continuity of $\text{grad} J$ and $P_\Omega$ imply that $\text{grad}_P J$ is continuous (see Definition 2.25). Thus $\text{grad}_P J(x_k) \to \text{grad}_P J(x^*) = 0$. The result then follows from the previous theorem. \hfill $\square$

**Two Stage Algorithms**

A natural question arises following the theoretical results in the previous section: How does this algorithm perform on real problems? This question is easily answered by an appeal to Corollary 2.41. In finitely many iterations the gradient projection algorithm identifies the optimal face $\mathcal{F}(x^*)$. Once this occurs the algorithm reduces to the method of steepest descent, which can be a very slowly converging algorithm. On the other hand, the low cost per iteration and the identification properties of the gradient projection algorithm are desirable. This gives some motivation for the two stage algorithm introduced in [10], which combines a first stage in which the gradient projection algorithm is used to identify a face $\mathcal{F}$, with a second stage in which a fast converging unconstrained algorithm is used together with a projected line search to explore $\mathcal{F}$ further. We give the outline in the following algorithm.

**Algorithm 3.** (Two Stage Algorithm)
Stage 0: Select $x_0$, and set $k = 0$.

Stage 1: Compute $x_{k+1}^{GP}$ with initial guess $x_k$ using at least one gradient projection iteration with inexact line search Algorithm 2.

Stage 2: Compute $x_{k+1} \in \Omega$ such that $J(x_{k+1}) \leq J(x_{k+1}^{GP})$.

Update $k := k + 1$ and return to Stage 1.

Remark 2.42. If at each iteration in Stage 2 we choose $x_{k+1} = x_{k+1}^{GP}$ then Algorithm 3 is Algorithm 1. Hence, given the convergence theory that was developed above for the gradient projection algorithm, it is not surprising that the convergence result analogous to Theorem 2.38 can be proven for Algorithm 3. If $J$ is convex and coercive, we can prove an even stronger result.

Theorem 2.43. Assume that $J$ is Lipschitz continuous. Let $\{x_k\}$ be a sequence generated by Algorithm 3. Then every limit point of $\{x_k\}$ is a stationary point for problem (2.3). In particular, if $x_k \to \hat{x}$ then $\hat{x}$ is a stationary point of (2.3).

Proof. In the proof of Theorem 2.38, replace $x_{k+1}$ by the first gradient projection iterate computed in Stage 1 and $\lambda_k$ by the corresponding line search parameter value. The remainder of the proof goes through unchanged.

Corollary 2.44. Let $J : \mathbb{R}^N \to \mathbb{R}$ be a smooth, strictly convex, coercive function that is Lipschitz continuous. If $\{x_k\}$ is the sequence generated by Algorithm 3, then $\{x_k\}$ converges to the unique global constrained minimizer $x^*$. 

Proof. First, by Theorems 2.9 and 2.10, the global constrained minimizer $x^*$ exists and is unique. Assume that we stop Algorithm 3 if it identifies $x^*$, i.e., if $\nabla_p J(x_k) = 0$ for some $k$. Then, if the sequence $\{x_k\}$ generated by Algorithm 3 is finite, it must terminate at $x^*$. Otherwise, Algorithm 3 generates an infinite sequence $\{x_k\}$. Since $J$ is coercive and $\{J(x_k)\}$ is a nonincreasing sequence, $\{x_k\}$ is bounded. Hence, it has a convergent subsequence, which, by Theorems 2.43 and 2.22, must converge to $x^*$, and thus $\{J(x_k)\}$ converges to $J(x^*)$. Now, by Taylor's Theorem [32, Theorem 2.1], given $p \in \mathbb{R}^N$,

$$J(x^* + p) = J(x^*) + \langle \nabla J(x^*), p \rangle + \frac{1}{2} \langle \text{Hess} \ J(x^* + tp)p, p \rangle,$$

for some $t \in (0, 1)$. Letting $p = x_k - x^*$ yields

$$J(x_k) - J(x^*) = \langle \nabla J(x^*), x_k - x^* \rangle + \frac{1}{2} \langle \text{Hess} \ J(x^* + t(x_k - x^*))(x_k - x^*), x_k - x^* \rangle$$

$$\geq \frac{1}{2} \langle \text{Hess} \ J(x^* + t(x_k - x^*))(x_k - x^*), x_k - x^* \rangle$$

$$\geq \frac{\lambda_{\text{min}}}{2} \|x_k - x^*\|^2,$$

where $\lambda_{\text{min}}$ is defined by equation (2.23). The first inequality follows from Corollary 2.21. Since $\{J(x_k)\}$ converges to $J(x^*)$, the above inequality shows that $\{x_k\}$ converges to $x^*$.

Remark 2.45. As in the case of the gradient projection algorithm, we can replace the sufficient decrease condition (2.22) by (2.31), and Theorem 2.43 and will hold as long as the $\lambda_k$'s are either bounded away from zero, or a curvature condition is
satisfied for each $\lambda_k$. This fact is proven in [10]. The result of Corollary 2.44 will also
hold, as its proof does not depend upon which sufficient decrease condition is used.

Algorithm 3 also has the same identification properties as does the gradient projection
algorithm, regardless of whether (2.22) or (2.31) is used as the sufficient decrease
condition. The proof of the following theorem is the same as that of Corollary 2.41.

**Theorem 2.46.** Let $J \in C^2(\mathbb{R}^N, \mathbb{R})$ and suppose that the iterates $\{x_k\}$ generated by
Algorithm 3 converge to a nondegenerate local minimizer $x^*$. Then $\mathcal{A}(x_k) = \mathcal{A}(x^*)$
for all $k$ sufficiently large.

We now need to determine what algorithms we will use in Stage 2 of Algorithm
3. We begin with the conjugate gradient algorithm.

**Unconstrained Minimization**

If $x^*$ is a solution of (2.3), once $\mathcal{A}(x^*)$ is identified, problem (2.3) reduces to the
unconstrained minimization problem

$$\min_{\xi} \hat{J}(\xi). \tag{2.32}$$

where $\hat{J}$ and $\xi$ are defined in the paragraph following the proof of Lemma 2.28. We
can then use techniques for unconstrained optimization to solve (2.32).

We begin our discussion of unconstrained optimization techniques for finding a
solution of problem (2.32) with a special case.
The Conjugate Gradient Algorithm

Definition 2.47. If $\hat{J} : \mathbb{R}^N \to \mathbb{R}$ is defined by

$$
\hat{J}(x) = \frac{1}{2} \langle x, Ax \rangle - \langle x, b \rangle + c,
$$

(2.33)

where $A \in \mathbb{R}^{N \times N}$, $b \in \mathbb{R}^N$ and $c \in \mathbb{R}$, then $\hat{J}$ is a quadratic function.

Remark 2.48. Without loss of generality we may assume that $A$ is symmetric [35, p. 144]. Furthermore, if $A$ is positive definite, the unique global constrained minimizer is given by $x^* = A^{-1}b$, which can be obtained by solving $\text{grad} \, \hat{J}(x) = 0$.

The conjugate gradient algorithm (CG) is an iterative algorithm that minimizes (2.33), or equivalently, solves $Ax = b$, where $A$ is SPD.

Algorithm 4. (Conjugate Gradient Method for Quadratic Minimization)

\begin{align*}
\nu & := 0; \\
x_0 & := \text{initial guess} \\
g_0 & := Ax_0 - b; \quad \% \text{initial gradient} \\
p_0 & := -g_0; \quad \% \text{initial search direction} \\
\delta_0 & = ||g_0||^2; \\
\text{begin CG iterations} \\
\quad h_\nu & := Ap_\nu; \\
\quad \tau_\nu & := \delta_\nu / \langle p_\nu, h_\nu \rangle; \quad \% \text{line search parameter} \\
\quad x_{\nu+1} & := x_\nu + \tau_\nu p_\nu; \quad \% \text{update approximate solution}
\end{align*}
\( \mathbf{g}_{\nu+1} := \mathbf{g}_\nu + \tau_\nu \mathbf{h}_\nu; \quad \% \text{update gradient} \)

\[ \delta_{\nu+1} = \| \mathbf{g}_{\nu+1} \|^2; \]

\[ \beta_\nu := \delta_{\nu+1}/\delta_\nu; \]

\[ \mathbf{p}_{\nu+1} := -\mathbf{g}_{\nu+1} + \beta_\nu \mathbf{p}_\nu; \quad \% \text{update search direction} \]

\[ \nu := \nu + 1; \]

end CG iterations

We next provide a characterization of the CG iterates \( \mathbf{x}_\nu \). See [39] or [1] for details.

**Definition 2.49.** The \( \nu^{th} \) Krylov subspace generated by an SPD matrix \( A \) and a vector \( \mathbf{v} \in \mathbb{R}^n \) is given by

\[ \mathcal{S}_\nu(A, \mathbf{v}) \overset{\text{def}}{=} \text{span}(\mathbf{v}, A\mathbf{v}, \ldots, A^{\nu-1}\mathbf{v}) \]

\[ = \{ p(A)\mathbf{v} \mid p \in \Pi^{\nu-1} \}, \]

where \( \Pi^{\nu-1} \) denotes the set of polynomials of degree less than or equal to \( \nu - 1 \).

**Definition 2.50.** For an SPD matrix \( A \), the \( A \)-norm of a vector \( \mathbf{x} \in \mathbb{R}^N \) is defined by \( \| \mathbf{x} \|^2_A = \mathbf{x}^T A \mathbf{x} \).

