SCALING RANK-ONE UPDATING FORMULA AND ITS APPLICATION 
IN UNCONSTRAINED OPTIMIZATION 

L.P. SUN

A Thesis Submitted To 
The Australian National University 
For The Degree Of Doctor Of Philosophy 
June, 1990
DECLARATION

Some material in chapter 5 to 9 is contained in Osborne and Sun (1987), (1988), (1989a), (1989b) and represents joint work with my supervisor Dr. M.R. Osborne. With this qualification, and unless otherwise stated, the work in this thesis is my own.


ACKNOWLEDGEMENTS

It is a pleasure to be able to thank some of those who have helped me on the long and sometimes winding road to completing this thesis. Firstly, I express my heartfelt gratitude to my supervisor Dr. M.R.Osborne for his sustained enthusiasm, instruction with skill and patience, and concrete help. Dr. S.Roberts and Dr. R.S.Anderssen read early drafts of material which became a part of Chapter two.

Thanks also to the Australian National University for financial support, to Prof. C.Heyde and Prof. P.Hall for moral support and to Ms. J.Goodwin and Ms. B.L.Scala for their ardent help.
# TABLE OF NOTATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cond(·)</td>
<td>condition number of a matrix</td>
</tr>
<tr>
<td>I</td>
<td>unit matrix</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>n-dimensional space</td>
</tr>
<tr>
<td>$x$</td>
<td>variable in an optimization problem</td>
</tr>
<tr>
<td>$x_k$</td>
<td>iterates in an iterative method</td>
</tr>
<tr>
<td>$x^*$</td>
<td>local minimizer</td>
</tr>
<tr>
<td>$g(x)$</td>
<td>gradient vector</td>
</tr>
<tr>
<td>$G(x)$</td>
<td>Hessian matrix</td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$O(·)$, $o(·)$</td>
<td>&quot;big O&quot; and &quot;little o&quot; notation</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>element in set</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>equal by definition</td>
</tr>
</tbody>
</table>
ABSTRACT

This thesis deals with algorithms used to solve unconstrained optimization problems. We analyse the properties of a scaling symmetric rank one (SSR1) update, prove the convergence of the matrices generated by SSR1 to the true Hessian matrix and show that algorithm SSR1 possesses the quadratic termination property with inexact line search. A new algorithm (OCSSR1) is presented, in which the scaling parameter in SSR1 is chosen automatically by satisfying Davidon's criterion for an optimally conditioned Hessian estimate. Numerical tests show that the new method compares favourably with BFGS. Using the OCSSR1 update, we propose a hybrid QN algorithm which does not need to store any matrix. Numerical results show that it is a very promising method for solving large scale optimization problems. In addition, some popular technologies in unconstrained optimization are also discussed, for example, the trust region step, the descent direction with supermemory and the detection of large residual in nonlinear least squares problems.

The thesis consists of two parts. The first part gives a brief survey of unconstrained optimization. It contains four chapters, and introduces basic results on unconstrained optimization, some popular methods and their properties based on quadratic approximations to the objective function, some methods which are suitable for solving large scale optimization problems and some methods for solving nonlinear least squares problems. The second part outlines the new research results, and contains five chapters. In Chapter 5, the scaling rank one updating formula is analysed and studied. Chapter 6, Chapter 7 and Chapter 8 discuss the applications for the trust region method, large scale optimization problems and nonlinear least squares. A final chapter summarizes the problems used in numerical testing.
TABLE OF CONTENTS

Declaration \hspace{1cm} ii
Acknowledgements \hspace{1cm} iii
Table of Notation \hspace{1cm} iv
Abstract \hspace{1cm} v
Table of Contents \hspace{1cm} vi

Part I: A Survey of Unconstrained Optimization \hspace{1cm} 1

Chapter 1: Introduction
1.1 Optimality Conditions \hspace{1cm} 2
1.2 Convergence Analysis of Descent Methods \hspace{1cm} 4
1.3 Rate of Convergence \hspace{1cm} 8
1.4 Quadratic Objective Function \hspace{1cm} 9

Chapter 2: Newton-like Methods
2.1 Taylor Series Model \hspace{1cm} 12
2.2 Newton's Method and Its Modifications \hspace{1cm} 13
2.3 Quasi-Newton Methods \hspace{1cm} 16
2.4 Broyden Family \hspace{1cm} 17
2.5 BFGS, DFP and the Convex Class \hspace{1cm} 20
2.6 Updates with Optimal Condition in the Broyden Family \hspace{1cm} 21
2.7 Trust Region Methods \hspace{1cm} 23
Chapter 3: Methods for Large Scale Optimization

