A Levenberg-Marquardt Method For Large-Scale
Bound-Constrained Nonlinear Least-Squares

by

Shidong Shan

BSc (Hon.), Acadia University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University of British Columbia
(Vancouver)

July 2008
© Shidong Shan 2008
Abstract

The well known Levenberg-Marquardt method is used extensively for solving nonlinear least-squares problems. We describe an extension of the Levenberg-Marquardt method to problems with bound constraints on the variables. Each iteration of our algorithm approximately solves a linear least-squares problem subject to the original bound constraints. Our approach is especially suited to large-scale problems whose functions are expensive to compute; only matrix-vector products with the Jacobian are required. We present the results of numerical experiments that illustrate the effectiveness of the approach. Moreover, we describe its application to a practical curve fitting problem in fluorescence optical imaging.
Contents

Abstract .................................................. ii
Contents .................................................. iii
List of Tables ............................................ v
List of Figures ........................................... vi
Acknowledgements ....................................... vii

1 Introduction ........................................... 1
  1.1 Overview ........................................... 2
  1.2 Definitions and notation ........................... 3

2 Background on Optimization .......................... 5
  2.1 Unconstrained optimization ........................ 5
     2.1.1 Line-search methods .......................... 6
     2.1.2 Trust-region methods ........................ 6
  2.2 Bound-constrained optimization ................... 8
     2.2.1 Active-set methods ........................... 8
     2.2.2 Gradient projection methods ................... 9
  2.3 Algorithms for bound-constrained optimization .... 10

3 Unconstrained Least Squares ......................... 12
  3.1 Linear least squares ............................... 12
     3.1.1 Normal equations ............................. 12
     3.1.2 Two norm regularization ........................ 13
  3.2 Nonlinear least squares ........................... 14
     3.2.1 The Gauss-Newton method ......................... 14
     3.2.2 The Levenberg-Marquardt method .................. 16
     3.2.3 Updating the damping parameter ................. 17

4 Nonlinear Least Squares With Simple Bounds ........ 20
  4.1 Motivation .......................................... 20
  4.2 Framework .......................................... 21
  4.3 Stabilization strategy ............................. 24
  4.4 The subproblem solver ............................. 25
<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 The BCNLS algorithm</td>
<td>27</td>
</tr>
<tr>
<td>5 Numerical Experiments</td>
<td>30</td>
</tr>
<tr>
<td>5.1 Performance profile</td>
<td>30</td>
</tr>
<tr>
<td>5.2 Termination criteria</td>
<td>31</td>
</tr>
<tr>
<td>5.3 Benchmark tests with Algorithm 566</td>
<td>31</td>
</tr>
<tr>
<td>5.4 CUTEr feasibility problems</td>
<td>34</td>
</tr>
<tr>
<td>6 Curve Fitting in Fluorescence Optical Imaging</td>
<td>38</td>
</tr>
<tr>
<td>6.1 Brief introduction to time domain optical imaging</td>
<td>38</td>
</tr>
<tr>
<td>6.2 Mathematical models of fluorescence signals</td>
<td>40</td>
</tr>
<tr>
<td>6.3 Curve fitting</td>
<td>41</td>
</tr>
<tr>
<td>6.4 Fluorescence lifetime estimation</td>
<td>42</td>
</tr>
<tr>
<td>6.4.1 Fitting parameters</td>
<td>42</td>
</tr>
<tr>
<td>6.4.2 Parameter bounds</td>
<td>43</td>
</tr>
<tr>
<td>6.4.3 Simulated data for lifetime fitting</td>
<td>43</td>
</tr>
<tr>
<td>6.4.4 Experimental results</td>
<td>43</td>
</tr>
<tr>
<td>6.5 Fluorophore depth estimation</td>
<td>45</td>
</tr>
<tr>
<td>6.6 Optical properties estimation</td>
<td>48</td>
</tr>
<tr>
<td>6.6.1 Simulation for estimating optical properties</td>
<td>49</td>
</tr>
<tr>
<td>6.6.2 Results for optical properties</td>
<td>50</td>
</tr>
<tr>
<td>6.7 Discussion</td>
<td>50</td>
</tr>
<tr>
<td>7 Conclusions and Future Work</td>
<td>52</td>
</tr>
<tr>
<td>Bibliography</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Results of fifteen benchmark tests from Algorithm 566 ........ 32
5.2 Summary of the benchmark tests from Algorithm 566 ....... 33
5.3 Summary of CUTEr feasibility tests ...................... 36

6.1 Comparison of lifetime fitting with and without DC component .. 47
6.2 Comparison of lifetime fitting using BCNLS and ART.LM ........ 48
6.3 Comparison of depth fitting results for (A) all data points, (B)
data points with depth > 1mm, (C) data points with SNR > 20,
(D) data points with SNR > 20 and depth > 1mm ............... 48
6.4 Comparison of depth fitting results for BCNLS and ART.LM for
data points with SNR > 20 and depth > 1mm .................. 49
6.5 $\mu_a$ and $\mu'_a$ values used in simulation ....................... 50
6.6 Comparison of results of optical properties with ART.LM and
BCNLS .................................................. 50
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Geometrical interpretation of least squares</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>The image of function ( q(\rho) )</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>Performance profiles on benchmark problems from Algorithm 566: number of function evaluations</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Performance profiles on benchmark problems from Algorithm 566: CPU time</td>
<td>35</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance profile on CUTEr feasibility problems, number of function evaluations</td>
<td>37</td>
</tr>
<tr>
<td>5.4</td>
<td>Performance profile on CUTEr feasibility problems, CPU time</td>
<td>37</td>
</tr>
<tr>
<td>6.1</td>
<td>Time domain optical imaging</td>
<td>39</td>
</tr>
<tr>
<td>6.2</td>
<td>Fluorescent decay curve</td>
<td>39</td>
</tr>
<tr>
<td>6.3</td>
<td>Fluorophore identification with lifetime</td>
<td>40</td>
</tr>
<tr>
<td>6.4</td>
<td>An example of fitted TPSF curve</td>
<td>44</td>
</tr>
<tr>
<td>6.5</td>
<td>Lifetime fitting results of simulated data with varying depth, without fitting DC component</td>
<td>45</td>
</tr>
<tr>
<td>6.6</td>
<td>Lifetime fitting results of simulated data with varying depth, with fitting DC component</td>
<td>46</td>
</tr>
<tr>
<td>6.7</td>
<td>Estimated depth versus true depth</td>
<td>47</td>
</tr>
<tr>
<td>6.8</td>
<td>Relative accuracy of calculated depth versus signal SNR</td>
<td>49</td>
</tr>
</tbody>
</table>
Acknowledgements

I am grateful to my supervisor Dr. Michael Friedlander, who made the subject of Numerical Optimization exciting and beautiful. With his knowledge, guidance, and assistance, I have managed to accomplish this work.

I would like to thank Dr. Ian Mitchell for his careful reading of this thesis and his helpful comments that have improved this thesis.

I am fortunate to have had the financial support for this work from the Natural Sciences and Engineering Research Council of Canada (NSERC) in the form of a postgraduate scholarship. Also, I am thankful for the MITACS Accelerate BC internship opportunity at ART Advanced Research Technologies Inc.

It has been a privilege to work with Guobin Ma, Nicolas Robitaille, and Simon Fortier in the Research and Development group at ART Advanced Research Technologies Inc. Their work and ideas on curve fitting in fluorescence optical imaging have formed a strong foundation for Chapter 6 in this thesis. In addition, I wish to thank the three of them for reading a draft of Chapter 6 and their valuable comments and suggestions for improving the chapter.

The Scientific Computing Laboratory at the University of British Columbia has been a pleasant place to work. I have enjoyed the support and company of my current and past colleagues: Ewout van den Berg, Elizabeth Cross, Marisol Flores Garrido, Hui Huang, Dan Li, Tracy Lau, and Jelena Sirovljevic. Among them, Ewout van den Berg has always been of great help to me.

I am deeply indebted to my parents and family for their unconditional love and understanding. My brother Hongqing has inspired me and set a great example for me to fulfill my dreams. Thanks are also due to many of my friends for their care and friendship over the years.

Above all, I sincerely thank my best friend and partner Bill Hue, whose love, support, and faithful prayers give me strength and courage to cope in difficult times. Without him, I could have not achieved this far in my academic endeavor.

Finally, I want to dedicate this thesis to my late father, who suddenly passed away while I was working on this thesis. In his life, my father was always passionate that his children would have every educational opportunity. There is no doubt that my father would be very proud if he could live to witness another academic accomplishment of his youngest daughter.
To my father, Kunsong Shan
Chapter 1

Introduction

This thesis proposes an algorithm for nonlinear least-squares problems subject to simple bound constraints on the variables. The problem has the general form

\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \|r(x)\|_2^2
\]

subject to \( \ell \leq x \leq u, \)

where \( \ell, u \in \mathbb{R}^n \) are vectors of lower and upper bounds on \( x, \) and \( r \) is a vector of real-valued nonlinear functions

\[
r(x) = \begin{bmatrix}
    r_1(x) \\
r_2(x) \\
    \vdots \\
r_m(x)
\end{bmatrix}
\]

we assume throughout that each \( r_i(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is a twice continuously differentiable nonlinear function.

Nonlinear least-squares problems arise in many data-fitting applications in science and engineering. For instance, suppose that we model some process by a nonlinear function \( \phi \) that depends on some parameters \( x, \) and that we have the actual measurements \( y_i \) at time \( t_i; \) our aim is to determine parameters \( x \) so that the discrepancy between the predicted and observed measurements is minimized. The discrepancy between the predicted and observed data can be expressed as

\[
r_i(x) = \phi(x, t_i) - y_i.
\]

Then, the parameters \( x \) can be estimated by solving the nonlinear least-squares problem

\[
\min_{x} \frac{1}{2} \|r(x)\|_2^2.
\]  

(1.1)

Very large nonlinear least-squares problems arise in numerous areas of applications, such as medical imaging, tomography, geophysics, and economics. In many instances, both the number of variables and the number of residuals are large. It is also very common that only the number of residuals is large. Many approaches exist for the solution of linear and nonlinear least-squares problems, especially for the case where the problems are unconstrained. See Bjorck [3] for a comprehensive discussion of algorithms for least squares, including detailed error analyses.
Because of the high demand in the industry, nonlinear least-squares software is fairly prevalent. Major numerical software libraries such as SAS, and programming environments such as Mathematica and Matlab, contain nonlinear least-squares implementations. Other high-quality implementations include, for example, DFNLP [50], MINPACK [38, 41], and NL2SOL [27]; see Moré and Wright [44, Chapter 3]. However, most research has focused on the nonlinear least-squares problems without constraints. For problems with constraints, the approaches are fewer (also see Bjorck [3]). In practice, the variables $x$ often have some known lower and upper bounds from its physical or natural settings. The aim of this thesis is to develop an efficient solver for large-scale nonlinear least-squares problems with simple bounds on variables.

### 1.1 Overview

The organization of the thesis is as follows. In the remainder of this chapter, we introduce some definitions and notation that we use in this thesis. Chapter 2 reviews some theoretical background on unconstrained and bound-constrained optimization. On unconstrained optimization, we summarize the basic strategies such as line-search and trust-region methods. For bound-constrained optimization, we review active-set and gradient-projection methods. We also briefly discuss some existing algorithms for solving the bound-constrained optimization problems, including ASTRAL [53], BCLS [18], and L-BFGS-B [57].

Chapter 3 focuses on unconstrained least-squares problems. We start with a brief review of linear least-squares problems and the normal equations. We also review two-norm regularization techniques for solving ill-conditioned linear systems. For nonlinear least-squares problems, we discuss two classical nonlinear least-squares algorithms: the Gauss-Newton method and the Levenberg-Marquardt method. We also explain the self-adaptive updating strategy for the damping parameter in the Levenberg-Marquardt method. These two methods and the updating strategy are closely related to the proposed algorithm in this thesis.

Chapter 4 is the main contribution of this thesis. We explain our proposed algorithm, named BCNLS, for solving the bound-constrained nonlinear least-squares problems. We describe our algorithm within the ASTRAL [53] framework. The BCNLS algorithm is based on solving a sequence of linear least-squares subproblems; the bounds in the subproblem are copied verbatim from the original nonlinear problem. The BCLS least-squares solver forms the computational kernel of the approach.

The results of numerical experiments are presented in Chapter 5. In Chapter 6, we illustrate one real data-fitting application arising in fluorescence optical imaging. The BCNLS software package and its documentation are available at http://www.cs.ubc.ca/labs/scl/bcnls/.
1.2 Definitions and notation

In this section, we briefly define the notation used throughout the thesis.

The gradient of the function \( f(x) \) is given by the \( n \)-vector

\[
\mathbf{g}(x) = \nabla f(x) = \begin{bmatrix}
\frac{\partial f(x)}{\partial x_1} \\
\frac{\partial f(x)}{\partial x_2} \\
\vdots \\
\frac{\partial f(x)}{\partial x_n}
\end{bmatrix},
\]

and the Hessian is given by

\[
H(x) = \nabla^2 f(x) = \begin{bmatrix}
\frac{\partial^2 f(x)}{\partial^2 x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial^2 x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial^2 x_n}
\end{bmatrix}.
\]

The Jacobian of \( r(x) \) is the \( m \)-by-\( n \) matrix

\[
J(x) = \nabla r(x)^T = \begin{bmatrix}
\nabla r_1(x)^T \\
\nabla r_2(x)^T \\
\vdots \\
\nabla r_m(x)^T
\end{bmatrix}.
\]

For nonlinear least-squares problems, the objective function has the special form

\[
f(x) = \frac{1}{2} r(x)^T r(x),
\]

and, in this case,

\[
g(x) = J(x)^T r(x),
\]

\[
H(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x).
\]

Most specialized algorithms for nonlinear least-squares problems exploit the special structure of the nonlinear least-squares objective function (1.2). Our proposed algorithm also explores this structure and makes use of the above structural relations.