**Theorem 2.51.** For \( \nu = 1, 2, \ldots \), the CG iterates \( \mathbf{x}_\nu \) satisfy

\[ \mathbf{x}_\nu = \arg \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{S}_\nu(A, \mathbf{x}_0)} \| \mathbf{x} - \mathbf{x}_* \|_A. \quad (2.34) \]

The corresponding iterative solution errors \( \mathbf{e}_\nu = \mathbf{x}_\nu - \mathbf{x}_* \) satisfy

\[ \| \mathbf{e}_\nu \|_A = \min_{q \in \pi_1} \| q(A) \mathbf{e}_0 \|_A, \quad (2.35) \]
where $\pi^\nu_1$ denotes the set of polynomials $q(t)$ of degree less than or equal to $\nu$ for which $q(0) = 1$.

**Corollary 2.52.** If $A \in \mathbb{R}^{N \times N}$ is SPD and has $m$ distinct eigenvalues (note that $m \leq N$), then for any $b \in \mathbb{R}^n$ and any initial guess $x_0 \in \mathbb{R}^n$, the conjugate gradient algorithm will yield the exact solution to the system $Ax = b$ in at most $m$ iterations.

**Remark 2.53.** In the case where $x_0 = 0$, CG generates a sequence of polynomial approximations $p_\nu(A)$ to $A^{-1}$. If $b$ is nondegenerate (i.e., the projections onto each of the eigenspaces are nonzero) then after at most $m$ iterations, $p_\nu(\lambda_i) = 1/\lambda_i$ for each eigenvalue $\lambda_i$ of $A$, and $p_\nu(A) = A^{-1}$.

Using a particular scaled Chebyshev polynomial $p \in \pi_1^\nu$ in (2.35) [1], one can compute the iterative error bound

$$\|e_\nu\|_A \leq 2 \left(\frac{\sqrt{\text{cond}(A)} - 1}{\sqrt{\text{cond}(A)} + 1}\right)^\nu \|e_0\|_A,$$  \hspace{1cm} (2.36)

where cond$(A)$ is the condition number of $A$ and is given by cond$(A) = \|A\| \|A^{-1}\|$. Hence, a small condition number implies a rapid rate of CG convergence.

We can accelerate CG by transforming the system $Ax = b$ into a system $\hat{A}\hat{x} = \hat{b}$ on which CG has a faster rate of convergence. We do this via preconditioning. Preconditioning is a change of variables from $x$ to $\hat{x}$ by a nonsingular matrix $C$, i.e., $\hat{x} = Cx$. Under this change of variables, the quadratic function $\hat{J}$ given in Definition 2.47 is expressed as

$$\hat{J}(\hat{x}) = \frac{1}{2} \hat{x}^T (C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T \hat{x}.$$  \hspace{1cm} (2.37)
If we use CG to minimize \( \hat{J} \) under this change of variables, the convergence rate will depend on the condition number of \( C^{-T}AC^{-1} \) rather than on the condition number of \( A \). Hence, if we choose \( C \) so that the condition number of \( C^{-T}AC^{-1} \) is smaller than that of \( A \), CG will converge more rapidly. We can also obtain more rapid convergence if the eigenvalues of \( C^{-T}AC^{-1} \) are clustered. See the discussion following Theorem 5.1 in [32].

To implement the preconditioned conjugate gradient (PCG) method, we apply CG to the problem of minimizing \( \hat{J} \) in terms of \( \hat{x} \). We then reexpress all of the equations in Algorithm 4 in terms of \( x \) by using the change of variables equation \( \hat{x} = Cx \). The resulting algorithm is given by Algorithm 5.3 in [32], which does not use the matrix \( C \) explicitly. Instead it uses the matrix \( M = C^TC \). \( M \) is known as the preconditioning matrix. \( M = I \) corresponds to no preconditioning, in which case PCG and CG are equivalent algorithms.

The algorithm presented next uses PCG as a tool for the unconstrained minimization of general nonlinear functions.

**The Newton-CG-Trust Region Method**

As one might expect, the problem of minimizing a general nonlinear function is much more difficult than is quadratic minimization. Two well known methods for general nonlinear minimization are the method of steepest descent and Newton’s method (see [32]). If certain conditions are satisfied, the method of steepest descent is a globally convergent algorithm. Unfortunately, as we have already mentioned,
the steepest descent method tends to be very slow in converging. On the other hand, Newton’s method has a locally quadratic convergence rate, but is not globally convergent. Even if the cost function is convex, the Newton iterates may fail to converge. See [13, p. 24] for an illustration. Two approaches are typically used to globalize Newton’s method. The first is a line search (see [32]), and the second is a trust region strategy [13, 32]. The key idea is to form at each iteration \( \nu \) a quadratic approximation to \( J(x_{\nu} + s) \),

\[
m_\nu(s) = J(x_{\nu}) + g_\nu^T s + \frac{1}{2} s^T H_\nu s,
\]

where \( g_\nu = \text{grad } J(x_{\nu}) \) and \( H_\nu = \text{Hess } J(x_{\nu}) \). Then solve the constrained minimization problem

\[
\min_s m_\nu(s) \quad \text{subject to } \|s\|_\nu \leq \Delta_\nu,
\]

where \( \| \cdot \|_\nu \) is a matrix norm which may depend on the Newton iteration \( \nu \), and \( \Delta_\nu \) is a positive parameter called the trust region radius. This parameter is adjusted at each iteration \( \nu \) in a manner which guarantees that (i) \( m_\nu(s) \) accurately approximates \( J(x_{\nu} + s) \) within the trust region \( \{s \mid \|s\|_\nu \leq \Delta_\nu\} \); and (ii) the constrained minimizer \( s_\nu \) for (2.38)-(2.39) yields a new approximation \( x_{\nu+1} = x_{\nu} + s_\nu \) which sufficiently decreases \( J \) in order to guarantee convergence to a minimizer. See [32, Algorithm 4.1] for a specific example.

Problem (2.38)-(2.39) has no closed-form solution, and obtaining a highly accurate approximation can be very expensive. For large-scale problems, a very effective approximate solution technique for subproblem (2.39) is the Steihaug-Toint algorithm.
The key idea is to apply conjugate gradient (CG) iterations to minimize \( m_\nu(s) \), or equivalently, to solve the linear system \( H_\nu s = -g_\nu \). If the initial guess \( s_{\nu,0} \) is zero and \( H_\nu \) is SPD, the CG iterates \( s_{\nu,j} \) monotonically increase in norm and monotonically decrease \( m_\nu(s) \) [40]. Eventually, either the boundary of the trust region is crossed or the minimizer is attained. If \( H_\nu \) is indefinite, a check for negative curvature is added. If \( d_{\nu,j}^T H_\nu d_{\nu,j} \leq 0 \), where \( d_{\nu,j} \) denotes the \( j^{th} \) CG search direction, then one moves in this direction from the current CG iterate until the trust region boundary is crossed, and the CG iteration is terminated. As mentioned in the previous section, preconditioning can be incorporated to accelerate CG converge. In this case the trust region is determined by the vector norm induced by the SPD preconditioning matrix \( M_\nu \),

\[
\|s\|_{M_\nu} \overset{\text{def}}{=} \sqrt{s^T M_\nu s}.
\]  

(2.40)

Given gradient \( g = \text{grad } J(x_\nu) \), Hessian \( H = \text{Hess } J(x_\nu) \), preconditioning matrix \( M = M_\nu \) and trust region radius \( \Delta = \Delta_\nu \), to approximately solve the trust region subproblem (2.38)-(2.39):

**Algorithm 5.** (The Steihaug-Toint Truncated (preconditioned) CG Algorithm)

\[
s_0 := 0;
\]

\[
r_0 := g_\nu; \quad \% \text{ Initial CG residual}
\]

\[
y_0 := M^{-1} r_0; \quad \% \text{ Apply preconditioner}
\]

\[
d_0 := -y_0; \quad \% \text{ Initial CG search direction}
\]

\[
j := 0;
\]
Begin CG iterations

$$\kappa_j := d_j^T Hd_j;$$

If \( \kappa_j \leq 0, \) \hspace{1em} \% \hspace{0.5em} \text{Check for indefinite } H

$$\tau_j := \text{positive root of } ||s_j + \tau d_j||_M = \Delta;$$

$$s_{j+1} := s_j + \tau_j d_j; \hspace{1em} \% \hspace{0.5em} \text{Move to boundary}$$

Stop.

End if

$$\alpha_j := \frac{r_j^T y_j}{\kappa_j};$$

If \( ||s_j + \alpha_j d_j||_M \geq \Delta, \) \hspace{1em} \% \hspace{0.5em} \text{Boundary crossed}

$$\tau_j := \text{positive root of } ||s_j + \tau d_j||_M = \Delta;$$

$$s_{j+1} := s_j + \tau_j d_j; \hspace{1em} \% \hspace{0.5em} \text{Backtrack to boundary}$$

Stop.

End if

$$s_{j+1} := s_j + \alpha_j d_j; \hspace{1em} \% \hspace{0.5em} \text{Update solution}$$

$$r_{j+1} := r_j + \alpha_j Hd_j; \hspace{1em} \% \hspace{0.5em} \text{Update CG residual}$$

$$y_{j+1} := M^{-1} r_{j+1}; \hspace{1em} \% \hspace{0.5em} \text{Apply preconditioner}$$

$$\beta_{j+1} := \frac{r_{j+1}^T y_{j+1}}{r_j^T y_j};$$

$$d_{j+1} := -r_{j+1} + \beta_{j+1} d_j; \hspace{1em} \% \hspace{0.5em} \text{Update CG search direction}$$

$$j := j + 1;$$

end CG iterations
We will refer to the resulting algorithm as the Newton-CG-trust region (NCGTR) method. NCGTR is globally convergent (see [32, Section 4.3]).