3.1 Conjugate Gradient Methods
3.2 Quasi-Newton Methods with Limited Memory

Chapter 4: Nonlinear Least Squares Problem

4.1 Preliminary
4.2 Gauss-Newton-type Methods
4.3 All Newton-type Methods
4.4 Hybrid Methods

Part 2: The SSR1 Update and Its Applications

Chapter 5: The Symmetric Rank-One Update

5.1 History
5.2 SR1 Formula
5.3 SSR1 Update Formula
5.4 Quadratic Termination Property on Algorithm SSR1
5.5 Convergence of the Matrices Generated by SSR1 Update
5.6 The Optimal Choice of $\theta$
5.7 Numerical Results

Chapter 6: An Application of the OCSSR1 Update for Trust Region Methods

6.1 Trust Region Methods on a Subspace
6.2 Choice of Subspace
6.3 Convergence Analysis
6.4 Calculation of the Trust Region Step
Chapter 7: An Application of the OCSSR1 Update for Large Scale Optimization Problems

7.1 A Scaling Conjugate Gradient Direction 86
7.2 A Quasi-Newton Direction 87
7.3 One-Step Quasi-Newton Iteration 88
7.4 Two-Step Quasi-Newton Iterations 89
7.5 Quadratic Termination Property 90
7.6 Numerical Results 98
7.7 Conclusion 103

Chapter 8: An Application of the OCSSR1 Update in Nonlinear Least Squares Problem

8.1 An Idea for Improving the Hessian Estimate 104
8.2 A Switch for the Hybrid Approach 106
8.3 A Hybrid Algorithm 107
8.4 Numerical Results 109

Chapter 9: Numerical Experiments

9.1 Test Functions 112
9.2 Some Popular Algorithms 124
9.3 Statements on Numerical Tests 125

References 126
PART ONE

A SURVEY OF UNCONSTRAINED OPTIMIZATION

1.1 Optimality Conditions

In unconstrained optimization, the basic problem considered is

\[ \text{Minimize } f(x) \]

where \( f(x) \) is a real, continuous, differentiable function. For (1.1), the first

Lemma 1.1.1: The point \( x^* \) is a strong local minimum of \( f(x) \) if there exists an \( \varepsilon > 0 \)

such that

\[ f(x^*) < f(x) \text{ for all } x \in B(x^*, \varepsilon) \]

where \( B(x^*, \varepsilon) = \{ x \mid \| x - x^* \| < \varepsilon \} \).

Definition 1.1.2: The point \( x^* \) is a weak local minimum of \( f(x) \) if there exists an \( \varepsilon > 0 \)

such that

\[ f(x^*) \leq f(x) \text{ for all } x \in B(x^*, \varepsilon) \]
CHAPTER ONE
INTRODUCTION

Methods for solving unconstrained optimization problem seek the minimizing point of a nonlinear function of \( n \) real variables. Such problems occur in almost all areas of science and engineering, and in many areas of the social sciences. Before discussing methods for unconstrained optimization, it is necessary to outline the basic theoretical background. This is the principle purpose of this chapter.

1.1 Optimality Conditions

In unconstrained optimization, the basic problem considered is

\[
\text{(UO) Minimize } f(x)
\]

where \( f(x): \mathbb{R}^n \rightarrow \mathbb{R} \) is a real continuously differentiable function. For (UO), the first concept is related to the definition of the solution.

**Definition 1.1.1** The point \( x^* \) is a strong local minimum of \( f(x) \) if there exists an \( \varepsilon > 0 \) such that

\[
f(x^*) < f(x),
\]

for all \( x \in N(x^*, \varepsilon), x \neq x^* \) where \( N(x^*, \varepsilon) = \{ x \mid \| x - x^* \| < \varepsilon \} \).

**Definition 1.1.2** The point \( x^* \) is a weak local minimum of \( f(x) \) if there exists an \( \varepsilon > 0 \) such that

\[
f(x^*) \leq f(x),
\]

for all \( x \in N(x^*, \varepsilon); \)
One of the important questions is when does \( f(x) \) possess a minimum? The following optimality condition gives an answer.

*Theorem 1.1.3 (necessary conditions)* Let \( x^* \) be a strong local minimum for (UO), then it is a critical point, i.e.,
\[
g(x^*) = 0.
\]

In addition, for some \( \varepsilon > 0 \), if \( f(x) \) is a twice continuously differentiable function over \( N(x^*,\varepsilon) \), then its Hessian \( G(x^*) \) is positive semi-definite.