We use the following abbreviations throughout:

- \( x_k \): the value of \( x \) at iteration \( k \);
- \( x^* \): a local minimizer of \( f \);
- \( \ell_k, u_k \): the lower and upper bounds for the subproblem at iteration \( k \).
- $r_k = r(x_k)$, vector values of $r$ at $x_k$;
- $f_k = f(x_k)$, function value of $f$ at $x_k$;
- $J_k = J(x_k)$, Jacobian of $r(x)$ at $x_k$;
- $g_k = g(x_k)$, gradient of $f(x)$ at $x_k$;
- $H_k = H(x_k)$, Hessian of $f(x)$ at $x_k$. 
Chapter 2

Background on Optimization

This chapter reviews some fundamental techniques for unconstrained and bound-constrained optimization. We begin with some basic strategies for unconstrained optimization problems, then we examine some specialized methods for bound-constrained problems.

2.1 Unconstrained optimization

The general unconstrained optimization problem has the simple mathematical formulation

$$\minimize_{x \in \mathbb{R}^n} f(x).$$

(2.1)

Hereafter we assume that $f(x) : \mathbb{R}^n \to \mathbb{R}$ is a smooth real-valued function. By Taylor's theorem, a smooth function can be approximated by a quadratic model in some neighbourhood of each point $x \in \mathbb{R}^n$.

The optimality conditions for unconstrained optimization problem (2.1) are well-known. We summarize the conditions below without explanation. For more detailed descriptions, see Nocedal and Wright [47, Chapter 2].

First-Order Necessary Conditions: If $x^*$ is a local minimizer and $f$ is continuously differentiable in an open neighbourhood of $x^*$, then $g(x^*) = 0$.

Second-Order Necessary Conditions: If $x^*$ is a local minimizer of $f$ and $\nabla^2 f$ exists and is continuous in an open neighbourhood of $x^*$, then $g(x^*) = 0$ and $H(x^*)$ is positive semidefinite.

Second-Order Sufficient Conditions: Suppose that $\nabla^2 f$ is continuous in an open neighbourhood of $x^*$, $g(x^*) = 0$, and $H(x^*)$ is positive definite. Then $x^*$ is a strict local minimizer of $f$.

In general, algorithms for unconstrained optimization generate a sequence of iterates $\{x_k\}_{k=0}^\infty$ using the rule

$$x_{k+1} = x_k + d_k,$$

where $d_k$ is a direction of descent. The two main algorithmic strategies are based on different subproblems for computing the updates $d_k$. 
2.1.1 Line-search methods

Line search methods obtain a descent direction $d_k$ in two stages. First, $d_k$ is found such that
\[ g_k^T d_k < 0. \tag{2.2} \]
That is, it requires that $f$ is strictly decreasing along $d_k$ within some neighbourhood of $x_k$. The most obvious choice for search direction is $d_k = -g_k$, which is called the **steepest-descent direction**. However, any direction $d_k$ that solves the equations
\[ B_k d_k = -g_k \]
for some positive definite matrix $B_k$ also satisfies (2.2).

Second, the distance to move along $d_k$ is determined by a step length $\alpha_k$ that approximately solves the one-dimensional minimization problem
\[
\min_{\alpha > 0} f(x_k + \alpha d_k).
\]
Popular line search conditions require that $\alpha_k$ gives **sufficient decrease** in $f$ and ensures that the algorithm makes reasonable progress. For example, the **Armijo condition** imposes sufficient decrease by requiring that
\[ f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma_1 \alpha_k g_k^T d_k, \tag{2.3} \]
for some constant $\sigma_1 \in (0, 1)$. The **curvature condition** imposes “reasonable progress” by requiring that $\alpha_k$ satisfies
\[ g(x_k + \alpha_k d_k)^T d_k \geq \sigma_2 g_k^T d_k, \tag{2.4} \]
for $\sigma_2 \in (\sigma_1, 1)$. The sufficient decrease (2.3) and curvature conditions (2.4) collectively are known as the **Wolfe conditions**. The **strong Wolfe conditions** strengthen the curvature condition and require that, in addition to (2.3), $\alpha_k$ satisfies
\[ |g(x_k + \alpha_k d_k)^T d_k| \leq \sigma_2 |g_k^T d_k|. \]

2.1.2 Trust-region methods

Trust-region methods were first developed for nonlinear least-squares problems [32, 37, 45]. When $f(x)$ is twice continuously differentiable, Taylor’s theorem gives
\[ f(x + d) = f + g^T d + \frac{1}{2} d^T H d + O(\|d\|^3). \]
Thus the quadratic model function $m_k$ used at each iterate $x_k$ is
\[ m_k(d) = f_k + g_k^T d + \frac{1}{2} d^T H_k d. \tag{2.5} \]

The fundamental idea of trust-region methods is to define a trust-region radius $\Delta_k$ for each subproblem. At each iteration, we seek a solution $d_k$ of the
subproblem based on the quadratic model (2.5) subject to some trusted region:

$$\begin{align*}
\text{minimize} & \quad m_k(d) \\
\text{subject to} & \quad \|d\|_2 \leq \Delta_k.
\end{align*}$$

(2.6)

One of the key ingredients in a trust-region algorithm is the strategy for choosing the trust-region radius $\Delta_k$ at each iteration. Given a step $d_k$, we define the quantity

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{m_k(0) - m_k(d_k)},$$

(2.7)

which gives the ratio between the actual reduction and predicted reduction. The actual reduction represents the actual decrease of the objective function for the trial step $d_k$. The predicted reduction in the denominator of (2.7) is the reduction predicted by the model function $m_k$. The choice of $\Delta_k$ is at least partially determined by the ratio $\rho_k$ at previous iterations. Note that the predicted reduction should always be nonnegative. If $\rho_k$ is negative, the new objective value $f(x_k + d_k)$ is greater than the current value $f_k$, so the step must be rejected. If $\rho_k$ is close to 1, there is good agreement between the model and the function over the step, so it is safe to expand the trust region for the next iteration. If $\rho_k$ is positive but significantly smaller than 1, we do not alter the trust region, but if it is close to zero or negative, we shrink the trust region $\Delta_k$ at next iteration.

Given some constants $\eta_1, \eta_2, \gamma_1,$ and $\gamma_2$ that satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{and} \quad 0 < \gamma_1 \leq \gamma_2 < 1,$$

we can update the trust-region radius by

$$\Delta_k+1 \in \begin{cases} 
(\Delta_k, \infty), & \text{if } \rho_k > \eta_2; \\
[\gamma_2 \Delta_k, \Delta_k], & \text{if } \rho_k \in [\eta_1, \eta_2]; \\
(\gamma_1 \Delta_k, \gamma_2 \Delta_k), & \text{if } \rho_k < \eta_1.
\end{cases}$$

(2.8)

See, e.g. Conn, Gould and Toint [9, Chapter 7]. Note that (2.6) can be solved equivalently as

$$\begin{align*}
\text{minimize} & \quad g_k^T d + \frac{1}{2} d^T (H_k + \lambda_k I) d,
\end{align*}$$

for some positive $\lambda_k$ that is larger than the leftmost eigenvalue of $H_k$. Thus, the solution satisfies

$$\begin{align*}
(H_k + \lambda_k I) d &= -g_k.
\end{align*}$$

(2.9)

The relationship between (2.6) and (2.9) is used in Chapter 4 when we describe the BCNLS algorithm. For a summary of trust-region methods, see Nocedal and Wright [47, Chapter 4].
2.2 Bound-constrained optimization

Bound constrained optimization problems have the general form

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad \ell \leq x \leq u,
\end{align*}
\]

where \( \ell, u \in \mathbb{R}^n \) are lower and upper bounds on the variables \( x \). The feasible region is often called a “box” due to its rectangular shape. Note that both the lower and upper bounds may be optional, and when some components of \( x \) lack an upper or a lower bound, we set the appropriate components of \( \ell \) or \( u \) to \(-\infty\) or \(+\infty\), respectively.

We define the set of binding constraints \([20, 53]\) by

\[
B(x^*) = \left\{ i \mid \begin{array}{l}
x_i^* = \ell_i, \quad \text{if } g(x^*)_i \geq 0, \\
x_i^* = u_i, \quad \text{if } g(x^*)_i \leq 0, \\
i = u_i.
\end{array} \right\}
\]

Furthermore, the strictly binding set is defined as

\[
B_s(x^*) = B(x^*) \cap \{ i : g(x^*)_i \neq 0 \}.
\]

The standard first-order necessary condition for a local minimizer \( x^* \) requires

\[
g(x^*)_i = 0 \quad \text{for } i \notin B(x^*). \tag{2.11}
\]

The expression (2.11) shows the importance of the binding constraints to the bound-constrained optimization problems. The first-order necessary condition is also called the first-order Karush-Kuhn-Tucker (KKT) condition. The second-order sufficient condition for \( x^* \) to be a local minimizer of the bound-constrained problem (2.10) is that it satisfies the KKT condition and also

\[
w^T H(x^*) w > 0,
\]

for all vectors \( w \neq 0 \) such that \( w_i = 0 \) for each \( i \in B_s(x^*) \). See [47, Chapter 12] for the detailed theory of constrained optimization.

2.2.1 Active-set methods

Active-set methods for constrained optimization partition the variables into active and inactive sets. Those variables whose indices are in the binding set are also classified as fixed variables, while the variables whose indices are not active are classified as free variables.

Given any set of free variables, we can define the reduced gradient and the reduced Hessian matrix, respectively, as the gradient and the Hessian matrix of \( f(x) \) with respect to the free variables. In this terminology, the second-order sufficient condition requires that the reduced gradient be zero and that the reduced Hessian matrix be positive definite at \( x^* \).
In general, active-set algorithms for the solution of bound-constrained problems use unconstrained minimization techniques to explore the reduced problem defined by a set of free variables. Once this exploration is complete, a new set of free variables is chosen with the aim of driving the reduced gradient to zero.

One drawback of the active-set method is that the working set changes slowly [47]. Usually at each iteration at most one constraint is added to or dropped from the working set. Thus the active-set method may require many iterations to converge on large-scale problems. For example, if there are \( k_0 \) active constraints at the starting point \( x_0 \), and \( k_s \) constraints are active at the solution, then at least \( |k_s - k_0| \) iterations will be required to reach the solution.

2.2.2 Gradient projection methods

A gradient-projection method is a special type of active-set method that allows the active set to change rapidly from iteration to iteration. It is a very efficient method when there is an effective method for the projection onto the feasible set. For example, projection onto bound constraints can be done with linear cost. However, there are also other sets that can be efficiently projected onto, see, e.g., Conn, Gould, and Toint [9, Chapter 12].

Gradient projection methods can be used to minimize both convex and non-convex functions; the feasible set, however, must be convex. One popular approach to the gradient projection methods is to implement each iteration by a two-stage process. First, we compute an active face of the box by searching along the steepest descent direction \(-g\) from the current point \( x \), and we compute a generalized Cauchy point by searching along the piecewise-linear path in the feasible region. The working set is then defined to be the set of bound constraints that are active at the Cauchy point. In the second stage, we compute a search direction using the active face by solving a subproblem in which the active components of \( x \) are in the working set. This approach is implemented by BCLS [18], TRON [33], Lancelot [8], L-BFGS-B [57], and some others.

Given an arbitrary point \( x \), the projection of \( x \) onto the feasible bounded region is defined as follows. The \( i \)th component of the projection of \( x \) is given by

\[
P(x, \ell, u)_i = \begin{cases} 
\ell_i & \text{if } x_i < \ell_i, \\
x_i & \text{if } x_i \in [\ell_i, u_i], \\
u_i & \text{if } x_i > u_i.
\end{cases}
\]

The projected gradient of the objective function \( f \) at a point \( x \) onto the feasible region is defined by

\[
\tilde{g}(x) = P(x - g(x), \ell, u) - x.
\] (2.12)

It is well-known that \( x^* \) is a first-order stationary point for (2.10) if and only if \( \tilde{g}(x^*) = 0 \) [6], i.e.,

\[
x^* = P(x^* - g(x^*), \ell, u).
\]

If the second-order sufficient condition can be satisfied, then the gradient-projection method is guaranteed to identify the active set at a solution in a
finite number of iterations. After it has identified the correct active set, the gradient-projection algorithm reduces to the steepest-descent algorithm on the subspace of free variables. As a result, this method is often used in conjunction with a second-order method in order to achieve a faster rate of convergence.

2.3 Algorithms for bound-constrained optimization

Many algorithms have been proposed to solve (2.10). For example, algorithms have been proposed by Byrd et al [5], Conn et al [6, 7, 8], Hager and Zhang [23, 24], Lin and Moré [33], Zhu et al [57], and others. The main framework of recent approaches follows the two-stage process as described in Section 2.2.2. However, the algorithms differ on how to identify the active face and how to compute the search direction at each iteration. For example, the method in [6, 7] uses a two-norm trust-region method, and it computes the search step by applying the truncated conjugate-gradient method to the quadratic approximation of the objective function on the active face subject to the trust-region constraints. A similar approach is also used by Lin and Moré [33] in TRON, but the truncated conjugate-gradient method is applied to the quadratic approximation of the objective function on the subspace parallel to the active face. The active-set method proposed by Hager and Zhang [23, 24] uses a backtracking line-search to identify the active face and apply the unconstrained method to the objective function on the active face. One well-known quasi-Newton method for solving (2.10) is the L-BFGS-B algorithm [57], which computes a search direction by truncating the unconstrained L-BFGS [34] update relative to the subspace parallel to the active face.