The BFGS Method

Quasi-Newton methods with line search globalization provide an alternative means of solving unconstrained minimization problems. The following algorithm gives the general form of a quasi-Newton method with a line search.

Algorithm 6. (Quasi-Newton Method with Line Search)

\[
\begin{align*}
\nu & := 0; \\
x_0 & := \text{initial guess for minimizer;} \\
H_0 & := \text{initial Hessian approximation;} \\
g_0 & := \text{grad } J(x_0); \quad \% \text{ initial gradient}
\end{align*}
\]

begin quasi-Newton iterations

\[
\begin{align*}
p_{\nu+1} & := -H_\nu^{-1}g_\nu; \quad \% \text{ compute quasi-Newton step} \\
\tau_{\nu+1} & := \arg \min_{\tau > 0} J(x_\nu + \tau p_{\nu+1}); \quad \% \text{ line search} \\
x_{\nu+1} & := x_\nu + \tau_{\nu+1}p_{\nu+1}; \quad \% \text{ update approximate solution} \\
g_{\nu+1} & := \text{grad } J(x_{\nu+1}); \quad \% \text{ new gradient} \\
H_{\nu+1} & := \text{updated Hessian approximation;} \\
\nu & := \nu + 1; \\
\end{align*}
\]

end quasi-Newton iterations
The choice at each iteration of the Hessian approximation \( H_\nu \) determines the particular quasi-Newton method. If \( H_\nu = \text{Hess} J(x_\nu) \) for each \( \nu \) then Algorithm 6 is Newton’s method with a line search, which has a locally quadratic convergence rate (see [32, Theorem 3.7]). If \( H_\nu = I \) for each \( \nu \) then Algorithm 6 is the steepest descent method, which has a linear convergence rate (see [32, Theorem 3.3]). We seek a quasi-Newton method which balances the low cost per iteration of the steepest descent method with the fast local convergence of Newton’s method.

Many recursive formulas exist for the Hessian approximation \( H_\nu \) (see [32, Chapter 8]). The BFGS recursion is perhaps the most popular. To implement the BFGS recursion, given an approximation \( H_\nu \) to Hess \( J(x_\nu) \), one first computes \( s_\nu = x_{\nu+1} - x_\nu \) and \( y_\nu = \text{grad} J(x_{\nu+1}) - \text{grad} J(x_\nu) \). One then computes an approximation to \( \text{Hess} J(x_{\nu+1}) \),

\[
H_{\nu+1} = H_\nu - \frac{H_\nu s_\nu s_\nu^T H_\nu}{\langle H_\nu s_\nu, s_\nu \rangle} + \frac{y_\nu y_\nu^T}{\langle y_\nu, s_\nu \rangle}.
\]  

(2.41)

If \( H_\nu \) is symmetric positive definite and \( \langle y_\nu, s_\nu \rangle > 0 \), then \( H_{\nu+1} \) will also be symmetric positive definite. This fact is important in our computation of \( p_{\nu+1} \) at each iteration in Algorithm 6.

We will refer to Algorithm 6 with Hessian update (2.41) as the BFGS method. It is important to note that, as with the steepest descent method, the BFGS method requires only function and gradient information. Consequently, the BFGS method has a lower cost per iteration than does Newton’s method. On the other hand, information from previous iterations is carried along. The result is that the BFGS
method has a faster rate of convergence than does the steepest descent method, as is
demonstrated in the following theorem [32, Theorem 8.6].

**Theorem 2.54.** Suppose $J \in \mathcal{C}^2$ and that the BFGS method generates a sequence
\$
\{x_k\}
\$
that converges to a minimizer $x^*$ at which Hess $J(x^*)$ is Lipschitz continuous.
Suppose also that\$
\sum_{k=1}^{\infty} ||x_k - x^*|| < \infty.
\$
Then $x_k$ converges to $x^*$ at a superlinear rate.

BFGS has a limited memory variant [32, p. 224], which requires no explicit
matrix storage for the approximate Hessian matrix. It is based on the recursion for
the inverse,
\[
H_{\nu+1}^{-1} = \left( I - \frac{s_{\nu}y_{\nu}^T}{\langle y_{\nu}, s_{\nu} \rangle} \right) H_{\nu}^{-1} \left( I - \frac{y_{\nu}s_{\nu}^T}{\langle y_{\nu}, s_{\nu} \rangle} \right) + \frac{s_{\nu}s_{\nu}^T}{\langle y_{\nu}, s_{\nu} \rangle}.
\] (2.42)

Given $g \in \mathbb{R}^n$, computation of $H_{\nu+1}^{-1}g$ requires a sequence of inner products involving
$g$ and the $s_{\nu}$'s and $y_{\nu}$'s, together with the application of $H_0^{-1}$. If $H_0$ is SPD and the
"curvature condition" $y_{\nu}^Ts_{\nu} > 0$ holds for each $\nu$, then each of the $H_{\nu}$'s is also SPD.
"Limited memory" means that at most $n$ vector pairs \$(s_{\nu}, y_{\nu})$, \ldots, $(s_{\nu-n+1}, y_{\nu-n+1})\$$
are stored and at most $n$ steps of the recursion are taken, i.e., if $\nu \geq n$, apply the
recursion (2.42) for $\nu, \nu - 1, \ldots, \nu - n$, and set $H_{\nu-n}^{-1}$ equal to an SPD matrix $M_\nu^{-1}$. We
will refer to $M_\nu$ as the preconditioning matrix. In standard implementations, $M_\nu$
is taken to be a multiple of the identity [32]. Choices of $M_\nu$ based on structure of the cost
function $J$ can also be taken. We will refer to Algorithm 6 with this limited memory
Hessian update as the L-BFGS method. The L-BFGS method is particularly useful
for solving large problems in which matrix storage is an issue, and despite modest
storage requirements the resulting algorithm often yields an acceptable, albeit linear, rate of convergence.
CHAPTER 3

LOWER BOUND CONSTRAINED QUADRATIC PROGRAMMING

In this chapter we will concentrate on methods for finding the solution of

$$\min_{x \geq 0} J(x)$$

(3.1)

when $J$ is quadratic (see Definition 2.47). This problem is known as a lower bound constrained quadratic program. The gradient and Hessian of $J$ are given by $\text{grad } J(x) = Ax - b$ and $\text{Hess } J(x) = A$, respectively. We will assume that $A$ is SPD. Then $J$ is strictly convex and coercive. The next theorem then follows immediately from Theorems 2.9 and 2.27.

**Theorem 3.1.** Let $J$ be a quadratic function with a positive definite Hessian matrix $A$. Then (3.1) has a unique global solution $x^*$, and $\text{grad}_p J(x) = 0$ if and only if $x = x^*$.

**Remark 3.2.** One consequence of this theorem is that if $\{x_k\} \subset \Omega$ is a sequence such that $\text{grad}_P J(x_k) \rightarrow 0$ then $\{x_k\}$ converges to the global solution $x^*$ of problem (3.1). This is a very powerful result. In general, even if we know that $\{x_k\}$ converges to some $\hat{x} \in \Omega$ with $\text{grad}_P J(x_k) \rightarrow 0$, there is no guarantee that $\hat{x}$ is a local constrained minimizer. For this to be true $\text{Hess}_R J(\hat{x})$ must be positive semidefinite (see Theorem 2.30), which does not guarantee that $\hat{x}$ is the unique global constrained minimizer of $J$. 
Remark 3.3. Once $\mathcal{A}(\mathbf{x}^*)$ is determined, (3.1) becomes an unconstrained quadratic minimization problem, and hence, to find the global constrained minimizer, we only need to solve a linear system. Finding the exact solution of a large linear system is often not practical. If this is the case, and the corresponding matrix is SPD, CG provides an excellent means for finding an approximate solution. This, together with Algorithm 3, motivate the two stage algorithms for large-scale bound constrained quadratic programming presented in this chapter.

**The Gradient Projection-Conjugate Gradient Algorithm**

The Gradient Projection Conjugate Gradient algorithm (GPCG), is an algorithm developed for the minimization of large-scale bound constrained quadratic programs [30]. It follows the general framework of the two stage algorithm given by Algorithm 3. The idea behind GPCG is to use the identification properties of the gradient projection algorithm to determine a face

$$
\mathcal{F}(\mathbf{x}_k) \overset{\text{def}}{=} \{ \mathbf{x} \in \Omega \mid [\mathbf{x}_k]_i = L_i \text{ for } i \in \mathcal{A}(\mathbf{x}_k) \text{ and } [\mathbf{x}_k]_i > L_i \text{ otherwise} \} \tag{3.2}
$$

suitable for further exploration, together with the CG iteration to aid in finding an approximate minimum of $\mathcal{J}$ on $\mathcal{F}(\mathbf{x}_k)$.

Given an approximation $\mathbf{x}_k$ to the global minimizer $\mathbf{x}^*$ of (3.1), GPCG generates $\mathbf{x}_{k+1}$ in the following way: First, we set $\mathbf{y}_0 = \mathbf{x}_k$ and generate a sequence $\{\mathbf{y}_j\}$ of gradient projection iterates using Algorithm 2. We terminate the iteration as soon
as either

\[ J(y_{j-1}) - J(y_j) \leq \gamma \max \{ J(y_{i-1}) - J(y_i) \mid i = 1, \ldots, j \}, \tag{3.3} \]

where \(0 < \gamma < 1\), or

\[ A(y_j) = A(y_{j-1}). \tag{3.4} \]

We then replace \(x_k\) by \(y_{j_k}\), where \(j_k\) is the first index in which either (3.3) or (3.4) are satisfied. Typically we take \(\gamma = \frac{1}{4} \) [30]. If (3.3) holds, the implication is that the gradient projection algorithm is not making sufficient progress. When (3.4) holds the implication is that we are near the optimal face \(\mathcal{F}(x^*)\), since, unless close to a solution to (3.1), gradient projection iterations tend to produce changes in the active set. In either case, we use CG to explore the face \(\mathcal{F}(x_k)\) further.