*Theorem 1.1.4 (sufficient conditions)* Let \( f(x) \) be a twice continuously differentiable function over \( N(x^*,\varepsilon) \). If
\[
\|g(x^*)\| = 0
\]
and \( G(x^*) \) is positive definite, then \( x^* \) is a strong local minimum for (UO). That is, there exists some \( \varepsilon > 0 \) such that
\[
f(x^*) < f(x), \text{ for } x \neq x^* \text{ and } x \in N(x^*,\varepsilon).
\]

*Definition 1.1.5* The point \( x^* \) is a global minimum of \( f(x) \) if
\[
f(x^*) \leq f(x),
\]
for all \( x \in \mathbb{R}^n \).

By means of Weierstrass' theorem, the existence of a global minimum for (UO) can be guaranteed:
Theorem 1.1.6 If \( \lim_{k \to \infty} f(x_k) = \infty \) for every sequence \( \{x_k\} \) such that \( \lim_{k \to \infty} \|x_k\| = \infty \), or, more generally, if the set \( \{x \mid f(x) \leq c\} \) is nonempty and compact for some \( c \in \mathbb{R} \), then there exists a global minimum for \((UO)\).

However, except in special cases, it is usually difficult to find a global minimum for \((UO)\). In general, a numerical method is only expected to provide a local minimum of \( f(x) \). Under convexity assumptions on \( f(x) \), a local minimum for \((UO)\) is also its global minimum.

Theorem 1.1.7 Let \( f(x) \) be convex and continuously differentiable over \( \mathbb{R}^n \). Then a point \( x^* \) is a global minimum for \((UO)\) if and only if

\[
g(x^*) = 0.
\]

Further conditions are required to guarantee the uniqueness of the global minimum (for example, strict convexity).

1.2 Convergence Analysis of Descent Methods

Most methods for solving \((UO)\) are based on iteration and descent. Assume that \( x_k \) is given, we determine a search direction \( d_k \) and find a steplenth \( \alpha_k \) along \( d_k \) so that the objective function \( f(x) \) is decreased; that is,

\[
\begin{align*}
f(x_k + \alpha_k d_k) < f(x_k).
\end{align*}
\] (1.2.1)

The point \( x_k + \alpha_k d_k \) is regarded as a new approximation to \( x^* \), and the above process is repeated. Thus we obtain a sequence \( \{x_k\} \). Here \( d_k \) is a descent direction which satisfies
\[
\begin{cases}
    d_k^T g_k < 0 & \text{if } g_k \neq 0, \\
    d_k = 0 & \text{if } g_k = 0.
\end{cases}
\] (1.2.2)

On the other hand, the requirement for the inequality (1.2.1) that the objective function \( f(x) \) decreases on each iteration is not sufficient to guarantee convergence to a stationary point and opens up the possibility of premature termination because it permits negligible reduction in \( f(x) \). Suitable procedures for finding the steplenth \( \alpha_k \) must be related to the convergence of an algorithm. There are a number of rules for choosing \( \alpha_k \):

**Rule 1.2.1 (exact line search)** \( \alpha_k \) is chosen so that

\[
f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k).
\]

For a step \( \alpha_k^* \) which is obtained by the exact line search, we have

\[
g(x_k + \alpha_k^* d_k)^T d_k = 0. \tag{1.2.3}
\]

**Rule 1.2.2 (Armijo, 1966)** Fixed scalar \( \beta \), and \( \tau \) with \( \beta \in (0, 1) \), and \( \tau \in (0, 1/2) \) are selected, and we set \( \alpha_k = \beta^{m_k} \), where \( m_k \) is the first nonnegative integer \( m \) for which

\[
f(x_k) - f(x_k + \beta^m d_k) \geq -\tau \beta^m g_k^T d_k.
\]

**Rule 1.2.3 (Goldstein, 1965)** A fixed scalar \( \tau \in (0, 1/2) \) is selected, and \( \alpha_k \) is chosen to satisfy
The right-hand inequality of Rule 1.2.3 may exclude the minimizing point of $f(x_k + \alpha d_k)$ when $f(x_k + \alpha d_k)$ is non-quadratic for $\alpha$. For this reason, a different steplenth termination criterion based on the slopes is in common use.