More recently, Xu and Burke [53] have proposed the ASTRAL algorithm for solving large-scale nonlinear bound-constrained optimization problems. ASTRAL is an active-set algorithm that uses both active-set identification techniques and limited memory BFGS updating for the Hessian approximation. At each iteration, ASTRAL uses a gradient projection step to determine an active face and then forms an $\ell_\infty$ trust-region quadratic programming (QP) subproblem relative to the active face. The trust-region subproblems are solved using primal-dual interior point techniques. The ASTRAL algorithm has a close relationship to the proposed algorithm in this thesis, thus we will describe the ASTRAL algorithm in more detail in Chapter 4.

One particular software package for solving bound-constrained problems that plays an important role in this thesis is BCLS [18]. The BCLS algorithm solves the linear least-squares problem with simple bounds on the variables

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Ax - b \|_2^2 + \frac{1}{2} \delta^2 \| x \|_2^2 + c^T x \\
\text{subject to} & \quad l \leq x \leq u,
\end{align*}$$

(2.13)

where $A$ is $m \times n$ (can be any shape), $b \in \mathbb{R}^m$, $l, u, c \in \mathbb{R}^n$, and $\delta$ is a damping parameter.
In Chapter 4, we show that the problem (2.13) has the exact form for the subproblems to be solved in our proposed algorithm (BCNLS) for bound-constrained nonlinear least-squares. In fact, the BCNLS algorithm utilizes the BCLS package to solve the subproblem at each iteration. We explain the BCLS algorithm and describe how to use it effectively within the BCNLS algorithm in Chapter 4.
Chapter 3

Unconstrained Least Squares

This chapter reviews some theoretical background on unconstrained least squares. We start with a preliminary review of linear least squares in Section 3.1, and we describe the two-norm regularization techniques for solving the ill-conditioned linear systems. In Section 3.2, we describe two classical methods for unconstrained nonlinear least squares problems: the Gauss-Newton method and the Levenberg-Marquardt method. We also describe a self-adaptive strategy for updating the damping parameters in the Levenberg-Marquardt method.

3.1 Linear least squares

The classical linear least-squares problem has the form

$$\minimize_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|^2,$$

where $A$ is an $m \times n$ matrix, and $b$ is an $m$-vector. This problem belongs to a special class of optimization problems with much interest. Firstly, the problem (3.1) is a special case of nonlinear least squares. Secondly, the classical methods for nonlinear least squares, such as the Gauss-Newton method and the Levenberg-Marquardt method in Section 3.2, iteratively solve linear least-squares subproblems.

Geometrically, to solve the problem (3.1), we want to find a vector $x \in \mathbb{R}^n$ such that the vector $Ax \in \mathbb{R}^m$ is the closest point in the range of $A$ to the vector $b$. This geometrical interpretation is illustrated in Figure 3.1.

The problem (3.1) has been studied extensively, and well-known numerical algorithms exist. See Lawson and Hanson [29], Golub and Van Loan [21, Chapter 5], and Trefethen and Bau [52, Lecture 11] for more detail.

3.1.1 Normal equations

For a given $x$, define the residual by

$$r = b - Ax.$$  

A vector $x \in \mathbb{R}^n$ minimizes the norm of the residual if and only if $r$ is orthogonal to the range of $A$:

$$A^T r = 0.$$
This condition is satisfied if and only if
\[ A^T A x = A^T b. \] (3.2)

The equations in (3.2) is called the normal equations. This system is nonsingular if and only if \( A \) has full rank. Consequently, the solution \( x \) is unique if and only if \( A \) has full rank.

### 3.1.2 Two norm regularization

Regularization is a common technique for ill-conditioned linear least squares problems [17]. The solutions of ill-posed problems may not be unique or may be sensitive to the problem data. The regularized least-squares problem has the quadratic form
\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|^2 + \frac{1}{2} \delta^2 \|x\|^2. \tag{3.3}
\]
for some \( \delta > 0 \). Solving (3.3) is equivalent to solving the linear least squares problem
\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \left\| \begin{bmatrix} A & \delta I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2. \tag{3.4}
\]
Note that the matrix in (3.4) necessarily has full column rank. Historically, the study of problem (3.4) as a function of \( \delta \) has also been called ridge regression, damped least squares, or Tikhonov regularization [51].

The regularization is a technical device of changing the ill-posed problem to an approximate problem whose solutions may be well-behaved and preferable. For example, it may be preferable to have approximate solutions with small norms. The intent of using \( \delta \) is to prevent \( \|x\|_2 \) from getting large when \( A \) is ill-conditioned. By performing singular value decomposition \( A = U \Sigma V^T \), we can show that
\[
\|x_\delta\|_2^2 = \sum_{i=1}^n \left( \frac{u_i^T b}{\sigma_i + \delta} \right)^2.
\]
where \( u_i \) is the \( i \)th column of the orthogonal matrix \( U \) and \( \sigma_i \) is the \( i \)th singular value of \( A \). Note that increasing \( \delta \) would cause the norm \( \| x \| \) to decrease. Thus we can obtain some acceptable compromise between the size of the solution vector \( x \) and the size of the norm of the residuals by choosing proper \( \delta \). For further discussion about regularization and the choice of parameter \( \delta \), refer to Lawson and Hanson [30, Chapter 25].

The two-norm regularization can also be regarded as a trust-region method. The solution \( x \) to equations (3.3) and (3.4) also solves

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Ax - b \|_2^2 \\
\text{subject to} & \quad \| x \|_2 \leq \Delta,
\end{align*}
\]

for some \( \Delta > 0 \).

It can be shown that the solution of (3.4) is a solution of the trust-region problem (3.5) if and only if \( x \) is feasible and there is a scalar \( \delta \geq 0 \) such that \( x \) satisfies (3.4) and \( \delta (\Delta - \| x \|_2) = 0 \). See Nocedal and Wright [47, Chapter 10] for the proof.

Instead of setting the trust region \( \Delta \) directly, two-norm regularization modifies \( \Delta \) implicitly by adjusting the damping parameter \( \delta \). The connection between the trust-region method and two-norm regularization is also revisited in Section 3.2.2.

Regularization techniques play a key role in this thesis. The BCLS algorithm in Section 4.4 solves a bound-constrained two-norm regularized linear least-squares problem. The Levenberg-Marquardt method discussed in Section 3.2.2 solves a regularized linear least squares subproblem at each iteration. Moreover, the algorithm for the nonlinear least squares problems with simple bounds described in Chapter 4 solves a regularized linear least squares subproblem with bound constraints at each iteration.

### 3.2 Nonlinear least squares

With the theoretical background of optimization in Chapter 2 and methods for linear least-squares problems from Section 3.1, we now describe two classical methods for unconstrained nonlinear least-squares problems (1.1), namely, the Gauss-Newton method and the Levenberg-Marquardt method. For a more detailed description of these methods, refer to [47, Chapter 10].

#### 3.2.1 The Gauss-Newton method

Perhaps one of the most important numerical methods is Newton’s method. The Newton search direction is derived from the second-order Taylor series approximation to \( f(x + d) \),

\[
f(x + d) \approx f + g^T d + \frac{1}{2} d^T H d. \tag{3.6}
\]
Assume that $H$ is positive definite. By minimizing (3.6) at each step $k$, the Newton search direction is computed as the solution of the linear system

$$Hd = -g.$$ 

Newton’s method requires $H$ to be positive definite in order to guarantee that $d$ is a descent direction. Also, $H$ must be reasonably well-conditioned to ensure that $||d||$ is small. When $H$ is positive definite, Newton’s method typically has quadratic convergence. However, when $H$ is singular, the Newton direction is not defined. Since Newton’s method requires computation of the second derivatives, it is only used when it is feasible to compute $H$.

The Gauss-Newton method differs from Newton’s method by using an approximation of the Hessian matrix. For nonlinear least squares problems, instead of using the Hessian in (1.4), the Gauss-Newton method approximates the Hessian as

$$H \approx J^T J. \quad (3.7)$$

By (1.3), we have $g = J^T r$. Thus at each iteration, the Gauss-Newton method obtains a search direction by solving the equations

$$J^T Jd = -J^T r. \quad (3.8)$$

The Hessian approximation (3.7) of the Gauss-Newton method gives a number of advantages over the Newton’s method. First, the computation of $J^T J$ only involves the Jacobian computation, and it does not require any additional derivative evaluations of the individual residual Hessians $\nabla^2 r_i(x)$, which are needed in the second term of (1.4).

Second, the residuals $r_i(x)$ tend to be small in many applications, thus the first term $J^T J$ in (1.4) dominates the second term, especially when $x_k$ is close to the solution $x^*$. In this case, $J^T J$ is a close approximation to the Hessian and the convergence rate of Gauss-Newton is similar to that of Newton’s method.

Third, $J^T J$ is always at least positive semi-definite. When $J$ has full rank and the gradient $g$ is nonzero, the Gauss-Newton search direction is a descent direction and thus a suitable direction for a line search. Moreover, if $r(x^*) = 0$ and $J(x^*)^T J(x^*)$ is positive definite, then $x^*$ is an isolated local minimum and the method is locally quadratically convergent.

The fourth advantage of Gauss-Newton arises because the equations (3.8) are the normal equations for the linear least-squares problem

$$\min_d \frac{1}{2} ||Jd + r||^2_2. \quad (3.9)$$

In principle, we can find the Gauss-Newton search direction by applying linear least-squares algorithms from Section 3.1 to the subproblem (3.9).

The subproblem (3.9) also suggests another view of the Gauss-Newton method. If we consider a linear model for the vector function $r(x)$ as

$$r(x_k + d) \approx r_k + J_k d,$$
then the nonlinear least-squares problem can be approximated by
\[ \frac{1}{2} \| r(x_k + d) \|^2 \approx \frac{1}{2} \| J_k d + r_k \|^2. \]
Thus, we obtain the Gauss-Newton search direction by minimizing the linear least-squares subproblem (3.9).

Implementations of the Gauss-Newton method usually perform a line search in the search direction \( d \), requiring the step length \( \alpha_k \) to satisfy conditions as those discussed in Chapter 2, such as the Armijo and Wolfe conditions described in Section 2.1.1. When the residuals are small and \( J \) has full rank, the Gauss-Newton method performs reasonably well. However, when \( J \) is rank-deficient or near rank-deficient, the Gauss-Newton method can experience numerical difficulties.

### 3.2.2 The Levenberg-Marquardt method

Levenberg [32] first proposed a damped Gauss-Newton method to avoid the weaknesses of the Gauss-Newton method when \( J \) is rank-deficient. Marquardt [37] extended Levenberg's idea by introducing a strategy for controlling the damping parameter. The Levenberg-Marquardt method is implemented in the software package MINPACK [38, 42].

The Levenberg-Marquardt method modifies the Gauss-Newton search direction by replacing \( J^T J \) with \( J^T J + \delta^2 D \), where \( D \) is diagonal, \( \delta \geq 0 \).

At each iteration, the Levenberg-Marquardt search direction \( d \) satisfies
\[ (J^T J + \delta^2 I) d = -J^T r. \] (3.10)

The Levenberg-Marquardt method can also be described using the trust-region framework of Chapter 2. Let \( \Delta \) be the two-norm of the solution to (3.10), then \( d \) is also the solution of the problem
\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Jd + r \|_2^2 \\
\text{subject to} & \quad \| d \|_2 \leq \Delta.
\end{align*}
\]

Hence, the Levenberg-Marquardt method can be regarded as a trust-region method. The main difference between these two methods is that the trust-region method updates the trust region radius \( \Delta \) directly, whereas the Levenberg-Marquardt method updates the parameter \( \delta \), which in turn modifies the value of \( \Delta \) implicitly. For more description of the differences between the two methods, see [42, 43, 55, 56] for details.

Note that the system (3.10) is the normal equations for the damped linear least-squares problem
\[ \min_{d} \frac{1}{2} \left\| \begin{bmatrix} J \\ \delta I \end{bmatrix} d + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|_2^2. \] (3.11)
The damped least-squares problem (3.11) has the same form as (3.4) in Section 3.1.2. Similar to the Gauss-Newton method, the Levenberg-Marquardt method solves the nonlinear least-squares problem by solving a sequence of linear least-squares subproblems (3.11). The local convergence of the Levenberg-Marquardt method is also similar to that of the Gauss-Newton method. The key strategy decision of the Levenberg-Marquardt method is how to choose and update the damping parameter \( \delta \) at each iteration.

### 3.2.3 Updating the damping parameter

Much research has been done on how to update \( \delta \). See, for example, [14, 36, 42, 46] for more details. From equations (3.10), if \( \delta \) is very small, the Levenberg-Marquardt step is similar to the Gauss-Newton step; and if \( \delta \) is very large, then \( d \approx -\frac{1}{\delta} g \), and the resulting step is very similar to the steepest descent direction. Thus, the choice of \( \delta \) in the Levenberg-Marquardt method affects both the search direction and the length of the step \( d \). If \( x \) is close to the solution, then we want the faster convergence of the Gauss-Newton method; if \( x \) is far from \( x^* \), we would prefer the robustness of the steepest descent method.

Similar to the trust-region method, the ratio \( \rho_k \) (2.7) between the actual reduction of the objective function and the predicted reduction of the quadratic model is used to control and update the damping parameter \( \delta_k \) in the Levenberg-Marquardt method. In general, if the step \( d_k \) is successful and \( \rho_k \) is satisfactory, we accept \( d_k \) and reduce \( \delta \); otherwise, we reject \( d_k \) and enlarge \( \delta \). Naturally, when \( \rho_k \) is close to 1 and the step \( d_k \) is very successful, we may reduce \( \delta_k \) to near 0, then the next step would become an approximate Gauss-Newton step. On the other hand, when \( d_k \) is large and \( \rho_k < 0 \), it is reasonable to enlarge \( \delta_k \) to move the search direction to a descent direction.