CG creates a new sequence of iterates \(\{d_j\}\) which, in a finite number of steps, converges to a solution of the reduced system

\[ \text{Hess}_R J(x_k) d = \hat{b}, \tag{3.5} \]

where \(\hat{b}_i = b_i \) for \(i \in \mathcal{I}(x_k)\) and is zero otherwise. We terminate CG if (3.3) holds, where \(y_j \equiv x_k + d_j\) for each \(j\). In this case, we typically take \(\gamma = \frac{1}{10} \) [30]. Again, the implication here is that CG is not making sufficient progress. We then let \(p_k = d_{j_k}\), where \(j_k\) is the first index that satisfies (3.3) and perform an inexact line search to find \(\lambda_k\). Then \(x_{k+1} := P_\Omega(x_k + \lambda_k p_k)\). At this point we compute the binding set

\[ \mathcal{B}(x_{k+1}) \equiv \{ i \in \mathcal{A}(x_{k+1}) \mid \frac{\partial J}{\partial x_i}(x_{k+1}) \geq 0 \}. \]
Based on the observation that if \( x \in \mathcal{F}(x^*) \) then \( B(x) = A(x) \) \([30]\), we return to CG if \( B(x_{k+1}) = A(x_{k+1}) \). Otherwise, we return to the gradient projection stage of the algorithm. The basic outline of GPCG is then given as follows:

**Algorithm 7.** (GPCG Algorithm)

**Step 0:** Select initial guess \( x_0 \), and set \( k = 0 \).

**Step 1:** Given \( x_k \).

1. Do Algorithm 1 with \( y_0 = x_k \) until either (3.3) or (3.4) are satisfied.
2. Return \( x_k \leftarrow y_{j_k} \).

**Step 2:** Given \( x_k \).

1. Do CG iterations on the reduced system (3.5) until (3.3) is satisfied.
2. Set \( p_k = d_{j_k} \) and perform a line search. Return \( x_{k+1} \).

**Step 3:** Given \( x_{k+1} \).

1. If the stopping criteria is satisfied, STOP.
2. If \( B(x_{k+1}) = A(x_{k+1}) \), update \( k := k + 1 \) and return to Step 2.
3. Otherwise, update \( k := k + 1 \) and return to Step 1.

**Remark 3.4.** If at Step 3, (2) we decide to return to Step 2, we do one of two things. If \( A(x_{k+1}) = A(x_k) \) then the reduced system (3.5) has not changed, in which case we continue where we had left off in the previous CG iteration. Otherwise, we restart CG. In both cases, we set \( j = 1 \) in (3.3).
Remark 3.5. The line search algorithm used in GPCG, and that given in [30], is very similar to the quadratic backtracking line search given by Algorithm 2. First, we define

$$\beta_1 \overset{\text{def}}{=} \arg \sup \{ \beta > 0 \mid x + \beta p \in \Omega \}, \quad (3.6)$$

where $\sup$ denotes the least upper bound. Then, if $x + \beta p \in \Omega$ for all $\beta$, $\beta_1 = \infty$. $x$ and $p$ are determined by where the line search is being performed within Algorithm 7. In Step 1, $x = y_j$ and $p = -\nabla_{\beta} J(y_j)$. In Step 2, $x = x_k$ and $p = p_k$. In either case, we begin by computing

$$\lambda^0 = \frac{-\langle \nabla J(x), p \rangle}{\langle \text{Hess} J(x) p, p \rangle}. \quad (3.7)$$

Then, given a $\lambda^i$ that does not satisfy (2.31), we compute $\lambda^{i+1}$ by first defining $\hat{\lambda}^{i+1}$ to be the minimizer of the quadratic function $q_k$ that satisfies $q_k(0) = \phi_k(0)$, $q_k'(0) = \phi_k'(0)$, and $q_k(\lambda^i_k) = \phi_k(\lambda^i_k)$. Then

$$\lambda^{i+1} \overset{\text{def}}{=} \text{median} \left[ \frac{1}{100} \lambda^i, \hat{\lambda}^{i+1}, \frac{1}{2} \lambda^i \right]. \quad (3.8)$$

The line search algorithm is then given follows:

Algorithm 8. (GPCG Line Search Algorithm)

(0) Compute $\lambda^0$ as in (3.7), and set $i = 0$.

(1) If $\lambda^i \leq \beta_1$,

Case 1: $i = 0$. Set $\lambda = \lambda^0$ and STOP.

Case 2: $i > 0$. Set $\lambda = \beta_1$ and STOP.
(2) Otherwise, if \( \lambda^i \) satisfies (2.31), set \( \lambda = \lambda^i \) and STOP.

(3) Otherwise, compute \( \lambda^{i+1} \) by equation (3.8).

(4) Update \( i := i + 1 \) and return to (1).

We now prove that Algorithm 8 will generate a suitable \( \lambda \) in a finite number of iterations.

**Theorem 3.6.** In Algorithm 8, there exists an \( N \) such that such that \( \lambda^N \) satisfies the sufficient decrease condition (2.31).

**Proof.** Given \( \mathbf{x} \) and \( \mathbf{p} \) within either Step 1 or Step 2 of Algorithm 7, define \( \hat{\phi}(\lambda) = J(\mathbf{x}) + \lambda \langle \text{grad } J(\mathbf{x}), \mathbf{p} \rangle + \frac{\lambda^2}{2} \langle \text{Hess } J(\mathbf{x}) \mathbf{p}, \mathbf{p} \rangle \). Given our assumptions on \( J, \text{Hess } J(\mathbf{x}) = A \) for all \( \mathbf{x} \), where \( A \) is SPD. Hence \( \hat{\phi} \) is a strictly convex quadratic function of \( \lambda \). Define \( \ell(\lambda) = J(\mathbf{x}) + \frac{\lambda}{2} \langle \text{grad } J(\mathbf{x}), \mathbf{p} \rangle \). It is not difficult to show that \( \langle \text{grad } J(\mathbf{x}), \mathbf{p} \rangle < 0 \), and that \( \ell \) intersects \( \hat{\phi} \) at \( \lambda = 0 \) and \( \lambda = \lambda^0 \). Then, since \( \hat{\phi} \) is strictly convex, \( \hat{\phi}(\lambda) \leq \ell(\lambda) \) for all \( \lambda \in [0, \lambda^0] \). Hence, if \( \lambda^0 \leq \beta_1 \) then \( \lambda = \lambda^0 \) satisfies sufficient decrease condition (2.31) with \( \mu = \frac{1}{2} \). Otherwise, \( \lambda^0 > \beta_1 \), and by the argument given in the proof of Theorem 2.36, there exists and \( N \) such that \( \lambda^N \leq \beta_1 < \lambda^{N-1} \). Hence, either there exists an \( i < N \) such that \( \lambda = \lambda^i \) satisfies the sufficient decrease condition (2.31) with \( \mu \in (0, \frac{1}{2}) \), or Algorithm 8 returns \( \lambda = \beta_1 \), which satisfies sufficient decrease condition (2.31) with \( \mu = \frac{1}{2} \). \( \square \)

**Remark 3.7.** Theorem 3.6 tells us that Algorithm 7 is well-defined. Furthermore, Algorithm 7 is a two stage algorithm of the form of Algorithm 3, and hence, since
$J$ is convex and coercive, the result of Corollary 2.44 holds. The fact that $J$ is also quadratic allows for the even stronger result given in the next theorem, a proof of which can be found in [30].

**Theorem 3.8.** Let $J$ be a strictly convex quadratic function, and suppose that $\mathbf{x}^*$ is the (nondegenerate) solution of problem (3.1). Then Algorithm 7 terminates at the solution $\mathbf{x}^*$ in a finite number of iterations.

**Remark 3.9.** Since the global solution $\mathbf{x}^*$ is nondegenerate, GPCG must identify the optimal face $\mathcal{F}(\mathbf{x}^*)$ before it terminates. Therefore the asymptotic convergence rate of GPCG will be driven by the convergence rate of CG (see Theorems 2.52 and 2.36).

**Remark 3.10.** We have implemented several slight variants of Algorithm 7, but any improvements in performance that resulted were negligible.

---

**The GPCG-FM Algorithm**

We now present the Friedlander-Martinez (FM) modification of GPCG presented in [14]. We begin by defining two new vectors.

**Definition 3.11.** Let $J : \mathbb{R}^N \to \mathbb{R}$ be smooth. We define the internal gradient of $J$ at $\mathbf{x}$ by

$$[\text{grad}_I J(\mathbf{x})]_i = \begin{cases} \text{grad } J(\mathbf{x})_i, & i \in \mathcal{I}(\mathbf{x)} \\ 0, & \text{otherwise}, \end{cases}$$
and the chopped gradient of $J$ at $\mathbf{x}$ by

$$[\text{grad}^C J(\mathbf{x})]_i = \begin{cases} 
\text{grad } J(\mathbf{x})_i, & i \in \mathcal{A}(\mathbf{x}) \text{ and } \text{grad } J(\mathbf{x})_i > 0 \\
0, & \text{otherwise.}
\end{cases}$$

We can think of $\text{grad}^C J(\mathbf{x})$ as the exit gradient, because it is nonzero only for those indices $i$ for which $\mathbf{x} - \alpha \frac{\partial J}{\partial x_i}(\mathbf{x}) \mathbf{e}_i \notin \Omega$ for all $\alpha > 0$.