**Rule 1.2.4** (Wolfe 1969, 1971) $\alpha_k$ is choosen so that

$$f(x_k) - f(x_k + \alpha_k d_k) \geq -\tau \alpha_k g_k^T d_k$$

and

$$g(x_k + \alpha_k d_k)^T d_k \geq \beta g_k^T d_k.$$  \hfill (1.2.4a, 1.2.4b)

where $\tau$ and $\beta$ are some scalars with $\tau \in (0, 1/2)$ and $\beta \in (\tau, 1)$. Relation (1.2.4a), in view of $g_k^T d_k < 0$, guarantees a sufficient decrease for $f(x)$. Usually $\tau$ is chosen very close to zero, for example $\tau = 10^{-4}$ is a recommended value. Relation (1.2.4b) ensures that $\alpha_k$ is not too small, and $\beta = 0.9$ is a common value. In practice, a more stringent two-sided test on the slope can be used in place of (1.2.4b)

$$|g(x_k + \alpha_k d_k)^T d_k| \leq -\beta g_k^T d_k.$$ \hfill (1.2.4c)

The following lemma shows that under mild assumptions there is an interval of steplenth $\alpha$ satisfying (1.2.4a), (1.2.4b) or (1.2.4a), (1.2.4c).

**Lemma 1.2.5** Assume that there is a scalar $c$ such that $f(x) \geq c$ for all $x \in \mathbb{R}^n$, and that $g_k^T d_k < 0$. If $\tau \in (0, 1/2)$ and $\beta \in (\tau, 1)$, then there exists an interval $[c_1, c_2]$ with $0 < c_1 < c_2$, such that every $\alpha \in [c_1, c_2]$ satisfies Rule 1.2.4.
The kth step of the basic descent method is summed up in the following algorithm.

**Algorithm 1.2.6 (principal descent method)**

- **Step 1** Determine a search direction $d_k$ which satisfies the condition (1.2.2).
- **Step 2** Find a steplenth $\alpha_k$ such that (1.2.4a) and (1.2.4b) are satisfied.
- **Step 3** Set $x_{k+1} = x_k + \alpha_k d_k$.

We now introduce a condition on the descent direction.

**Definition 1.2.7** Let $\{x_k\}$ be a sequence generated by Algorithm 1.2.6. The sequence $\{d_k\}$ is said to be uniformly good for the sequence $\{x_k\}$ if for every convergent subsequence $\{x_{\kappa}\}$ for which

$$\lim_{k \to \infty} g_k \neq 0$$

$$\lim_{k \to \infty} \inf_{\kappa \in \kappa} \frac{g_k^T d_k}{\|d_k\|} > 0$$

and

$$\lim_{k \to \infty} \sup_{\kappa \in \kappa} \|d_k\| < \infty.$$ 

In other words, $\{d_k\}$ is uniformly good relative to $\{x_k\}$ if whenever a subsequence $\{g_k\}_{\kappa}$ tends to a nonzero vector, the corresponding subsequence of directions $d_k$ is bounded and does not tend to be orthogonal to $g_k$.

**Remark 1.2.8** The following two conditions guarantee the sequence $\{d_k\}$ is uniformly good (for example, see Bertsekas 1981): If, for scalars $c_1 > 0$, $c_2 > 0$ and all $k$,
(b) \( d_k = -W_k g_k \), with \( W_k \) a symmetric positive definite matrix satisfying

\[
c_1\|g_k\|\|z\|^2 \leq z^T W_k z \leq c_2\|g_k\|\|z\|^2,
\]

where \( z \in \mathbb{R}^n \). Then \( \{d_k\} \) is uniformly good for the sequence \( \{x_k\} \).

A global convergence result for the descent methods can now be established:

**Theorem 1.2.8** Let \( \{x_k\} \) be a sequence generated by Algorithm 1.2.6 with the Rule 1.2.4. If \( \{d_k\} \) is uniformly good relative to \( \{x_k\} \), then either \( g_k = 0 \) for some \( k \), or

\[
\lim_{k \to \infty} g_k = -\infty, \text{ or } \lim_{k \to \infty} g_k = 0.
\]

### 1.3 Rate of Convergence

For convergent descent methods, it is important to know the rate of convergence. Firstly we have the related definitions.

**Definition 1.3.1** Assume that \( x_k \in \mathbb{R}^n, k = 1,2,\ldots, \) and that \( \lim_{k \to \infty} x_k = x^* \). (a) If there exist \( k_t \geq 1 \) and \( \mu \in [0,1) \) such that for all \( k \geq k_t \),

\[
\|x_{k+1} - x^*\| \leq \mu \|x_k - x^*\|,
\]

then the sequence \( \{x_k\} \) is said to converge q-linearly to \( x^* \). (b) If for some sequence of scalars \( \{\mu_k\} \) that converge to 0

\[
\|x_{k+1} - x^*\| \leq \mu_k \|x_k - x^*\|,
\]

then the sequence \( \{x_k\} \) is said to converge \( \nu \)-linearly to \( x^* \).
then \( \{x_k\} \) is said to converge q-superlinearly to \( x^* \). (c) If there exist \( k_t \geq 1, p \geq 0 \) and \( \mu \geq 0 \) such that for all \( k \geq k_t \),

\[
\|x_{k+1} - x^*\| \leq \mu \|x_k - x^*\|^p,
\]

then \( \{x_k\} \) is said to converge to \( x^* \) with q-order at least \( p \). Specially, if \( p = 2 \), the convergence is said to be q-order of two.