We could adopt this strategy for updating the trust region radius \( \Delta_k \) and modify (2.8) for updating the damping parameter in the Levenberg-Marquardt method. For example, one possible approach, described by Fan [15], is

\[
\delta_{k+1} = \begin{cases} 
\frac{\delta_k}{4}, & \text{if } \rho_k > \eta_2, \\
\delta_k, & \text{if } \rho_k \in [\eta_1, \eta_2], \\
4\delta_k, & \text{if } \rho_k < \eta_1.
\end{cases}
\]

(3.12)

The main drawback of this approach is that it is not sensitive to small changes in the values of \( \rho_k \). There exist jumps in \( \delta_{k+1}/\delta_k \) across the thresholds of \( \eta_1 \) and \( \eta_2 \) [15].

Recently, Yamashita and Fukushima [54] show that if the Levenberg-Marquardt parameter is chosen as

\[
\delta_k = \|r_k\|,
\]

and if the initial point is sufficiently close to the solution \( x^* \), then the Levenberg-Marquardt method converges quadratically to the solution \( x^* \). Dan et al [10] propose an inexact Levenberg-Marquardt method for nonlinear least-squares problem with the parameter chosen as

\[
\delta^2_k = \|r_k\|'.
\]
where $\nu$ is a positive constant. The rationale of using a parameter $\nu$ in adjusting the damping parameter is based on the following observations. When the current iterate is too far away from the solution and $\|r_k\|$ is too large, $\delta_k$ could become too large; the resulting step $d_k$ could potentially be too small and prevent the sequence from converging quickly. On the other hand, when the current position is sufficiently close to the solution, $\|r_k\|$ maybe very small, so the $\delta_k$ could have no effect. Dan et al [10] show that the Levenberg-Marquardt method converges to the solution superlinearly when $\nu \in (0, 2)$, and it converges quadratically when $\nu = 2$.

More recently, Fan and Pan [14] propose a strategy of adjusting the damping parameter by

$$
\delta_k^2 = \alpha_k\|r_k\|^\nu,
$$

for $\nu \in (0, 2]$, and $\alpha_k$ is some scaling factor. To avoid the jumps in $\delta_{k+1}/\delta_k$ across the thresholds of $\eta_1$ and $\eta_2$ and make more use of the information of the ratio $\rho_k$, Fan and Pan use a scaling factor $\alpha_k$ at each iteration, where $\alpha_{k+1}$ is a function of $\rho_k$ and $\alpha_k$.

Define the continuous nonnegative function

$$
q(\rho) = \max\{1/4, 1 - 2(2\rho - 1)^3\}.
$$

The image of function (3.14) is illustrated in Figure 3.2. The function of $\alpha_{k+1}$ is then defined as

$$
\alpha_{k+1} = \alpha_kq(\rho_k).
$$

By using the function $q(\rho)$, the factor $\alpha_k$ is updated at a variable rate according to the ratio $\rho_k$, rather than simply enlarging or reducing the $\delta$ by some constant rate as in (3.12).

Combining (3.13), (3.14) and (3.15) together, we have an updating strategy which is referred to as the self-adaptive Levenberg-Marquardt method [14]. This technique is similar to the self-adaptive trust-region algorithm proposed by Hei [26]. Under the local error bound condition, Fan and Pan [14] show that
the self-adaptive Levenberg-Marquardt method converges superlinearly to the solution for \( \nu \in (0, 1) \), while quadratically for \( \nu \in [1, 2] \).

Note that the theoretical convergence proof of the self-adaptive updating strategy is for unconstrained nonlinear least-squares optimization. Nevertheless, in Chapter 4, we show that this updating strategy can be adapted and extended to bound-constrained nonlinear least-squares problems.
Chapter 4

Nonlinear Least Squares
With Simple Bounds

This chapter presents the main contribution of this thesis. We describe our proposed algorithm (BCNLS) for solving the nonlinear least-squares problem with simple bounds

\begin{equation}
\min_{x \in \mathbb{R}^n} \frac{1}{2} \| r(x) \|_2^2
\end{equation}

subject to \( \ell \leq x \leq u \),

where \( \ell, u \in \mathbb{R}^n \) are vectors of lower and upper bounds on \( x \), and \( r \) is a vector of real-valued nonlinear functions.

In Chapter 2, we briefly introduced some algorithms for bound-constrained nonlinear optimization problems. In this chapter, we show how we adopt the general framework of the ASTRAL [53] algorithm into the BCNLS algorithm for solving the nonlinear least-squares problems. We also describe how we make use of the BCLS [18] algorithm to solve a sequence of bound-constrained linear least-squares subproblems.

We begin this chapter with a brief description of the motivation of our approach. Then we illustrate the general framework of the proposed algorithm in Section 4.2. The strategies and techniques used for stabilizing and enhancing the performance of the algorithm are described in Section 4.3. Section 4.4 describes the BCLS algorithm for solving the subproblems. Finally, in Section 4.5, we show the detailed implementation steps of the BCNLS algorithm and summarize the basic features of the algorithm.

4.1 Motivation

Many algorithms (for example, ASTRAL, L-BFGS-B) exist for general nonlinear optimization with simple bounds, but such algorithms do not make use of any knowledge of the objective functions. On the other hand, efficient methods (for instance, BCLS) exist for solving linear least-squares problems with simple bounds. Hence, we want to develop an efficient algorithm specially designed for large-scale nonlinear least-squares problems with simple bounds on variables.

In the design of the BCNLS algorithm, we set the following three goals for our approach. First, we want an algorithm that scales well to large problems. Second, the algorithm should be very effective for the problems whose functions
and gradients are expensive to evaluate. That is, the algorithm must be frugal with the evaluation of functions and gradients. Third, we want to ensure that the variables are always feasible at each iteration, so that the intermediate results can be useful in practical applications.

To achieve the above goals, we use the principles of the classical Gauss-Newton and Levenberg-Marquardt method for unconstrained nonlinear least-squares problem. Our algorithm (BCNLS) solves the bound-constrained nonlinear least-squares problem by iteratively solving a sequence of bound-constrained linear least-squares subproblems. The subproblems are handled by the software package BCLS. To fully exploit the advantages of BCLS, the subproblems are formed by using the two-norm regularization instead of trust-region method. At each iteration, the damping parameter of the subproblem is updated using the self-adaptive updating strategy described in Section 3.2.3.

4.2 Framework

The BCNLS algorithm uses the same quadratic model (3.6) for the objective function and the same Hessian approximation (3.7) as in the Gauss-Newton method. The algorithm ensures that the variables $x^* = x + d$ are always feasible. That is, we superimpose the simple bound constraints

$$\ell \leq x_k + d_k \leq u,$$

for each subproblem at step $k$. The bounds (4.2) can be written as

$$\ell_k \leq d_k \leq u_k,$$

where $\ell_k = \ell - x_k$, and $u_k = u - x_k$ are the updated lower and upper bounds of $d$ at iteration $k$.

As with the Gauss-Newton method, we take advantage of the special structure (4.1) of the nonlinear least-squares objective. By combining (3.9) and (4.3), we have extended the classical unconstrained Gauss-Newton method to the nonlinear least-squares problem with simple bounds. The tentative subproblem has the form

$$\min_{d_k} \frac{1}{2} \| J_k d_k + r_k \|^2$$

subject to $\ell_k \leq d_k \leq u_k$.

However, if the subproblem (4.4) were to be solved without any modification, the proposed method may experience numerical difficulties when $J_k$ is nearly rank-deficient. The numerical solution of (4.4) is dependent on the condition of $J$ just as in the Gauss-Newton method.

In Chapter 3, we have described how the trust-region method and the Levenberg-Marquardt method can be used to remedy the numerical difficulties of the Gauss-Newton method. It is natural for us to adopt these remedies. Recall in Section 2.3, we have mentioned that trust-region methods have been
used in several algorithms [6, 7, 53] for solving the nonlinear optimization problems with simple bounds on variables. For bound-constrained problems, it is more convenient for the trust region to be defined by the $\| \cdot \|_\infty$ norm, which gives rise to a box-constrained trust-region subproblem. For this reason, the $\ell_\infty$ trust region is used in ASTRAL [53].

Our first approach is to mimic ASTRAL’s approach by adding an $\ell_\infty$ trust-region constraint to the subproblem (4.4).

\begin{equation}
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} \| J_k d_k + r_k \|_2^2 \\
\text{subject to} & \quad \ell_k \leq d_k \leq u_k, \\
& \quad \| d \|_\infty \leq \Delta_k.
\end{aligned}
\end{equation}

The role of $\| d \|_\infty \leq \Delta_k$ is to control the norm of $d$. We instead use the two-norm regularization to achieve the same effect. We can express the subproblem (4.5) in a two-norm regularized form

\begin{equation}
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} \| J_k d_k + r_k \|_2^2 + \frac{1}{2} \delta_k^2 \| d_k \|_2^2 \\
\text{subject to} & \quad \ell_k \leq d_k \leq u_k,
\end{aligned}
\end{equation}

for some $\delta_k > 0$. This subproblem has the benefit that it can be solved directly by the software package BCLS [18]. One remaining challenge lies in how to adjust $\delta_k$ at each iteration. For the algorithm to have a rapid convergence rate, it is important to have an effective strategy to update $\delta_k$ at each step. Essentially, the algorithm uses $\delta_k$ to control both the search direction and the step size.

Algorithm 4.1 outlines the basic framework of our approach. The main computational component of this algorithm is solving the bound-constrained linear least-squares subproblem (4.6). In this way, the bounds are handled directly by the subproblem solver. Thus, we can apply the self-adaptive damping strategy described in Section 3.2.3, and the local and global convergence properties remain the same as in the unconstrained cases.

At this point, it is necessary to show that the proposed framework is a reasonable approach for solving the bound-constrained nonlinear least-squares problem (4.1). The key global and local convergence properties of Algorithm 4.1 can be established by showing that the BCNLS algorithm has a very similar framework as the ASTRAL algorithm.

The ASTRAL algorithm solves a general bound-constrained nonlinear problem (2.10) by solving a sequence of $\ell_\infty$ trust-region subproblems. The subproblem has the basic template

\begin{equation}
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} d^T B_k d + s_k^T d \\
\text{subject to} & \quad l_k \leq d \leq u_k, \\
& \quad \| d \|_\infty \leq \Delta_k.
\end{aligned}
\end{equation}

The objective function is based on the quadratic model (2.5), where $B_k$ is a symmetric positive definite Hessian approximation of the objective function at step
Algorithm 4.1: Outline of the BCNLS method

1. initialization: $k \leftarrow 0$, $x_0$, $\delta_0$ given.
2. while not optimal do
3.     Solve the bound-constrained subproblem (4.6) for $d_k$.
4.     Compute the ratio $\rho_k$ (2.7).
5.     if $\rho_k$ is satisfactory then
6.         $x_{k+1} \leftarrow x_k + d_k$
7.     else
8.         $x_{k+1} \leftarrow x_k$
9.     end
10.    Update the damping parameter $\delta_k$.
11.    $k \leftarrow k + 1$.
12. end

$k$. The ASTRAL algorithm solves the quadratic trust-region subproblem (4.7) by using a primal-dual interior point method. The outline of the ASTRAL algorithm is shown in Algorithm 4.2.

Note that the gradient projection restart procedure (line 6) in Algorithm 4.2 is a strategy to assist in the identification of the active constraints at the current solution. Under the standard nondegeneracy hypothesis, Xu and Burke [53] established that the framework of Algorithm 4.2 converges both globally and locally to a first-order stationary point. Moreover, the sequence of $\{x_k\}$ generated by the ASTRAL algorithm converges quadratically to $x^*$ under standard assumptions. The ASTRAL algorithm and its convergence theory follow the pattern established by Powell in [49] and has been used for similar algorithms from the literature [4, 6, 9, 47]. For detailed theoretical proof of the convergence properties, see [53].

After all, the nonlinear least squares problem (4.1) is a special case of the nonlinear problem (2.10). The similarity between the frameworks of Algorithm 4.1 and Algorithm 4.2 suggests that the convergence properties of the ASTRAL algorithm apply to the BCNLS algorithm for the bound-constrained nonlinear least-squares problem. The main differences lie in the specific techniques used for solving the quadratic subproblems. In this sense, Algorithm 4.1 specializes the ASTRAL algorithm to a special class of nonlinear least-squares optimization problems with simple bounds on variables. The new approach takes advantage of the special structure of the objective function by utilizing the special relationships between its gradient, Hessian and the Jacobian. Our approach eliminates the need to build, compute and possibly modify the Hessian of the objective. Therefore, the BCNLS algorithm can make use of the function and gradient evaluations more efficiently than other similar approaches for the general nonlinear optimization problems.
Algorithm 4.2: Outline of the ASTRAL method

1. initialization: $k \leftarrow 0$, $x_0$ given, $\delta_0 > 0$, $\delta_0 \leftarrow \min\{\max\{u_i - l_i\}, \delta_0\}$;
2. initialize symmetric positive definite $B_0 \in \mathbb{R}$;
3. choose a set of controlling constants.
4. while not optimal do
5. if $\|d_k\|_{\infty}$ is not acceptable then
6. Perform a gradient projection restart step.
7. end
8. Solve the trust-region subproblem (4.7) for $d_k$.
9. Compute the ratio $\rho_k$ (2.7).
10. if $\rho_k$ is satisfactory then
11. $x_{k+1} \leftarrow x_k + d_k$
12. else
13. $x_{k+1} \leftarrow x_k$.
14. end
15. Update symmetric positive definite matrix $B_k$.
16. Update the trust-region radius.
17. $k \leftarrow k + 1$.
18. end

4.3 Stabilization strategy

The ASTRAL algorithm makes use of the $\ell_\infty$ trust-region on the search direction $d$ for the subproblems. The $\ell_\infty$ trust region constraint

$$\|d_k\|_{\infty} \leq \Delta_k$$

is equivalent to

$$-\Delta_k e \leq d_k \leq \Delta_k e,$$

where $e = (1, 1, \ldots, 1)^T$. By superimposing the $\ell_\infty$ trust region constraint onto the feasible bounds, the resulting feasible region continues to have a nice rectangular box shape, and the subproblems can be solved by applying the bound-constrained quadratic programming techniques. At each iteration $k$, the ASTRAL algorithm updates the trust-region radius $\Delta_k$ based on the performance of the previous steps. The strategy of updating the trust-region radius prevents the norm of $d$ from getting large and thus has the effect of stabilizing the algorithm.