GPCG-FM is similar to GPCG in that it combines gradient projection and CG iterations, though it does this in a very different way. In the first stage of GPCG-FM, given an approximate solution $\mathbf{x}_k$ to (3.1), a gradient projection step is taken only if

$$||\text{grad} J(\mathbf{x}_k)|| < ||\text{grad}^C J(\mathbf{x}_k)||^2 / 2D_k,$$  \hspace{1cm} (3.9)

where $D_k$ is a scalar. If (3.9) holds, the implication is that the “internal gradient” $\text{grad} J(\mathbf{x}_k)$ is “small” compared to the “exit gradient” $\text{grad}^C J(\mathbf{x}_k)$, and, hence, we ought to leave the current face $\mathcal{F}(\mathbf{x}_k)$. We leave $\mathcal{F}(\mathbf{x}_k)$ by computing $\lambda_k$ via an inexact line search. Then $\mathbf{x}_{k+1} := \mathcal{P}_{\Omega}(\mathbf{x}_k - \lambda_k \text{grad } J(\mathbf{x}_k))$. We continue with the gradient projection iterations until (3.9) no longer holds.

In the second stage of GPCG-FM, given $\mathbf{x}_k$ we apply CG to the reduced system (3.5). If $k = 0$ or $\mathcal{F}(\mathbf{x}_k) \neq \mathcal{F}_{k-1}$, we have a new reduced system and we restart CG. Then $\mathbf{p}_k = -\text{grad} J(\mathbf{x}_k)$. Otherwise, our linear system is the same as that of the previous iteration. We therefore compute the next CG direction $\mathbf{p}_k = -\text{grad} J(\mathbf{x}_k) + \beta_k \mathbf{p}_{k-1}$, where $\beta_k = ||\text{grad} J(\mathbf{x}_k)||^2 / ||\text{grad} J(\mathbf{x}_{k-1})||^2$, and $\mathbf{p}_{k-1}$ is the previous CG search direction. We then compute

$$\lambda_0 = \arg\min_{\lambda > 0} J(\mathbf{x}_k + \lambda \mathbf{p}_k) = \frac{\langle -\text{grad } J(\mathbf{x}_k), \mathbf{p}_k \rangle}{\langle A \mathbf{p}_k, \mathbf{p}_k \rangle}.$$
If \( x_k + \lambda_0 p_k \in \Omega \) we set \( \lambda_k = \lambda_0 \). Otherwise, we compute \( \lambda_k > 0 \) via an inexact line search. Then \( x_{k+1} := \mathcal{P}_\Omega(x_k + \lambda_k p_k) \). The basic outline of GPCG-FM is then given as follows:

**Algorithm 9.** (GPCG-FM Algorithm)

- **Step 0:** Select initial guess \( x_0 \), and set \( k = 0 \).

- **Step 1:** Given \( x_k \).
  
  (1) If \((3.9)\) holds then perform a line search. Return \( x_{k+1} \). Go to Step 3.

- **Step 2:** Given \( x_k \).
  
  (1) Compute CG direction \( p_k \).
    
    **Case 1:** \( k = 0 \) or \( \mathcal{F}(x_k) \neq \mathcal{F}(x_{k-1}) \). Then \( p_k = -\nabla f(x_k) \).
    
    **Case 2:** Otherwise, \( p_k = -\nabla f(x_k) + \beta_{k-1} p_{k-1} \).
  
  (2) Compute \( \lambda_0 = \arg\min_{\lambda > 0} J(x_k + \lambda p_k) \).
    
    **Case 1:** \( x_k + \lambda_0 p_k \in \Omega \). Then \( \lambda_k = \lambda_0 \).
    
    **Case 2:** Otherwise, perform a line search. Return \( \lambda_k \).
  
  (3) Set \( x_{k+1} = \mathcal{P}_\Omega(x_k + \lambda_k p_k) \). Update \( D_k \).

- **Step 3:** Given \( x_{k+1} \).
  
  (1) If the stopping criteria is satisfied, STOP.
  
  (2) Otherwise, update \( k := k + 1 \) and return to Step 1.

**Remark 3.12.** It is worth commenting on the difference between how GPCG and GPCG-FM use both the gradient projection and CG iterations. As far as the gradient projection iteration is concerned, there is marked difference. GPCG uses the
gradient projection iteration both as means of identifying the optimal face \( \mathcal{F}(x^*) \) and of decreasing the value of the objective function \( J \). GPCG-FM, on the other hand, uses gradient projection iterations primarily as a means of leaving the current face. Consequently, GPCG tends to take many more gradient projection steps than does GPCG-FM.

There is also a marked difference in the way in which CG is used. In GPCG, given a particular reduced system (3.5), CG is allowed to iterate outside of \( \Omega \) as long as a sufficient decrease in function value is occurring at each iteration. In contrast, in GPCG-FM, the CG iteration is stopped as soon as a CG iterate lies outside of \( \Omega \).

We have explored the question: Which algorithm uses CG and the gradient projection algorithm most effectively? We have done this by mixing the gradient projection stages of both algorithms with the CG stages and have found that, for our application, GPCG’s use of both the gradient projection algorithm and of CG is more effective. On the other hand, GPCG-FM, in its full complexity, allows for both a degenerate solution \( x^* \) and a symmetric positive semidefinite matrix \( A \) [14].

**Remark 3.13.** In [14] the quadratic backtracking line search given in Algorithm 8 is used, with changes made only in the sufficient decrease condition (2.31). In Step 1, replace (2.31) by

\[
\phi_k(\lambda) < \phi_k(0) - \|\text{grad}_I J(x_k)\| D_k,
\]

(3.10)
while in Step 2, we replace (2.31) with

$$\phi_k(\lambda) < \phi_k(0).$$  \hfill (3.11)

In [14], an additional modification is made to the the line search to handle the case when the global minimizer $x^*$ is degenerate. Since in our application $x^*$ is a nondegenerate stationary point, we will not include this modification in the current discussion. Also, in [14], the case is considered in which the Hessian matrix $A$ is symmetric positive semidefinite. As was stated earlier, we are restricting our attention to the case where $A$ is SPD. The result is that Step 2, (2) of Algorithm 9 is a simplification of what is used in [14]. In any case, Algorithm 9 is the algorithm presented in [14] restricted to the case where $A$ is SPD and the global constrained minimizer $x^*$ is nondegenerate.

**Remark 3.14.** The update of $D_k$ takes place within Step 2 of Algorithm 9. First, in Step 0 we fix $\tau > 0$. The update of $D_k$ is then given as follows:

**Step 2 of Algorithm 9:** Given $x_k$.

1. If $k = 0$ or $F(x_k) \neq F(x_{k-1})$ then $\Delta J_{max} = 0$.

2. Compute $\Delta J_k = J(x_{k-1}) - J(x_k)$ and $\Delta J_{max} = \max\{\Delta J_k, \Delta J_{max}\}$.

   **Case 1:** $\Delta J_k < .1 \times \Delta J_{max}$. Then $D_{k+1} = \max\{\tau, .1 \times D_k\}$.

   **Case 2:** Otherwise, $D_{k+1} = D_k$.

**Remark 3.15.** The convergence result for GPCG-FM found in [14, Section 3] reduces to Theorem 3.8 for Algorithm 9, given the simplifications discussed in Remark
3.13. Note that GPCG-FM is not a two stage algorithm of the form of Algorithm 3, since the sufficient decrease condition (3.10) is used during gradient projection iteration.

**Other Possibilities**

A natural question arises following the analysis of the previous two algorithms. Namely, why not use an algorithm other than CG to solve the reduced system (3.5) in Stage 2? Theorem 2.51 tells us that CG is optimal, but at each CG iteration a Hessian vector multiplication is required. If this computation is either prohibitively expensive or has to be estimated, then it may be that CG is not the optimal choice.

In [16], Algorithm 9 is modified so that the Barzilai-Borwein algorithm [36] replaces CG for the unconstrained subspace minimization in Step 2. Step 1 and Step 3 remain unchanged. This method needs only gradient evaluations and has been shown to be superlinearly convergent for a class of problems [15]. Numerical experiments have shown that though this algorithm converges for our particular problem, it is not competitive with the CG based algorithms discussed in this chapter. This is most likely due to the fact that we can compute Hessian vector multiplications exactly and with a relatively low cost.

At the end of Chapter 2 we mentioned the preconditioned CG algorithm, which can be a much faster converging algorithm than is CG. A logical question, then, is: Can we use preconditioned CG in place of CG in the above algorithms? In [3] diagonal
preconditioners are explored. Unfortunately, diagonal preconditioners have proven to be ineffective when used on our problem. In [33] a strategy for preconditioning is used in which information from CG iterations within one iteration is saved and is then used to build a preconditioner for CG during the next iteration. But, because of changes in the active set from iteration to iteration, this has not proven to be an effective strategy. On the other hand, we know that the algorithms above eventually identify the optimal face. Once this occurs, this preconditioning strategy can be applied. When implemented, for our test problems, this strategy did not prove to be effective either. We have tried other preconditioning strategies, but to this point none have proven effective enough to be worth mentioning.

**Numerical Results**

In this section we solve the minimization problem (1.12), which is given by

$$
\min_{f \geq 0} J_\alpha(f),
$$

where $J_\alpha$ is given by

$$
J_\alpha(f) = \frac{1}{2} \|Sf - d\|^2 + \frac{\alpha}{2} \|f\|^2.
$$

Here $d$ and $f$ are $(n_x \cdot n_y) \times 1$ vectors and $S$ is the $(n_x \cdot n_y) \times (n_x \cdot n_y)$ BTTB matrix $S$ from the image formation model (1.4). In this chapter, $n_x = n_y = 128$. Since the matrix $S$ is BTTB the multiplication of a vector by $S$ can be computed using two-dimensional fast fourier transforms (FFTs) with order $(n_x \cdot n_y) \log(n_x \cdot n_y)$ cost.
Furthermore, to store $S$ we only need to store an $(n_x \cdot n_y) \times 1$ vector [42, Chapter 5]. In contrast, if $S$ is an arbitrary $(n_x \cdot n_y) \times (n_x \cdot n_y)$ matrix, $(n_x \cdot n_y)^2$ entries must be stored and matrix-vector multiplication cost is order $(n_x \cdot n_y)^2$. We restrict our attention to the discrete $L^2$ regularization functional $\frac{1}{2}||f||^2$

We can rewrite equation (3.13) as

$$J_\alpha(f) = \frac{1}{2} \langle f, Af \rangle - \langle f, b \rangle + c,$$  \hspace{1cm} (3.14)

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product, $A = S^*S + \alpha I$, $b = S^*d$, and $c = d^T d$. It is not difficult to show that the eigenvalues of $A$ are bounded away from zero by the positive regularization parameter $\alpha$, and hence, $A$ is positive definite. Problem (3.12)-(3.13) is therefore of the form of (3.1), and hence, we can apply GPCG and GPCG-FM.

We use four different data sets. Each of these data sets is synthetically generated using the matrix-vector model (1.4). As was stated in Chapter 1, the matrix $S$ in (3.14) depends on the PSF, which in turn depends upon the phase $\phi$. The phase and corresponding PSF used to generate our data is displayed in Figure 2. We use two synthetically generated true images $f_{\text{true}}$. The first is a binary star, and the second is a satellite. Because of the assumption made regarding the noise in the collection of the image $d$, which is characterized in equation (1.7), we can vary the intensity of $f_{\text{true}}$ in order to change the percentage of noise in the corresponding collected image $d$. We generate data with both one percent and ten percent noise. The percent noise
Figure 2. Phase and Corresponding PSF. To the left is a mesh plot of the discrete phase $\phi_{\text{true}}$. To the right is a mesh plot of the center $32 \times 32$ pixels of the corresponding PSF.

is given by

$$\frac{\sqrt{E(\|\eta\|^2)}}{\|Sf_{\text{true}}\|} \times 100,$$

where $E$ is the expected value [2]. Figure 3 contains plots of the true binary star and the corresponding data with one percent noise. The plot of the binary star data with ten percent noise looks very similar to that with one percent noise. In Figure 4 is a plot of the true satellite image and the corresponding data with one percent noise. The plot of the satellite data with ten percent noise appears much less smooth than that with one percent noise.

Let $f_\alpha$ be the global constrained minimizer of problem (3.12). We choose the regularization parameter $\alpha$ to approximately minimize $\|f_\alpha - f_{\text{true}}\|$. For the binary star data, $\alpha = 8 \times 10^{-8}$ for one percent noise and $\alpha = 5 \times 10^{-8}$ for ten percent noise.
Figure 3. Binary Star Object and Image Data. To the left is the upper left hand 64 \times 64 pixels of the true image. To the right is the corresponding data with one percent noise.

Figure 4. Satellite Object and Image Data. To the left is the upper left hand 64 \times 64 pixels of the true image. To the right is the corresponding data with one percent noise.
For the satellite data, \( \alpha = 8 \times 10^{-6} \) for one percent noise and \( \alpha = 3 \times 10^{-4} \) for ten percent noise. One would expect that less noise in the data would correspond to a smaller regularization parameter. The fact that this is not the case for the binary star data can be explained by the fact that the noise is predominantly Poisson, and in order to obtain a lower noise level we increase the intensity of binary star object. This corresponds to an increase in regularization parameter as well. The result is that the regularization parameter for the binary star data with one percent noise is smaller, relatively speaking, than the regularization parameter with ten percent noise. The difference in intensities between the true satellite images has the same effect.

As was noted, a smaller noise level requires a smaller, at least in relative terms, regularization parameter, which results in a more ill-conditioned, and hence, a more difficult minimization problem. On the other hand, given that there is less noise and that the regularization parameter is smaller, more information is available, and hence, we can expect that the corresponding reconstruction to be better.

Figures 5-6 contain plots of highly accurate numerical approximations of the global minimizer \( f_{\alpha} \) obtained from an implementation of GPCG for each data set with the choices of the parameter \( \alpha \) given above. The large differences in scaling between both the binary star images and two satellite images is due to the change in scaling of the true objects mentioned above. Note that the images with higher intensity correspond to the lower noise level.
Figure 5. Binary Star Data Reconstructions. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.

Figure 6. Satellite Data Reconstructions. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.
Figure 7. Solution Error Plots for Binary Star Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise.

We now compare the performance of GPCG and GPCG-FM (see Algorithms 7 and 9 respectively) for solving (3.12)-(3.13). Since the cost of implementing our algorithms is dominated by the number of FFTs used during implementation, each of the convergence plots found in Figures 7-10 gives a measure of error versus the cumulative FFT count. We define the relative (iterative) solution error to be

$$
\frac{||f_k - f_{true}||}{||f_{true}||}
$$

and the (iterative) regularized solution error to be

$$
\frac{||f_k - f_\alpha||}{||f_\alpha||},
$$

where the choices of $\alpha$, which depend upon the data set being used, are given above.
Figure 8. Regularized Solution Error Plots for Binary Star Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise.

Figure 9. Solution Error Plots for Satellite Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denoted the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise.
Figure 10. Regularized Solution Error Plots for Satellite Data with Quadratic Cost Function. The solid line denotes the GPCG iterates. The dashed line denotes the GPCG-FM iterates. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise.

In each figure, the convergence plots corresponding to either the binary star or satellite true image for both one and ten percent noise levels are set beside one another. These figures show that, as was expected, for either choice of true image, the minimization problem is more expensive to solve for the data with one percent noise, and the resulting solution error is lower. We can also see that in all four examples, when the solution error has stabilized, the regularized solution error is far from converging. This suggests that in terms of reconstruction quality, it is not necessary to run a minimization algorithm to convergence in order to receive a relatively good reconstruction of the true image.
We notice that GPCG is superior to GPCG-FM for each of the four minimization problems presented. A natural question is: Why? GPCG and GPCG-FM are similar in that they are both algorithms that have the general framework of Algorithm 3. Furthermore, both algorithms use the gradient projection algorithm in stage 1 and CG in stage 2. As mentioned in Remark 3.12, the ways in which the two algorithms use CG and the gradient projection algorithm are very different. By mixing stage 1 and stage 2 of GPCG and GPCG-FM and implementing the resulting algorithms on several test problems we have come to the conclusion that GPCG uses both the gradient projection algorithm and the CG algorithm in a more efficient way than does GPCG-FM. And, naturally, this results in GPCG being a better algorithm for this particular application. We expect to be able to improve the performance of GPCG by using parallel programming techniques [28].

As can be seen in Figure 11, none of the global constrained minimizers of the functions corresponding to the four data sets under consideration is degenerate (see Definition 2.23), but all are very nearly degenerate. The nearer to degenerate is a minimizer the more difficult is the corresponding minimization problem. Figure 11 also shows us that as the noise level increases from one to ten percent, the global minimizers are nearer to degenerate. Consequently, the increase in difficulty in solving the minimization problem as the noise level decreases is not due to the corresponding solution being nearer to degenerate.
Figure 11. Distribution of Gradient Components for Quadratic Cost Function. The horizontal axis shows the $\log_{10}$ of the size of the gradient components corresponding to the active indices. Note that the data has been binned. The vertical axis shows the number of occurrences for each size. The left hand plot corresponds to the binary star data. The right hand plot corresponds to the satellite data. The solid line corresponds to the data with one percent noise. The dashed line corresponds to the data with ten percent noise.

We note that the test problems with the binary star data are more difficult to solve than those with the satellite data at the corresponding error levels. We conjecture that this is due in large part to the fact that at the optimal face $\mathcal{F}(f^*)$, a much larger number of constraints are active for the binary star data than for the satellite data. Consequently, determining the optimal face is much more difficult in the case of the binary star. This conjecture is supported by our computational experience. With the satellite data we observed that the active set stabilized much earlier in the iterations than it did with the binary star data. On the other hand, once the optimal face is
determined the resulting linear system is much smaller in the case of the binary star, and hence, CG ought to converge more quickly.

Finally, when generating our data we used MATLAB’s random number generator with the initial state set to 0 to add Gaussian noise. We tested the numerical performance of the above algorithms on data that was generated with different initial states and found that at the ten percent noise level the performance of the algorithms was nearly identical in terms of convergence history and in overall cost. At the one percent noise level, when the minimization problem becomes more unstable and regularization parameters are smaller, the convergence history was quantitatively very similar, but the overall cost differed with different initial states.
CHAPTER 4

LOWER BOUND CONstrained CONVEX Programming

In this chapter we present algorithms for solving

$$\min_{x \in \Omega} J(x) \quad (4.1)$$

when $J$ is strictly convex (see Definition 2.2) and coercive (see Definition 2.8), and $\Omega = \{f \in \mathbb{R}^N \mid f \geq 0\}$. This class of functions is important to us for several reasons. First, the coercivity of $J$ guarantees the existence of a global constrained minimizer $x^*$ of $J$ on $\Omega$ (see Theorem 2.9). Secondly, the strict convexity of $J$ guarantees that $x^*$ is the unique local constrained minimizer of $J$ on $\Omega$ (see Theorem 2.22). Hence, if two different algorithms generate sequences that drive $\text{grad}_p J$ to zero, we know that both sequences converge to the same stationary point. On the other hand, if $J$ has more than one local minimizer, two different algorithms may generate sequences that drive $\text{grad}_p J$ to zero, but that converge to different stationary points.