The BCNLS algorithm uses a very similar strategy to the ASTRAL algorithm. Rather than updating the trust-radius directly, we use the two-norm regularization technique and formulate the subproblem as the form of (4.6). The damping parameter $\delta_k$ is then updated by using the self-adaptive Levenberg-Marquardt strategy in Section 3.2.3. The theoretical equivalence between the trust-region method and the two-norm regularization techniques has been established in Section 3.2.2 and Section 3.2.3. Compared with the trust-region
approach, the two-norm regularization has the advantages of reducing the con-
dition number of the matrix in the subproblem and preventing \(|d|\) from get-
ting large. Furthermore, the two-norm regularization strategy gives rise to a
nicely formatted linear least-squares subproblem with simple bounds on the
variables (4.6), which can be solved directly by BCLS.

4.4 The subproblem solver

The kernel of the computational task in the BCNLS algorithm is the solution
of the bound-constrained linear least-squares subproblems (4.6). The overall
efficiency of the algorithm ultimately depends on the efficient solution of a se-
quence of such subproblems. For large-scale problems, the Jacobian matrices are
potentially expensive to compute and it would require relatively large storage
space if we were to store the Jacobian matrices explicitly. To make the BCNLS
algorithm amenable to large-scale nonlinear least-squares problems, we want to
make sure that the optimization algorithms for the subproblem do not rely on
matrix factorizations and do not require the explicit computation of Jacobian
and the Hessian approximations. In the remainder of this section, we show that
the software package BCLS has the desirable features to make it a viable choice
for solving the subproblem in the BCNLS algorithm.

BCLS is a separate implementation for solving linear least-squares problems
with bounded constraints. Descriptions of the software package BCLS can be
found in [18], [19, Section 5.2], and [31, Section 2.4]. For the completeness of
presenting the BCNLS algorithm, we reproduce a brief description of the BCLS
algorithm here using the generic form of the problem (2.13). Because the linear
term \(c^\top x\) is not used in the BCNLS algorithm, we set \(c = 0\) and eliminate the
linear term. Thus, the simplified version of the linear least-squares problem for
the BCLS algorithm has the form

\[
\begin{array}{ll}
\text{minimize} & \frac{1}{2} \|Ax - b\|_2^2 + \frac{1}{2} \delta^2 \|x\|_2^2 \\
\text{subject to} & \ell \leq x \leq u.
\end{array}
\]  

(4.8)

The BCLS algorithm is based on a two-metric projection method (see, for
example, [2, Chapter 2]). It can also be classified as an active-set method. At
all iterations, the variables are partitioned into two different sets: the set of
free variables (denoted by \(x_B\)) and the set of fixed variables (denoted by \(x_N\)),
where the free and fixed variables are defined as in classical active-set methods
in Section 2.2.1. Correspondingly, the columns of matrix \(A\) are also partitioned
into the free (denoted by \(A_B\)) and fixed (denoted by \(A_N\)) components based on
the set of indices for the free and fixed variables.

At each iteration, the two-metric projection method generates independent
descent directions \(\Delta x_B\) and \(\Delta x_N\) for the free and fixed variables. The search
directions are generated from an approximate solution to the block-diagonal
linear system

\[
\begin{bmatrix}
A_B^T A_B + \delta^2 I & 0 \\
0 & D
\end{bmatrix}
\begin{bmatrix}
\Delta x_B \\
\Delta x_N
\end{bmatrix}
= A_B^T r - \delta^2 x,
\]

(4.9)

where \( r = b - Ax \) is the current residual and \( D \) is a diagonal matrix with strictly positive entries. The right-hand side of the equation (4.9) is the negative of the gradient of the objective in (4.8). The block-diagonal linear system (4.9) is equivalent to the following two separate systems

\[
\begin{align*}
(A_B^T A_B + \delta^2 I) \Delta x_B &= A_B^T r - \delta^2 x_B & (4.10a) \\
D \Delta x_N &= A_N^T r - \delta^2 x_N & (4.10b)
\end{align*}
\]

Thus, a Newton step (4.10a) is generated for the free variables \( x_B \), and a scaled steepest-descent step (4.10b) is generated for the fixed variables \( x_N \). The aggregate step \((\Delta x_B, \Delta x_N)\) is then projected onto the feasible region and the first minimizer is computed along the piecewise linear projected-search direction. See [9] for a detailed description on projected search methods for bound-constrained problems.

The linear system (4.9) is never formed explicitly in the BCLS algorithm. Instead, \( A_B x \) is computed equivalently as a solution to the least-squares problem

\[
\text{minimize} \quad \frac{1}{2} \| A_B x_B - r \|^2 + \frac{1}{2} \| x_B + \Delta x_B \|^2.
\]

(4.11)

The solution to (4.11) can be found by applying the conjugate-gradient type solver LSQR [48] to the problem

\[
\text{minimize} \quad \frac{1}{2} \| A_B \Delta x_B - r \|^2 + \frac{1}{2} \| x_B + \Delta x_B \|^2.
\]

(4.12)

where \( \beta = \max\{\delta, \epsilon\} \), and \( \epsilon \) is a small positive constant. If \( \delta < \epsilon \), the resulting step is effectively a modified Newton step, and it has the advantage of safeguarding against rank-deficiency in the current sub-matrix \( A_B \). The major computational work in the LSQR algorithm is determined by the total number of matrix-vector products with the matrix and its transpose. In order to control the amount of work carried out by LSQR in solving the problem (4.12), the BCLS algorithm employs an inexact Newton strategy [11] to control the accuracy of the computed sub-problem solutions \( \Delta x_B \).

The following notable features make the BCLS algorithm a perfect fit for solving the subproblems (4.6) in the BCNLS algorithm. First, the BCLS algorithm uses the matrix \( A \) only as an operator. The user only needs to provide a routine to compute the matrix-vector products with matrix \( A \) and its transpose. Thus the algorithm is amenable for solving linear least-squares problems in large scale. Second, the BCLS algorithm can take advantage of good starting points. The ability to warm-start a problem using an estimate of the solution is especially useful when a sequence of problems needs to be solved, and each is a

\[
\]
small perturbation of the previous problem. This feature makes BCLS work well as a subproblem solver in the BCNLS algorithm. In general, the solutions to the subsequent subproblems are generally close to each other, and the solution of the previous iteration can serve as a good starting point for the current iteration and thus speed up the solution process. Third, the bounded constraints are handled directly by the BCLS package, which leaves the BCNLS framework straightforward to understand and easy to implement. Fourth, the formulation of the two-norm regularized linear least-squares problem (4.8) enables the BCLS algorithm to fully exploit the good damping strategies. Therefore, the integration of the BCLS algorithm and the self-adaptive damping strategy contributes to the success of the BCNLS algorithm in this thesis.

4.5 The BCNLS algorithm

The implementation of the BCNLS algorithm requires the user to provide two user-defined function routines for a given bound-constrained nonlinear least-squares problem. The residual functions are defined by a user-supplied routine called \textit{funObj}, which evaluates the vector of residuals \( r(x) \) for a given \( x \). The Jacobian of \( r(x) \) is defined by another user-written function called \textit{Jprod}, whose essential function is to compute the matrix-vector products with Jacobian and its transpose, i.e., computes \( Jx \) and \( J^T y \) for given vectors \( x \) and \( y \) with compatible lengths for \( J \). The initial point \( x_0 \) and both lower and upper bounds are optional. If the initial starting point \( x_0 \) is provided but not feasible, then the infeasible starting point is first projected onto the feasible region and the projected feasible point is used as the initial point.

The BCNLS algorithm will output the solution \( x \), the gradient \( g \), and the projected gradient \( \hat{g} \) of the objective function at \( x \). In addition, it provides the user with the following information: the objective value at the solution, the total number of function evaluations, the number of iterations, the computing time, and the status of termination. Algorithm 4.3 describes our implementation of the BCNLS algorithm.

At the initialization stage, steps 1-5 define the damping updating strategy for the given problem. The positive constant \( \alpha_{\text{min}} \) at step 5 is the lower bound of the damping parameter, it is used for numerical stabilization and better computational results. It prevents the step from being too large when the sequence is near the solution. Step 7 sets the step acceptance criteria. Steps 8-10 give the termination criteria for the algorithm. The algorithm is structured around two blocks. Steps 19-22 are executed only when the solution of the previous iteration is successful (i.e., when \( x_k \) has an updated value). Otherwise, the algorithm repeatedly solves the subproblem by steps 24-39 using an updated damping parameter \( \delta_k \) until \( \rho_k \) is satisfactory.

The BCNLS algorithm uses only one function evaluation and at most one gradient evaluation per major iteration. In this way, the algorithm ensures that the minimum number of function and gradient evaluations are used in the solution process. This approach may lead to more calls of BCLS for solving the
Algorithm 4.3: Bound-Constrained Nonlinear Least Squares (BCNLS)

**input**: funObj, Jprod, x₀, ℓ, u

**output**: x, g, g, info

1. Set damping parameter \( \delta_0 > 0 \)
2. Set scaling factor \( \alpha_0 > 0 \)
3. Set updating constant \( \nu \in (0, 2] \)
4. Define a nonnegative nonlinear function \( q(\rho) \)
5. Set the minimum scaling factor \( 0 < \alpha_{\min} < \alpha_0 \)
6. Set the starting point \( d_0 \) for BCLS
7. Set step acceptance parameter \( 0 < p_0 < 1 \)
8. Set optimality tolerance \( \epsilon \)
9. Set maximum iterations
10. Set maximum function evaluations
11. \( k \leftarrow 0 \)
12. \( x_k \leftarrow x_0 \)
13. \( d_k \leftarrow d_0 \)
14. Compute the residuals \( r_k \leftarrow \text{funObj}(x_k) \)
15. \( \delta_k \leftarrow \alpha_0 \| r_k \|^\nu. \)
16. Set iStepAccepted \( \leftarrow \) true
17. while \( \| g_k \| > \epsilon \) do
    18.     if iStepAccepted then
        19.         Compute the gradient \( g_k \leftarrow J\text{prod}(r_k, 2) \)
        20.         Compute the projected gradient \( \tilde{g}_k \leftarrow P(x_k - g_k, \ell, u) - x_k \)
        21.         \( \ell_k \leftarrow \ell - x_k \)
        22.         \( u_k \leftarrow u - x_k \)
        23.     end
        24.     \( [d_k] \leftarrow \text{BCLS}(J\text{prod}, r_k, \ell_k, u_k, d_0, 0, \delta_k) \)
        25.     Compute new residuals \( r_{new} \leftarrow \text{funObj}(x_k + d_k) \)
        26.     Compute the ratio \( \rho_k \) (2.7).
        27.     if \( \rho_k > p_0 \) then
            28.         \( x_{k+1} \leftarrow x_k + d_k \)
            29.         \( r_{k+1} \leftarrow r_{new} \)
            30.         \( d_0 \leftarrow d_k \)
            31.         iStepAccepted \( \leftarrow \) true
        32.     else
            33.         \( x_{k+1} \leftarrow x_k \)
            34.         \( r_{k+1} \leftarrow r_k \)
            35.         iStepAccepted \( \leftarrow \) false.
        36.     end
        37.     Set \( \alpha_{k+1} \leftarrow \max\{\alpha_{\min}, \alpha_k q(\rho_k)\} \).
        38.     Update \( \delta_{k+1} \leftarrow \alpha_{k+1} \| r_k \|^\nu. \)
        39.     \( k \leftarrow k + 1 \).
    19. end
41. \( x \leftarrow x_k, g \leftarrow g_k, g \leftarrow \tilde{g}_k. \)
subproblems and potentially use more computation of the matrix-vector products with Jacobian and its transpose. However, in general, large-scale problems requires more computing time to evaluate the function and gradient values. Therefore, the BCNLS algorithm can perform well for large-scale problems.

Our implementation of Algorithm 4.3 shows that we have achieved our primary goals for the bound-constrained nonlinear least-squares solver. The BCNLS algorithm has the following basic features. First, two-norm regularization and damping techniques are used in solving the subproblems. Second, Hessian approximations are never formed and no second-order derivatives are required to be computed. Third, the simple bounds are handled directly by the subproblem solver BCLS and the sequence of \( \{x_k\} \) is always feasible at any time. Fourth, the BCLS algorithm makes use of the conjugate-gradient type method LSQR. By inheriting the significant features from the BCLS and LSQR software packages, the BCNLS algorithm uses the Jacobian only as an operator. It only requires the user to provide matrix-vector products for the Jacobian and its transpose. Thus the BCNLS package is amenable for bound-constrained nonlinear least-squares problems in large-scale.
Chapter 5
Numerical Experiments

We give the results of numerical experiments on two sets of nonlinear least-squares problems. The first set is composed of the fifteen benchmark nonlinear least-squares problems from the Netlib collection Algorithm 566 [1, 39, 40]. The second set of test problems are generated from the CUTEr testing environment [22].

Using these two sets of test problems, we compare the performances of BC-NLS, L-BFGS-B and ASTRAL. The L-BFGS-B Fortran subroutines were downloaded from [5, 57]. The ASTRAL Matlab source codes were downloaded from [53]. The comparisons are based on the number of function evaluations and CPU times.

5.1 Performance profile

To illustrate the performance, we use the performance profile technique described in Dolan and Moré [12]. The benchmark results are generated by running the three solvers on a set of problems and recording information of interest such as the number of function evaluations and CPU time.

Assume that we are interested in using CPU time as a performance measure, we define $t_{p,s}$ as the CPU time required to solve problem $p$ by solver $s$. Let $t_{\text{best}}$ be the minimum CPU time used by all solvers for problem $p$. The performance ratio is given by

$$r_{p,s} = \frac{t_{p,s}}{t_{\text{best}}}.$$ 

The ideas are the same when comparisons are based on the number of function evaluations.

The performance profile for solver $s$ is the percentage of problems that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. A plot of the performance profile reveals all of the major performance characteristics. In particular, if the set of problems is suitably large, then solvers with large probability $\rho_{s}(\tau)$ are to be preferred. One important property of performance profiles is that they are insensitive to the results on a small number of problems. Also, they are largely unaffected by small changes in results over many problems [12].

In this chapter, all the performance profile figures are plotted in the log-scale. The Matlab performance profile script perf.m was downloaded from http://www-unix.mcs.anl.gov/~more/cops [12]. If a solver $s$ fails on a given
problem \( p \), its corresponding measures, such as the number of function evaluations or CPU time, are set to NaN. In this way, the performance profile not only captures the performance characteristics, but also it shows the percentage of solved problems.

5.2 Termination criteria

In our numerical experiments, each algorithm is started at the same initial point. We use the following criteria to terminate the algorithms:

\[
\frac{\|f_k - f_{k-1}\|}{\max(1, \|f_k\|, \|f_{k-1}\|)} \leq \epsilon_{\text{mach}} \cdot 10^7 \\
\|\tilde{g}_k\| \leq \epsilon \\
f_k \leq \epsilon \quad (\ast) \\
\|d_k\| \leq \epsilon^2 \\
n_f \leq 1000,
\]

where \( \epsilon_{\text{mach}} \) is the machine epsilon, \( \epsilon \) is the optimality tolerance, \( \tilde{g}_k \) is the projected gradient (2.12) of the objective function, \( n_f \) is the number of function evaluations. Note that the termination criteria are the same as in the ASTRAL and L-BFGS-B solvers, except that (\ast) is particular to the BCNLS solver. If an algorithm terminates but the optimality condition \( \|\tilde{g}_k\| \leq \epsilon \) is not satisfied, we consider the algorithm to have failed on the given problem. However, since the objective function is in the form of least-squares, if an algorithm terminates with \( f_k \leq \epsilon \) and \( x_k \) is feasible, we consider the algorithm succeeded on the given problem, regardless of the value of the projected gradient.

5.3 Benchmark tests with Algorithm 566

The test set of benchmark least-squares problems from Netlib Algorithm 566 are unconstrained least-squares problems. We do not include Problems 1-3 from the original Netlib collection, as these are linear least-squares problems. The remaining fifteen nonlinear least-squares problems are used as the benchmark to test the three bound-constrained algorithms. We arbitrarily choose the bounds on the variables to be \( 0 \leq x \leq \infty \).

The original Netlib problems were implemented in Fortran subroutines. We recoded the problems in Matlab. Furthermore, we use the standard initial points as defined in Algorithm 566 for all the test problems. The results with fifteen test problems are given in Table 5.1.

To summarize and compare the overall performances of the three solvers based on the results in Table 5.1, we use the following criteria to judge the outcomes (success or failure) of the solvers on any problem. If both \( f > 10^{-4} \) and \( \|g\| > 10^{-4} \), then we consider the solver has failed on the problem, otherwise it is a success. Based on the criteria, we show the overall performance of the three algorithms in Table 5.2.
Table 5.1: Results of fifteen benchmark tests from Algorithm 566

| Problem | n | Algorithm | $f$       | $||g||$   | $n_f$ | time(Sec) |
|---------|---|-----------|-----------|-----------|-------|-----------|
| 1       | 2 | BCNLS     | $2.47e-15$| $1.30e-07$| 5     | $3.00e-02$|
|         |   | LBFGSB    | $5.08e-06$| $3.51e-01$| 22    | $1.12e-02$|
|         |   | ASTRAL    | $7.42e-06$| $5.36e-02$| 36    | $5.20e-01$|
| 2       | 3 | BCNLS     | $3.62e+02$| $2.50e+02$| 1     | $9.70e-01$|
|         |   | LBFGSB    | $3.62e+02$| $2.50e+02$| 22    | $1.43e-02$|
|         |   | ASTRAL    | $5.31e+01$| $1.29e-03$| 22    | $3.59e-01$|
| 3       | 4 | BCNLS     | $9.98e-30$| $9.99e-15$| 7     | $4.00e-02$|
|         |   | LBFGSB    | $6.82e-06$| $1.05e-02$| 21    | $1.51e-02$|
|         |   | ASTRAL    | $9.83e-06$| $1.27e-03$| 27    | $5.79e-01$|
| 4       | 2 | BCNLS     | $6.40e+01$| $4.62e-07$| 4     | $2.00e-02$|
|         |   | LBFGSB    | $6.40e+01$| $2.13e-14$| 5     | $7.95e-03$|
|         |   | ASTRAL    | $6.40e+01$| $1.74e-06$| 3     | $5.53e-02$|
| 5       | 3 | BCNLS     | $4.11e-03$| $3.29e-06$| 9     | $7.00e-02$|
|         |   | LBFGSB    | $1.54e-04$| $1.08e-06$| 34    | $2.18e-02$|
|         |   | ASTRAL    | $1.54e-04$| $6.90e-06$| 72    | $6.33e-01$|
| 6       | 4 | BCNLS     | $1.21e+04$| $7.30e+08$| 21    | $1.40e-01$|
|         |   | LBFGSB    | $7.30e+08$| $4.06e+10$| 4     | $8.36e-03$|
|         |   | ASTRAL    | $5.03e+06$| $2.86e+07$| 37    | $5.55e-01$|
| 7       | 2 | BCNLS     | $6.88e-03$| $1.53e-06$| 6     | $7.00e-02$|
|         |   | LBFGSB    | $6.88e-03$| $6.61e-06$| 33    | $2.95e-02$|
|         |   | ASTRAL    | $6.88e-03$| $2.72e-05$| 45    | $6.36e-01$|
| 8       | 3 | BCNLS     | $5.65e-08$| $2.66e-04$| 5     | $4.00e-02$|
|         |   | LBFGSB    | $5.17e-06$| $8.43e-03$| 21    | $1.48e-02$|
|         |   | ASTRAL    | $9.30e-06$| $3.08e-03$| 32    | $5.54e-01$|
| 9       | 4 | BCNLS     | $6.22e+01$| $3.52e-03$| 18    | $1.00e-01$|
|         |   | LBFGSB    | $6.22e+01$| $1.60e-05$| 21    | $1.31e-02$|
|         |   | ASTRAL    | $6.22e+01$| $1.14e-02$| 37    | $5.21e-01$|
| 10      | 3 | BCNLS     | $1.06e+05$| $1.14e+00$| 15    | $9.00e-02$|
|         |   | LBFGSB    | $1.06e+05$| $1.57e-03$| 30    | $2.10e-02$|
|         |   | ASTRAL    | $1.06e+05$| $1.66e-03$| 17    | $3.61e-01$|
| 11      | 6 | BCNLS     | $7.39e-06$| $2.33e-04$| 86    | $1.01e+02$|
|         |   | LBFGSB    | $4.81e-02$| $1.87e+00$| 145   | $2.29e+00$|
|         |   | ASTRAL    | $4.40e-02$| $2.29e+00$| 201   | $5.42e+00$|
| 12      | 4 | BCNLS     | $1.22e-14$| $2.50e-07$| 3     | $5.31e+00$|
|         |   | LBFGSB    | $1.00e+09$| $2.00e+06$| 4     | $4.83e+02$|
|         |   | ASTRAL    | $1.43e-22$| $6.90e-10$| 14    | $1.64e+01$|

Continued on next page
Table 5.1 – continued from previous page

<table>
<thead>
<tr>
<th>Problem</th>
<th>( n )</th>
<th>Algorithm</th>
<th>( f )</th>
<th>( |g| )</th>
<th>( n_f )</th>
<th>time(Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>5</td>
<td>BCNLS</td>
<td>(2.54e-02)</td>
<td>(2.57e-03)</td>
<td>11</td>
<td>(1.00e-01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LBFGSB</td>
<td>(2.54e-02)</td>
<td>(2.18e-06)</td>
<td>40</td>
<td>(3.07e-01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASTRAL</td>
<td>(2.54e-02)</td>
<td>(2.52e-02)</td>
<td>76</td>
<td>(1.37e+00)</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>BCNLS</td>
<td>(2.01e-02)</td>
<td>(8.45e-06)</td>
<td>15</td>
<td>(2.30e-01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LBFGSB</td>
<td>(2.01e-02)</td>
<td>(4.90e-06)</td>
<td>73</td>
<td>(3.36e-01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASTRAL</td>
<td>(2.01e-02)</td>
<td>(5.35e-05)</td>
<td>357</td>
<td>(5.65e+00)</td>
</tr>
</tbody>
</table>

Table 5.2: Summary of the benchmark tests from Algorithm 566

Overall, the three algorithms have similar success rates on this set of test problems. For Problem 15 (Chebyquad function), both L-BFGS-B and ASTRAL seem to have difficulties in finding the solution, but BCNLS has performed reasonably well on the problem, terminated with \( f = 7.39 \times 10^{-6} \) and \( \|g\| = 2.33 \times 10^{-4} \). Also, all three solvers failed on Problem 5 (Helical Valley Function), Problem 10 (the Meyer function) and Problem 14 (the Brown and Dennis function). For Problem 16 (the Brown almost-linear function), both BCNLS and ASTRAL performed well, but L-BFGS-B did not solve it. On the other hand, for Problem 17 (the Osborne 1 function), L-BFGS-B performed better than ASTRAL and BCNLS. For details on these function definitions, refer to [1, 39, 40].

Because both BCNLS and ASTRAL were written in Matlab, whereas L-BFGS-B was written in Fortran, we compare the number of function evaluations among the three algorithms. The computing time is compared only between BCNLS and ASTRAL. Figure 5.1 and Figure 5.2 illustrate the performance profiles for BCNLS, L-BFGS-B, and ASTRAL on the fifteen benchmark nonlinear least-squares problems. Figure 5.1 compares the number of function evaluations among the three solvers. We observe that the performance profile for BCNLS lies above both L-BFGS-B and ASTRAL for \( \tau < 2.6 \). That is, for approximately 55% of the problems, the performance ratio of the function evaluation for the BCNLS algorithm is within a factor 2.6 of the best possible ratio. Therefore, the performance profile of the number of function evaluations for BCNLS dominates that of L-BFGS-B and ASTRAL. We conclude that BCNLS is more efficient in terms of function evaluation. Figure 5.2 compares CPU time between BCNLS and ASTRAL. Clearly, the performance profile of CPU time for BCNLS dominates ASTRAL, thus BCNLS uses less CPU time than ASTRAL.
Figure 5.1: Performance profiles on benchmark problems from Algorithm 566: number of function evaluations.

5.4 CUTEr feasibility problems

The general constrained CUTEr test problems have the form

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad \ell_c \leq c(x) \leq u_c, \\
& \quad \ell_x \leq x \leq u_x,
\end{align*}$$

where $x \in \mathbb{R}^n$, $c(x)$ is a vector of $m$ nonlinear real-valued functions of $\mathbb{R}^n \to \mathbb{R}^m$, $\ell_c$ and $u_c$ are lower and upper bounds on $c(x)$, and $\ell_x$ and $u_x$ are lower and upper bounds on $x$.

We use problems from the CUTEr set to generate a set of nonlinear least-squares test problems. These problems are cast as that of finding a feasible point to (5.1). Such problems can arise as subproblems for more general optimization packages, where it is important to have a feasible starting point.

We can rewrite the constrained problem (5.1) as

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad c(x) - s = 0, \\
& \quad \ell_c \leq s \leq u_c, \\
& \quad \ell_x \leq x \leq u_x.
\end{align*}$$

Because we are only interested in getting a feasible point for the CUTEr prob-
Figure 5.2: Performance profiles on benchmark problems from Algorithm 566: CPU time.

Lemma, we can simply ignore the objective function in (5.2), and deal only with the constraints in (5.2). The feasibility problem can then be expressed as

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|f(x) - s\|^2 \\
\text{subject to} & \quad \ell_c \leq s \leq u_c, \\
& \quad \ell_x \leq x \leq u_x.
\end{align*}
\]

To fit this problem statement into the standard bound-constrained nonlinear least-squares framework, we define

\[
\begin{align*}
\bar{x} &= \begin{bmatrix} x \\ s \end{bmatrix}, & \ell = \begin{bmatrix} \ell_x \\ \ell_c \end{bmatrix}, & \bar{u} = \begin{bmatrix} u_x \\ u_c \end{bmatrix}, & \bar{c}(\bar{x}) = c(x) - s.
\end{align*}
\]

Now, the feasibility problem has the standard form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|\bar{c}(\bar{x})\|^2 \\
\text{subject to} & \quad \ell \leq \bar{x} \leq \bar{u}.
\end{align*}
\]

Clearly, the solution to (5.3) satisfies all the constraints in the original CUTEr problem (5.1). We refer to the problems generated from CUTEr in such a way as the CUTEr feasibility problems.