Consequently, by restricting our attention to strictly convex and coercive functions $J$, we can better compare algorithms. Finally, and perhaps most importantly for us, convex minimization is important in imaging applications [42]. We will make the further assumption that the eigenvalues of the Hessian of $J$ have the boundness property given by equation (2.23), and that $J$ is Lipschitz continuous. The function (1.6)-(1.10) we wish to minimize has these properties as well.
Convex Gradient Projection-Conjugate Gradient

Convex Gradient Projection-Conjugate Gradient (C-GPCG) is very similar to GPCG in that it combines the gradient projection method and CG. Given an approximation $\mathbf{x}_k$ to the solution of problem (4.1), we begin by taking one projected gradient iteration via Algorithm 2 to get $\mathbf{x}_{k+1}^{GP}$. Given the assumptions that we have made about the function $\mathcal{J}$, Algorithm 2 is well defined (see Theorem 2.36). We then use CG to create a sequence of iterates $\{\mathbf{d}_j\}$, with initial guess $\mathbf{d}_0 = \mathbf{0}$, that converges in a finite number of steps to a minimizer of the quadratic model

$$q_k(\mathbf{d}) = \frac{1}{2} \langle \text{Hess}_R \mathcal{J}(\mathbf{x}_{k+1}^{GP}) \mathbf{d}, \mathbf{d} \rangle + \langle \mathbf{d}, \text{grad}_I \mathcal{J}(\mathbf{x}_{k+1}^{GP}) \rangle.$$  

(4.2)

Here $\text{grad}_I \mathcal{J}$ is given in Definition 3.11 and $\text{Hess}_R \mathcal{J}$ is given in Definition 2.29.

We terminate CG if

$$j = \text{CG}_{\text{max}},$$  

(4.3)

where $\text{CG}_{\text{max}}$ is an upper bound on the number of CG iterations set in Step 0; or if

$$||\mathbf{r}_j|| < ||\mathbf{r}_0|| \cdot \min \left(\frac{1}{2}, \sqrt{||\mathbf{r}_0||}\right),$$  

(4.4)

where $\mathbf{r}_j = \text{Hess}_R \mathcal{J}(\mathbf{x}_{k+1}^{GP}) \mathbf{d}_j + \text{grad}_I \mathcal{J}(\mathbf{x}_{k+1}^{GP})$ is the gradient of $q_k$ at $\mathbf{d}_j$; or if

$$q_k(\mathbf{d}_{j-1}) - q_k(\mathbf{d}_j) \leq \gamma \max \{q_k(\mathbf{d}_{i-1}) - q_k(\mathbf{d}_i) \mid i = 1, \ldots, j\}.$$  

(4.5)

We take $\gamma = \frac{1}{4}$ in (4.5). Note the similarities between (4.5) and equation (3.3), which is used as a stopping tolerance for CG in GPCG. If (4.5) holds, the implication is
that CG is not making sufficient progress. We then let $p_{k+1}^{GP} = d_k$, where $d_k$ is the first index that satisfies one of (4.3), (4.4), or (4.5), and we perform an inexact line search using Algorithm 2 to find $\lambda_{k+1}^{GP}$. We then set $x_{k+1} = \mathcal{P}_\Omega(x_{k+1}^{GP} + \lambda_{k+1}^{GP} p_{k+1}^{GP})$.

C-GPCG is then given as follows:

**Algorithm 10. (C-GPCG)**

1. **Step 0:** Select initial guess $x_0$, and set $k = 0$.
2. **Step 1:** Given $x_k$.
   
   (1) Take one projected gradient step using Algorithm 2. Return $x_{k+1}^{GP}$.
3. **Step 2:** Given $x_{k+1}^{GP}$.
   
   (1) Do CG iterations to minimize the quadratic (4.2) until (4.3), (4.4), or (4.5) is satisfied. Return $d_k$.
   
   (2) Do a projected line search with $p_{k+1}^{GP} = d_k$. Set $x_{k+1} = x_{k+1}^{GP} + \lambda_{k+1}^{GP} p_{k+1}^{GP}$.
4. **Step 3:** Given $x_{k+1}$.
   
   (1) If the stopping criteria is satisfied, STOP.
   
   (2) Otherwise, update $k := k + 1$ and return to Step 1.

**Remark 4.1.** Recall that in the GPCG algorithm (see Algorithm 7) more than one gradient projection step was allowed in Step 1, while in Step 1 of Algorithm 10 only one gradient projection iteration is taken. For our applications this is cost effective.

**Remark 4.2.** We will use Algorithm 2 to perform the line search in (2) of Step 2, only with a different sufficient decrease condition. In (1) of Algorithm 2 we replace
(2.22) with

\[ \phi_{k+1}^{GP}(\lambda_{k+1}) < \phi_{k+1}^{GP}(0), \]

where

\[ \phi_{k+1}^{GP}(\lambda) = J\left( \mathcal{P}_\Omega(x_{k+1}^{GP} + \lambda x_{k+1}^{GP}) \right). \]

We now must prove that Algorithm 2 with this modification generates a suitable \( \lambda_{k+1}^{GP} \) in finitely many iterations. Since \( J \) is strictly convex, Hess \( J(x) \), and hence \( \text{Hess}_R J(x) \), is SPD for all \( x \in \Omega \) [27]. It is not difficult to show then that if \( d_{j_k} \) is generated by applying CG to minimize (4.2), \( \langle \text{grad} J(x_{k+1}^{GP}), d_{j_k} \rangle < 0 \). Then since

\[ J(x_{k+1}^{GP} + \lambda d_{j_k}) = J(x_{k+1}^{GP}) + \lambda \langle \text{grad} J(x_{k+1}^{GP}), d_{j_k} \rangle + o(\lambda), \]

there exists a \( \delta > 0 \) such that \( J(x_{k+1}^{GP} + \lambda d_{j_k}) < J(x_{k+1}^{GP}) \) for all \( 0 < \lambda < \delta \), and hence, by the arguments in the proof of Theorem 2.36, Algorithm 2 generates a suitable \( \lambda_{k+1}^{GP} \) in finitely many iterations. Step 2 is therefore well defined and, consequently, so is Algorithm 10.

**Remark 4.3.** (Convergence) This algorithm clearly fits the general framework of Algorithm 3. Also, since we are restricting our attention to strictly convex, coercive functions that are Lipschitz continuous, by Corollary 2.44, Algorithm 10 generates a sequence which converges to the unique global constrained minimizer \( x^* \). Furthermore, by Theorem 2.46, Algorithm 10 identifies the optimal \( \mathcal{F}(x^*) \) in finitely many iterations.
Remark 4.4. Since CG is used in C-GPCG, the question of preconditioning naturally arises. Unfortunately, changes in the active set from iteration to iteration make preconditioning difficult. Nonetheless, computational experience as well as theoretical results (see Remark 4.3) show that in later iterations the active set stabilizes. Once this occurs, preconditioning strategies are more likely to be effective.

One possibility is to use limited memory BFGS matrices in a fashion similar to that suggested in [33] as preconditioners for CG. We do this by saving the vectors $s_k^{GP} := x_{k+1}^{GP} - x_k$ and $y_k^{GP} := \text{grad } J(x_{k+1}^{GP}) - \text{grad } J(x_k)$ after (1) of Step 1, and $s_k^{CG} := x_{k+1} - x_k^{GP}$ and $y_k^{CG} := \text{grad } J(x_{k+1}) - \text{grad } J(x_{k+1}^{GP})$ after (3) of Step 2. We then use L-BFGS recursion (2.42) to compute the preconditioning step within preconditioned CG at each iteration. This requires that $\langle s_k^{GP}, y_k^{GP} \rangle > 0$ and $\langle s_k^{CG}, y_k^{CG} \rangle > 0$ hold for every $k$. Note that

$$\langle s_k^{GP}, y_k^{GP} \rangle = \langle s_k^{GP}, \text{grad } J(x_{k+1}^{GP}) - \text{grad } J(x_k) \rangle$$

$$= \langle s_k^{GP}, \text{Hess } J(x_k + ts_k^{GP})s_k^{GP} \rangle$$

$$> 0.$$

The second equality follows from the mean value theorem for the function $h(t) = \langle s_k^{GP}, \text{grad } J(x_k + ts_k^{GP}) \rangle$ for some $t \in (0, 1)$. The last inequality follows from the fact that since $J$ is strictly convex, Hess $J$ is positive definite [27]. The proof that $\langle s_k^{CG}, y_k^{CG} \rangle > 0$ is the same.
Remark 4.5. We explored a modification to Algorithm 10 in which CG was replaced in Step 2 with the BFGS algorithm (see Algorithm 6). This algorithm, though effective for certain applications, was not effective on our test problems.

Numerical Results

In this section we solve

$$\min_{r \geq 0} J_\alpha(f),$$

(4.7)

where $J_\alpha$ was introduced in Chapter 1 and is given by

$$J_\alpha(f) = \sum_i ([Sf]_i + \sigma^2) - \sum_i d_i \log([Sf]_i + \sigma^2) + \frac{\alpha}{2} f^T L f.$$  

(4.8)

As in the previous chapter, $d$ and $f$ are $(n_x \cdot n_y) \times 1$ vectors and $S$ is the $(n_x \cdot n_y) \times (n_x \cdot n_y)$ BTBB matrix $S$ from the image formation model (1.4) (see previous chapter for details). We take $L$ to be either the identity matrix or the negative of the discrete Laplacian matrix with Neumann boundary conditions. In Chapter 1, we established that $J_\alpha$ is convex and Lipschitz continuous on $\Omega = \{ f \in \mathbb{R}^N \mid f \geq 0 \}$, and is strictly convex provided that the null spaces of the matrices $L$ and $S$ intersect only at $\{0\}$. This trivially holds for $L = I$. For $L$ equal to the negative Laplacian, we note that $L$ has a one dimensional subspace consisting of the constant vectors. Then, since the $S$ we use for our applications does not annihilate constant vectors, the null spaces of $S$ and $L$ intersect only at $\{0\}$. Consequently, for either choice of $L$, $J_\alpha$ is strictly convex. We also know that the eigenvalues of Hess $J$ have the bound property given
by equation (2.23). Hence, we can apply Algorithm 10 to solve problem (4.7), and
the results stated in Remark 4.3 will hold.

We use the same four data sets that were used in Chapter 3 (see Figures 2-4).
The phase and PSF used to generate our data is displayed in Figure 2. We define the
percent noise by (3.15).

Let \( f_\alpha \) be the global constrained minimizer of problem (4.7). We choose the
regularization parameter \( \alpha \) to approximately minimize \( \| f_\alpha - f_\text{true} \| \). For the binary
star data with identity regularization, \( \alpha = 4 \times 10^{-12} \) for one percent noise and \( \alpha = 2 \times 10^{-9} \) for ten percent noise. For the binary star data with Laplacian regularization,
\( \alpha = 1 \times 10^{-12} \) for one percent noise and \( \alpha = 4 \times 10^{-10} \) for ten percent noise. For the satellite data with identity regularization, \( \alpha = 1 \times 10^{-9} \) for one percent noise and
\( \alpha = 2 \times 10^{-6} \) for ten percent noise. For the satellite data with Laplacian regularization,
\( \alpha = 3 \times 10^{-10} \) for one percent noise and \( \alpha = 1 \times 10^{-6} \) for ten percent noise. Note that
the regularization parameter sizes corresponding to each data set are smaller than in
the quadratic case (see Chapter 3).

The reconstructions corresponding to identity regularization are nearly indistin-
guishable from those found in Figures 5 and 6 of Chapter 3. Taking \( L \) to be the
negative Laplacian penalizes vectors \( f \) with large discretized gradients, while \( L = I \)
does not. Consequently, we might expect the corresponding reconstructions to be
different. Figures 12 and 13 show highly accurate reconstructions for \( L \) equal to the
negative Laplacian. We see little difference between the binary star reconstructions
Figure 12. Binary Star Data Reconstructions with Negative Laplacian Regularization. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.

and those found in Figure 5. This may be due to the fact that the true image has very few nonzero elements and that the regularization parameters are small enough that vectors $\mathbf{f}$ with large discretized gradients see little penalization. A small value of $\alpha$ may also account for the fact that we see little difference between the reconstruction of the satellite data with one percent noise and the analogous reconstruction in Figure 6. This observation is supported by the fact that the reconstruction of the satellite data with ten percent noise, where $\alpha$ is substantially larger, is somewhat smoother than that found in Figure 6.

We now compare the performance of C-GPCG, preconditioned C-GPCG, the bound constrained variant of the limited memory BFGS algorithm [32] known as
Figure 13. Satellite Data Reconstructions with Negative Laplacian Regularization. On the left is a mesh plot of the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with one percent noise. On the right is a mesh plot of the the upper left $64 \times 64$ pixels of the reconstruction corresponding to the data with ten percent noise. Notice the difference in scaling between these two images.

LBFGS-B \cite{nocedal1980, byrd1995}, and the interior point method KNITRO \cite{byrd1999}. Preliminary numerical comparisons have shown that MATLAB’s large-scale bound constrained minimizer \cite{weintraub2003, gould2015} performs poorly on our applications, and hence, we do not include it in the comparisons. We also found that though KNITRO performs substantially better than MATLAB’s minimizer, it is not competitive with LBFGS-B or either of the methods presented in this chapter for solving our test problems. We therefore include KNITRO only in the comparisons with the binary star data with ten percent noise. For the sake of simplicity, in each of Figures 14-19 we save five BFGS vectors for the preconditioning matrix in preconditioned C-GPCG, and we save ten BFGS vectors for LBFGS-B. These appear to be the best choices for overall performance. Our
experience shows that though a slightly larger or smaller number of saved vectors yields minor improvements in performance for a particular test problem, it is not enough of an improvement to warrant changes in the numbers of saved vectors with each test problem. We set $CG_{max} = 10$ in both C-GPCG and preconditioned C-GPCG for each data set, with the exception of the binary star with one percent noise, in which $CG_{max} = 20$.

In each test case, negative Laplacian regularization seems to make little difference in terms of minimizing both the solution error and the computational cost. This is in addition to the fact that the corresponding reconstructions varied only slightly from those gotten with identity regularization. This may be due to the fact that neither true image is smooth. We conclude that negative Laplacian regularization is not appropriate for these particular test problems. We therefore only include plots with negative Laplacian regularization for the binary star data.

The convergence plots we present in Figures 14-19 are analogous to those found in Chapter 3. These figures show that for either choice of true image, the minimization problem is more expensive to solve for the data with one percent noise, and the resulting solution error is lower. We can also see that in each test problem, when the solution error has stabilized, the regularized solution error is far from converging. Hence, in terms of reconstruction quality, it is not necessary to iterate these minimization algorithms to convergence in order to receive a relatively good reconstruction of
Figure 14. Solution Error for Convex Cost Function with Identity Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO.

Figure 15. Regularized Solution Error for Convex Cost Function with Identity Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO.
Figure 16. Solution Error for Convex Cost Function with Negative Laplacian Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO.

Figure 17. Regularized Solution Error for Convex Cost Function with Negative Laplacian Regularization with Binary Star Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B. The line with circles in the right-hand plot denotes KNITRO.
Figure 18. Solution Error for Convex Cost Function with Identity Regularization with Satellite Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B.

Figure 19. Solution Error for Convex Cost Function with Identity Regularization with Satellite Data. The plot on the left shows results for one percent noise. The plot on the right shows results for ten percent noise. The solid line denotes preconditioned C-GPCG. The dashdot line denotes C-GPCG without preconditioning. The dashed line denotes LBFGS-B.
the true image. These findings are consistent with what we saw for quadratic $J_a$ (see Chapter 3).

With the exception of the solution error plots for the satellite data with ten percent noise found in Figures 19, in which C-GPCG without preconditioning minimizes the solution error with the least amount of cost, preconditioned C-GPCG is the superior algorithm. This can be seen most clearly in each of the regularized solution error plots and in the solution error plots for the binary star data with one percent noise. It seems that as the problems become more difficult, preconditioned C-GPCG is less effective. In our numerical simulations, we noticed that for the more difficult problems, the active at the solution is identified, or nearly so, long before the algorithm converges. Once this occurs, preconditioning strategies begin to work well. For the easier problems, the algorithms converge almost immediately after the active set at the solution is identified. Consequently, the preconditioning strategies are not given a chance to work.

It is natural to ask whether, for these test problems, it is better to minimize the quadratic function, as was done in Chapter 3, or the convex function presented in this section. Comparing Figures 14-19 with Figures 7-10 we see that although using $J_a$ defined by (4.8) yields slightly more accurate reconstructions, the cost in solving the minimization problem is much higher, and this is true for each test case. Minimizing the quadratic function using GPCG is therefore recommended.
Figure 20. Distribution of Gradient Components for Convex Cost Function with Identity Regularization. The horizontal axis shows the log_{10} of the size of the gradient components corresponding to the active indices. Note that the data has been binned. The vertical axis shows the number of occurrences for each size. The left hand plot corresponds to the binary star data. The right hand plot corresponds to the satellite data. The solid line corresponds to the data with one percent noise. The dashed line corresponds to the data with ten percent noise.

Figure 20 shows that none of the global constrained minimizers of the functions corresponding to the four data sets under consideration is degenerate (see Definition 2.23), but all are very nearly degenerate. We also see that the plots are very similar for different noise levels. Consequently, the increase in difficulty in solving the minimization problems as the noise level decreases is not due to the corresponding solutions being nearer to degenerate. Furthermore, if we compare Figure 20 with the degeneracy plots for the quadratic function given in Figure 11 (see Chapter 3) we see very similar plots. Thus, we conclude that the increase in difficulty in solving
the minimization problems in this section versus those in Chapter 3 is not due to corresponding solutions being nearer to degenerate.

Finally, we note that, just as in the quadratic case, the test problems with the binary star data are more difficult to solve than those with the satellite data at the corresponding error levels. Our conjecture as to why this is the case can be found at the end of Chapter 3.
CHAPTER 5

CONCLUSION

We have developed a two stage algorithm for minimizing a strictly convex function, and have shown that it is competitive with other existing algorithms. We have also shown that preconditioning strategies will work for bound constrained minimization problems.

The algorithm presented in Chapter 4 can be extended to the problem of minimizing nonconvex functions over a constraint space, and similar two stage algorithms are possible. We would like to implement a two stage algorithm for solving the minimization problem given by (1.17). We have developed a convergent two stage algorithm for this problem, but work still needs to be done. In particular, preconditioning strategies similar to those used in Chapter 4 need to be explored.

In this thesis we have concentrated on solving what are known as variational regularization problems [42]. An alternative approach is to use what are known as iterative regularization algorithms, which use iteration count as a regularization parameter [42]. A comparison between these two approaches needs to be done. Preliminary results seem to suggest that the algorithms we have implemented and developed are competitive with the iterative regularization algorithms in terms of the cost of implementation.
REFERENCES CITED