To select the testing problems from the CUTEr environment, we have used the CUTEr selection utility [16]. The basic criterion for the problem selection...
is that the CUTEr problem must have at least one nonlinear constraint. We do not impose any additional restriction on the objective function, the number of variables, bound conditions, and so on.

We installed both the CUTEr and SifDec environment in large scale on a Linux Pentium 4 PC with 3GHz processor and 1GB RAM. SifDec can compile all the selected problems. However, inside the Matlab CUTEr testing environment, the call to csetup would fail for large problems due to an “out of memory” error. We thus exclude very large test problems from the selected test list. As a result, a total of 197 CUTEr feasibility problems were tested on the three solvers BCNLS, L-BFGS-B, and ASTRAL. The maximum number of variables in the tested problems is 700, and the maximum number of constraints $c(x)$ is 284.

To compare the performances of the three solvers, we use the same criteria as in Section 5.3 to judge the success and failure of a solver on any problem. The overall outcomes for the solvers is given in Table 5.3. For this set of test problems, BCNLS has the highest success rate compared with L-BFGS-B and ASTRAL. It has solved more problems than either L-BFGS-B or ASTRAL. The outcomes in both Table 5.2 and Table 5.3 suggest that the overall success rate of BCNLS tends to increase as the size of the test set of nonlinear least-squares problems increases.

Figure 5.3 illustrates the performance profiles for the number of function evaluations for BCNLS, L-BFGS-B, and ASTRAL. We can see from Figure 5.3 that the curve of BCNLS lies above the curves of both L-BFGS-B and ASTRAL. The performance profile shows that BCNLS uses fewer function evaluations than both L-BFGS-B and ASTRAL in solving bound-constrained nonlinear least-squares problems. Figure 5.4 shows the performance profile for CPU time for BCNLS and ASTRAL. Because the performance profile of BCNLS dominates that of ASTRAL, Figure 5.4 shows that BCNLS is more efficient than ASTRAL in CPU time.

For large-scale problems, the cost of function evaluations usually dominates the computing time. The numerical experiments in the two sets of test problems show that BCNLS outperforms its counterparts in solving bound-constrained nonlinear least-squares problems. As the size of the test set and the number of variables increase, the benefits of BCNLS become more promising. The results are not surprising to us, because BCNLS takes advantage of the special structure of the nonlinear least-squares functions and thus is efficient in its use of function evaluations, whereas both L-BFGS-B and ASTRAL do not use any structure information about the functions.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Success</th>
<th>Failure</th>
<th>Success rate(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCNLS</td>
<td>183</td>
<td>14</td>
<td>92.9</td>
</tr>
<tr>
<td>LBFGSB</td>
<td>151</td>
<td>46</td>
<td>76.7</td>
</tr>
<tr>
<td>ASTRAL</td>
<td>166</td>
<td>31</td>
<td>84.3</td>
</tr>
</tbody>
</table>

Table 5.3: Summary of CUTEr feasibility tests
Figure 5.3: Performance profile on CUTEr feasibility problems, number of function evaluations

Figure 5.4: Performance profile on CUTEr feasibility problems, CPU time
Chapter 6

Curve Fitting in Fluorescence Optical Imaging

This chapter describes one industrial application of the BCNLS solver. In collaboration with ART Advanced Research Technologies Inc., a company that designs molecular imaging products for the medical and pharmaceutical industries, we have applied BCNLS for solving the nonlinear inverse problems in time-domain (TD) fluorescence optical-imaging processes.

It is a common practice in research and industry to estimate parameters through a curve fitting procedure. By comparing the measured signal with some model data, one can estimate a combination of parameters that minimizes the difference between measured data and the model signals through the general curve fitting techniques, such as least squares. In this chapter we describe three curve fitting applications that use BCNLS: estimation of fluorescence lifetime, fluorophore depth, and optical properties of a fluorophore.

Our description of the TD fluorescence problem is based on [13].

6.1 Brief introduction to time domain optical imaging

In small-animal fluorescence imaging, fluorophores or labelled biomolecules are injected into small animals such as mice. When an external light source illuminates a region of interest, the fluorophore transitions into an excited state and then re-emits a low-energy photon upon relaxation. The photon can then be detected by some optical detectors to generate fluorescence images. TD optical imaging uses photon-counting technology. It measures the arrival time of each detected photon. Figure 6.1 illustrates a typical configuration of a TD optical imaging system [28].

The fluorescent decay curve is fundamental to the TD fluorescent optical imaging techniques. Figure 6.2 illustrates a typical fluorescent decay curve, which is also commonly called the temporal point-spread function (TPSF) [28]. The TPSF curve contains important information about the fluorophore:

- depth: the location of the fluorophore;
Fluorescence lifetime is also defined as the average time a fluorophore remains in its excited state following excitation. Fluorescent lifetime is an intrinsic property of the fluorophore that depends on the nature of the fluorescent molecule and its environment. Fluorophore identification is therefore possible using fluorescence lifetime. The relationship between the lifetime and the slope of the TPSF is shown in Figure 6.3. In general, the shorter the lifetime of a fluorophore, the steeper the falling slope of the TPSF curve. The time-domain approach provides a unique opportunity to distinguish fluorescent signals with identical spectra based on the differences in their fluorescent lifetime. With TD technology, fluorophore depth can be estimated from the temporal characteristics of the TPSF, and fluorophore concentration can be calculated from the intensity and photon depth. In addition, TD technology allows 3D reconstruc-
6.2 Mathematical models of fluorescence signals

In fluorescence lifetime imaging, lifetime is measured at each pixel and displayed as contrast. In time domain, the fluorescence signal $F_0(t)$ is a sum of several decay curves over time:

$$F_0(t) = \sum_i A_i e^{-\frac{t}{\tau_i}} + DC,$$

where $t$ represents time, $A_i$ and $\tau_i$ are the amplitude and lifetime of the $i$th fluorophore component, and $DC$ represents the offset signal in the data. The amplitude is related to many characteristics of a fluorophore, such as quantum yield, extinction coefficient, concentration, volume, excitation and emission spectra, and so forth. In addition, the combination of the temporal profile of the excitation laser pulse and the system impulse response function (IRF), denoted by $S(t)$, also contributes to the signal. Thus, the measured signal $F(t)$ can be modeled as

$$F(t) = F_0(t) * S(t),$$

where $*$ denotes the convolution operator.

A convolution is an integral that expresses the amount of overlap of one function $g$ as it is shifted over another function $f$. Mathematically, the convolution of two function $f$ and $g$ over a finite range $[0, t]$ is given by

$$[f * g](t) = \int_0^t f(\tau)g(t - \tau)d\tau.$$

When fluorophore is embedded inside a bulk tissue or turbid medium, there will be two more terms contributing to the convolution: the propagation of
excitation light from source to fluorophore, denoted by $H(\vec{r}_s - \vec{r}_f, t)$, and the propagation of fluorescent light from fluorophore to detector, denoted by $E(\vec{r}_f - \vec{r}_d, t)$; see [13]. Let $\vec{r}$ represent a 3-dimensional location vector. The locations of fluorophore, light source, and detector are represented by $\vec{r}_f$, $\vec{r}_s$, and $\vec{r}_d$, respectively. The fluorescence signal $F(t)$ inside tissue has the form

$$F(t) = H(\vec{r}_s - \vec{r}_f, t) * F_0(t) * E(\vec{r}_f - \vec{r}_d, t) * S(t);$$

see [35]. Both the $H(\vec{r}_s - \vec{r}_f, t)$ and $E(\vec{r}_f - \vec{r}_d, t)$ are very complicated functions of many parameters, which require knowledge of tissue optical properties, such as the absorption coefficient $\mu_a$ and the reduced scattering coefficient $\mu'_s$, etc, and the spatial information of the fluorophore. Such information is usually not available in practical applications. For a detailed description of the equations, refer to [25, 35]. However, for a fluorescence signal from small-volume biological tissues—such as those obtained in mice—the light diffusion due to photon propagation, $H(\vec{r}_s - \vec{r}_f, t)$ and $E(\vec{r}_f - \vec{r}_d, t)$, does not significantly change the shape of the temporal profile of a fluorescence signal, while the photon propagation affects the amplitude and the peak position of fluorescence decay curve.

Under the condition of a relatively short optical path between the excitation and fluorescent signal, such as a fluorophore inside a mouse, the effect of light diffusion in tissues can be simplified as a time delay $\Delta t$. That is,

$$F(t) \approx F_0(t) * \delta(\Delta t) * S(t). \quad (6.2)$$

In practice, a measured signal usually contains a DC component. By substituting (6.1) into (6.2), we can express the mathematical model for fluorescence signal as

$$F(t) \approx DC + \delta(\Delta t) * S(t) * \sum_1 A_i e^{-\frac{t}{\tau}}. \quad (6.3)$$

Hereafter in this chapter, all numerical experiments are based on the mathematical model (6.3) for the curve fitting applications.

### 6.3 Curve fitting

In general, curve fitting is the process of minimizing the difference between the observed data $y_d(t_i)$ and the data from mathematical model $y_m(t_i)$ over some unknown parameters. Let

$$r_i = \frac{1}{\sigma_i} (y_d(t_i) - y_m(t_i))$$

be the residual between the observed data and predicted model, where $\sigma_i$ is a weighting term. The objective used for curve fitting is then

$$\chi^2 \equiv \frac{1}{2} \| r(x) \|^2 = \sum_i \left[ \frac{1}{\sigma_i} (y_d(t_i) - y_m(t_i)) \right]^2.$$
In practice, the unknown parameters usually have known lower and upper bounds from their physical settings or based on a priori knowledge of the problem. Thus, the bound-constrained curve fitting problem in TD fluorescence optical imaging has the standard general form of (4.1).

For all experiments on simulated data, we follow ART's research approach to estimate the fluorescence lifetime, depth, and optical properties separately through curve fitting procedure based on the mathematical model in Section 6.2. It would be interesting to combine all three separate curve fitting problems into one and find an optimized combination of all parameters. The optimization problem will remain the same in theory with the only difference of increased number of unknown parameters.

In Section 6.4, 6.5, and 6.6, we describe our experiments of applying the BCNLS solver to estimate the fluorescence lifetime, depth, and optical properties in TD technology. The results are presented in comparisons with ART's implementation of Levenberg-Marquardt algorithm (ART_LM). To precisely compare two optimization algorithms, both methods must use the identical stopping criteria. As a bound-constrained optimization solver, BCNLS uses an optimality condition based on the projected gradient of the objective function. Conversely, ART_LM is an unconstrained optimization method which has an optimality condition based on the gradient of the objective function. In order to compare the computational efficiency, we need to adjust the stopping criteria in different algorithms so that two algorithms have the same termination conditions. This change may slightly affect the reported numerical results in this chapter.

6.4 Fluorescence lifetime estimation

In small-animal fluorescence lifetime imaging, measurements of fluorescence lifetime can yield information on the molecular microenvironment of a fluorescent molecule. Many factors in the microenvironment have effect on the lifetime of a fluorophore. The measurements of lifetime can be used as indicators of those factors. In in vivo studies, these factors can provide valuable diagnostic information relating to the functional status of diseases. Furthermore, the lifetime measurements represent considerable practical advantages, as the measurements are generally absolute, being independent of the fluorophore concentrations.

6.4.1 Fitting parameters

In lifetime fitting procedure, the parameters that need to be estimated include $A_i$, $\tau_i$, $\Delta t$, and $DC$. For a multiple lifetime fluorescence signal composed of $n$ fluorophores, there will be $2n + 2$ parameters. However, one can use two strategies to reduce the number of fitting parameters by two. One strategy is to estimate the DC component separately; another is to use relative amplitude $a_i$ instead of absolute amplitude $A_i$. The normalized amplitude $a_i$ is given by

$$a_i = \frac{A_i}{\sum_j A_j}.$$
This allows one of the relative amplitudes to be determined by the normalization feature of \( Q_i \), i.e., \( \sum \alpha_i = 1 \). Our numerical experiments adopt both of these strategies, and so the total number of fitting parameters is \( 2n \) for a fluorescence signal \( F(t) \) composed of \( n \) fluorophore components.

6.4.2 Parameter bounds

In fluorescence lifetime fitting, the unknown parameters have known bounds:

\[
\tau_{\text{min}} \leq \tau_i \leq \tau_{\text{max}} \\
0 \leq \alpha_i \leq 1 \\
0 \leq \Delta t \leq T/6 \\
0 \leq DC \leq C_{\text{max}}/3,
\]

where \( T \) is the size of the time window of a TPSF, \( C_{\text{max}} \) is the count maximum of a TPSF, \( \tau_{\text{min}} = 0.25\text{ns} \), \( \tau_{\text{max}} = T/4 \). The bounds of \( \alpha_i \) and the lower bound of DC originate from their physical limits. All other bounds are practical limits in order to increase the reliability of fitting results.

6.4.3 Simulated data for lifetime fitting

All data used in the experiments are generated and provided by ART. Several sets of simulated data were used to estimate parameters through solving the nonlinear least squares optimization problem (4.1). Each data set was generated by varying one of the following parameter:

- fluorophore depth \( d \)
- signal DC level
- relative amplitude \( \alpha_i \)
- lifetime gap \( \Delta \tau \)
- mean lifetime \( \bar{\tau} \)
- maximal signal amplitude \( C_{\text{max}} \)

the other parameters are held fixed. In order to obtain statistics of fitting results, random Poisson noise was generated and added to the fluorescence signal, and multiple trials were performed for each set of simulation.

6.4.4 Experimental results

Three sets of error measurements are defined to characterize the performance of different optimization methods for curve fitting. The figure of merit includes the fitting error, standard deviation, and error margin of fitted values. The fitting error reflects the fitting accuracy. The standard deviation indicates the
repeatability of the fitting. Error margin refers to the maximum error in each data set. The error margin measures the robustness of the fitting method.

Figure 6.4 shows one example of the curve fitted by BCNLS in fluorescence lifetime fitting process. In this example, the known parameters in the simulated data are set as $\tau_1 = 0.98, \tau_2 = 1.82, f_1 = 0.25, f_2 = 0.75, DC = 0$, and the computed parameters are $\tau_1 = 1, \tau_2 = 1.84, f_1 = 0.26, f_2 = 0.74, DC = -2$, and the final objective value $\chi^2 = 1.0003$. For a data set with about 1000 data points, the fitted errors of reconstructed values are within reasonable limits for an industry problem.

Some typical results for lifetime fitting are illustrated in Figure 6.5 and Figure 6.6, where Figure 6.5 results from not treating DC as a fitting parameter, and Figure 6.6 results from including DC as a fitting parameter. In both cases, the same simulated data file with varying fluorophore depth was used. In Figures 6.5 and 6.6, the computed results for each fitting parameter are presented in terms of the error of fitted values related to the true values (errTr), the error margin (errMax), the standard deviation (std), and the objective value at the solution ($\chi^2$).

Table 6.1 summarizes the effect of DC component on fluorophore lifetime fitting results from Figures 6.5 and 6.6. It shows that estimating DC component separately has the effect of reducing overall fitting errors and hence improve the accuracy of the fitting results. This suggests that reducing the number of fitting parameters results in fewer degrees of freedom of the system and increases the likelihood of the fitting parameters being close to the true value.

For numerical examples of ART’s implementation of Levenberg-Marquardt
algorithms (ART\_LM), refer to [13]. Table 6.2 shows a comparison of BCNLS and ART\_LM on the same simulated data. Note that BCNLS gets better results in terms of fitting errors. However, ART\_LM takes slightly less computation time compared with BCNLS. This is likely due to the parameter transformation used in ART\_LM to handle the bound constraints indirectly.

### 6.5 Fluorophore depth estimation

Fluorescent inclusion depth can be estimated by fluorescent TPSF peak position if the lifetime of the fluorophore and medium optical properties are known \textit{a priori} [25]. Due to noise, it is not accurate to simply take the peak position from the TPSF. Instead, a curve fitting procedure is employed to determine the depth. To avoid the instability due to noise at the beginning and ending parts of the TPSF, where the count level is low and relative noise level is high, we estimate fluorophore depth by only using the data portion near the peak of the TPSF.

The depth estimation also permits recovery of the fluorophore concentration. The fitting parameters in the fluorophore depth estimation consist of fluorophore depth, and amplitude. The bounds of all the parameters come from their physical limitations.

Similar to the lifetime fitting procedure, simulated data with random noise were generated and added to the fluorescence signal. A curve fitting procedure
Figure 6.6: Lifetime fitting results of simulated data with varying depth, with fitting DC component

is used to attempt to recover the known fluorophore depth and concentration. Similarly, error margin and standard deviation are used to compare the depth fitting results for different algorithms. Table 6.3 shows the depth fitting results using BCNLS for one set of benchmark data. The abbreviations used in Table 6.3 are

- \( d \): the calculated depth;
- \( \text{con} \): the computed fluorophore concentration;
- \( \text{mnErrMgn} \): the average of the error margin over all data sets;
- \( \text{rMnMgn} \% \): the average of the ratio of error margin to true value over all statistic data sets;
- \( \text{rMaxMgn} \% \): the maximum of the ratio of error margin to true value over all statistic data sets;
- \( \text{meanStd} \): the average of the standard deviation over all data sets;
- \( \text{rMeanStd} \% \): the average of the ratio of std to true value over all data sets;
- \( \text{rMaxStd} \% \): the maximum of the ratio of std to true value over all data sets.
Chapter 6. Curve Fitting in Fluorescence Optical Imaging

The following table compares lifetime fitting with and without DC component:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Comparison</th>
<th>No DC</th>
<th>With DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>errTr</td>
<td>0.09</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.51</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.18</td>
<td>0.11</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>errTr</td>
<td>0.09</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.38</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.12</td>
<td>0.22</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>errTr</td>
<td>0.10</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.38</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.15</td>
<td>0.13</td>
</tr>
<tr>
<td>DC</td>
<td>errTr</td>
<td>0.00</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.00</td>
<td>5.35</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.00</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of lifetime fitting with and without DC component

Figure 6.7: Estimated depth versus true depth

In Table 6.3, we compare the depth fitting results for four sets of data points: (A) all data points, (B) data points with depth greater than 1mm, (C) data points with SNR greater than 20, (D) data points with SNR greater than 20 and depth greater than 1mm. The results show that better results can be obtained in the depth fitting procedure by selecting those data points with SNR > 20 and depth > 1mm. Table 6.4 compares the depth estimation results from BCNLS and ART_LM for the latter case. Notice that all error margins and standard deviations obtained with BCNLS are very close to those obtained with the ART_LM algorithm.

Figure 6.7 illustrates the estimated depth compared with true depth. Figure 6.8 shows the relationship between the error of the estimated depth and the signal to noise ratio SNR. In general, signal with higher signal to noise ratio results in more accurate estimated depth. These results by BCNLS also match with the results from ART_LM very closely.
Table 6.2: Comparison of lifetime fitting using BCNLS and ART.LM

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Comparison</th>
<th>BCNLS</th>
<th>ART.LM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>errTr</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.17</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.09</td>
<td>0.11</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>errTr</td>
<td>0.13</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.35</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.17</td>
<td>0.23</td>
</tr>
<tr>
<td>( \alpha_1 )</td>
<td>errTr</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>0.29</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.14</td>
<td>0.15</td>
</tr>
<tr>
<td>DC</td>
<td>errTr</td>
<td>1.53</td>
<td>2.92</td>
</tr>
<tr>
<td></td>
<td>errMax</td>
<td>2.59</td>
<td>7.71</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.83</td>
<td>2.96</td>
</tr>
<tr>
<td>CPU time (seconds)</td>
<td></td>
<td>43.00</td>
<td>36.00</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of depth fitting results for (A) all data points, (B) data points with depth > 1mm, (C) data points with SNR > 20, (D) data points with SNR > 20 and depth > 1mm

6.6 Optical properties estimation

The effective optical properties, absorption coefficient \( \mu_a \) and reduced scattering coefficient \( \mu'_s \), of biological tissues play a critical role in most algorithms for processing optical fluorescence data. For small animals such as mice, there can be large variations of the optical properties within a given region of interest.

In estimation of optical properties, the curve fitting problem has the same general form as the curve fitting problem in Section 6.3. The unknown parameters are the optical properties \( \mu_a \) and \( \mu'_s \), and the amplitude \( A \). The lower and upper bounds of the optical properties are based on a priori knowledge of the medium optical properties values. In our experiments, these bounds are set as

\[
0.0001 \leq \mu_a \leq 0.2 \\
0.2 \leq \mu'_s \leq 3.5 \\
0.85 \leq A \leq 1.15
\]
Table 6.4: Comparison of depth fitting results for BCNLs and ART.LM for data points with SNR > 20 and depth > 1mm

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Depth</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BCNLs</td>
<td>ART.LM</td>
</tr>
<tr>
<td>mnErrMgn</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>rMnMgn</td>
<td>1.96</td>
<td>2.00</td>
</tr>
<tr>
<td>rMaxMgn</td>
<td>8.63</td>
<td>8.60</td>
</tr>
<tr>
<td>meanStd</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>rMeanStd</td>
<td>1.17</td>
<td>1.20</td>
</tr>
<tr>
<td>rMaxStd</td>
<td>5.29</td>
<td>5.30</td>
</tr>
</tbody>
</table>

Figure 6.8: Relative accuracy of calculated depth versus signal SNR

where the units of $\mu_a$ and $\mu'_a$ are mm$^{-1}$.

6.6.1 Simulation for estimating optical properties

An IRF curve with a peak count of 1000 and a full-width-at-half-max (FWHM, or the time between 50% of the maximum count level before and after the peak) of 400 picoseconds was numerically generated. The peak count of the TPSFs was fixed to be 1500. Different TPSF curves were calculated using each pair of $(\mu_a, \mu'_a)$ combination from the pre-selected sets of values in Table 6.5.

For comparisons, one set of tests was carried out without noise. For each pair of $(\mu_a, \mu'_a)$, multiple trials were performed by adding Poisson noise (shot noise) to the TPSF and IRF curves. The mean values of the fitted optical properties and standard deviation for all the trials were then computed. The relative errors are the differences between the fitted values and their corresponding true values are shown in Table 6.5.
Table 6.5: $\mu_a$ and $\mu'_s$ values used in simulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values (mm$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_a$</td>
<td>0.001, 0.005, 0.01, 0.03, 0.05, 0.1</td>
</tr>
<tr>
<td>$\mu'_s$</td>
<td>0.2, 0.5, 1, 2, 3</td>
</tr>
</tbody>
</table>

Table 6.6: Comparison of results of optical properties with ART.LM and BCNLS

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Statistic</th>
<th>ART.LM without noise</th>
<th>BCNLS without noise</th>
<th>ART.LM with noise</th>
<th>BCNLS with noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residual</td>
<td>Mean</td>
<td>0.00</td>
<td>0.00</td>
<td>65.60</td>
<td>65.71</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td>0.00</td>
<td>0.00</td>
<td>1.95</td>
<td>1.96</td>
</tr>
<tr>
<td>Time (s)</td>
<td>Mean</td>
<td>0.15</td>
<td>0.17</td>
<td>0.13</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td>0.03</td>
<td>0.08</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>Rel $\mu_a$ (%)</td>
<td>Mean</td>
<td>-0.00</td>
<td>0.00</td>
<td>-1.48</td>
<td>-0.22</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td>0.00</td>
<td>0.00</td>
<td>4.56</td>
<td>4.10</td>
</tr>
<tr>
<td>Rel $\mu'_s$ (%)</td>
<td>Mean</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.35</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td>0.00</td>
<td>0.00</td>
<td>0.79</td>
<td>0.75</td>
</tr>
</tbody>
</table>

6.6.2 Results for optical properties

We have compared the performance of the solver BCNLS with ART.LM, note that no parameter transform was used with ART.LM in this case. The comparison criteria for different algorithms include the relative error, residual function value at the solution, and computation time. Table 6.6 shows the results of optical properties for the simulated data. For the data without Poisson noise, BCNLS performs as well as ART.LM method. However, for the simulations with Poisson noise, BCNLS outperforms the ART.LM method.

6.7 Discussion

The numerical experiments on estimation of fluorescence lifetime, depth and optical properties at ART illustrate that BCNLS can be applied to solve the curve fitting problems with bounds on parameters in practice. The challenges in fluorescence lifetime fitting lie in multiple lifetime fitting. Although the experimental results presented in this chapter are limited to fluorescent signals of two fluorophore components, BCNLS can be easily adapted to the cases for multiple fluorescence lifetime curve fitting.

Moreover, curve fitting is a process of finding an optimized combination of unknown parameters by some optimization method. In theory, there are some differences between the solutions of curve fitting optimization and the reconstruction of the known values in the simulation. Given the mathematical model for fluorescence signal in Section 6.2, our experiments reveal that a good curve fitting optimization solution may not always result in satisfactory reconstruction.
of the unknown. Nevertheless, the experiments show that the BCNLS solver is able to handle the simple bounds on variables in this curve fitting application.
Chapter 7

Conclusions and Future Work

In this thesis, we have developed the BCNLS algorithm for bound-constrained nonlinear least squares that solves a sequence of bound-constrained linear least squares subproblems, using techniques motivated by the Levenberg-Marquardt method and $\ell_2$-norm regularization. The subproblems are solved by BCLS, an existing solver for bound-constrained linear least squares. Our approach differs substantially from previous general bound-constrained optimization methods. Our approach takes advantage of the special structure of the objective and uses the two-norm regularization to formulate the subproblems. BCNLS efficiently uses of function evaluations at the expense of potentially solving more linear linear-squares subproblems. The convergence properties of the new method are generally as good as, or better than, quasi-Newton methods such as L-BFGS-B, or the $\ell_\infty$ trust-region approach such as ASTRAL.

Preliminary results are promising. The industrial application of curve fitting in fluorescence optical imaging field suggests that the new algorithm may prove to be useful in practice.

There is much scope for further development of the algorithm described in this thesis.

One promising direction would be to allow the user to choose preconditioners to speed up the process of solving the subproblems for large-scale systems. It is well-known that conjugate-gradient methods can be accelerated if a nonsingular preconditioner matrix $M$ is available. Because the subproblem solver BCLS uses the conjugate gradient LSQR algorithm, we believe that preconditioning would be useful in our proposed algorithm. However, good preconditioners are hard to define, especially for large and sparse systems, although BCLS is capable to solve the subproblem with user-supplied preconditioning routines.

Finally, the damping parameter $\delta$ plays a critical role in the proposed algorithm BCNLS. The performance of the updating strategy for $\delta$ depends on problems. It is a longstanding challenge to find a perfect updating strategy for different optimization problems. The self-adaptive Levenberg-Marquardt strategy works well in general, but further investigation may lead to better updating strategies.
Bibliography


Bibliography