Secondly, the concept of Lipschitz continuity is useful for a descent algorithm in theoretical analysis.

**Definition 1.3.2** A matrix function \( G(x): \mathbb{R}^{nxn} \to \mathbb{R}^n \) is said to be Lipschitz continuous with constant \( \gamma \) in an open neighborhood \( N_c \in \mathbb{R}^n \), written \( G \in \text{Lip}_\gamma(N_c) \), if for all \( x_1, x_2 \in N_c \), there holds

\[
\|G(x_1) - G(x_2)\| \leq \gamma \|x_1 - x_2\|.
\] (1.3.1)

We report a convergence rate result for the principal descent method (Algorithm 1.2.6), which is given by Dennis and More (1974).

**Theorem 1.3.3** Assume that \( f(x) \) be a twice continuously differentiable in an open convex set \( \Gamma \), and that \( G \in \text{Lip}_\gamma(\Gamma) \). If a sequence \( \{x_k\} \) generated by Algorithm 1.2.6 with Rule 1.2.4 converges to \( x^* \) at which \( G^* \) is positive definite, and if

\[
\lim_{k \to \infty} \frac{\|g_k + G_k d_k\|}{\|d_k\|} = 0,
\] (1.3.2)

Then there is an index \( k_0 \) such that for all \( k > k_0 \), \( \alpha_k = 1 \) is admissible. Furthermore, if \( \alpha_k = 1 \) for all \( k > k_0 \), then \( \{x_k\} \) converges q-superlinearly to \( x^* \).
1.4 Quadratic Objective Function

A twice continuously differentiable function with positive definite Hessian matrix at $x^*$, when expanded about a local minimum $x^*$, can be approximated by a quadratic function. Therefore, many of the important characteristics of descent methods can be revealed by investigation of the case where the objective function is quadratic. Consider the quadratic function

$$F(x) = c + b^T x + \frac{1}{2} x^T Ax$$

(1.4.1)

where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix.

Conjugacy and quadratic termination are two important concepts which should not be neglected for minimizing a quadratic function.

**Definition 1.4.1** Given a symmetric positive definite matrix $A$ and $k$ nonzero vectors $z_1, \ldots, z_k$, if for all $i$ and $j$ with $i \neq j$ we have $z_i^T A z_j = 0$, then vectors $z_1, \ldots, z_k$ are said to be $A$-conjugate.

**Corollary 1.4.2** If $z_1, \ldots, z_k$ are $A$-conjugate, then they are linearly independent.

**Definition 1.4.3** (quadratic termination property) Let an algorithm be applied to minimize the quadratic function (1.4.1). If it will locate the minimizing point $x^*$ in at most a known finite number of iterations, then the algorithm is said to possess the quadratic termination property.

The following conclusion is well-known:
**Theorem 1.4.4** Let \( d_1, \ldots, d_n \) be a group of conjugate directions. Define the iteration for minimizing the quadratic function (1.4.1) by

\[
x_{k+1} = x_k + \alpha_k d_k, \quad k = 1, \ldots, n
\]

(1.4.2)

where \( x_1 \) is given and \( \alpha_k \) is determined by Rule 1.2.1 (exact line search). Then, for \( i = 1, \ldots, k \), we have

\[
g_{k+1}^T d_i = 0.
\]

(1.4.3)

Furthermore, for \( k = 1, \ldots, n \), \( x_{k+1} \) minimizes (1.4.1) over the linear manifold

\[
M_k = \{ z \mid z = x_1 + \gamma_1 d_1 + \cdots + \gamma_k d_k, \gamma_1, \ldots, \gamma_k \in \mathbb{R}\}
\]

and hence \( x_{n+1} \) minimizes (1.4.1) over \( \mathbb{R}^n \).

Theorem 1.4.4 shows that iterations using conjugate directions with exact line search have the quadratic termination property. Many good algorithms possess this property, although successful methods do exist which do not terminate for quadratic function.
CHAPTER TWO
NEWTON-LIKE METHODS

In unconstrained optimization, one of the important questions is how to choose search direction. According to the different search directions, the algorithms can be classified roughly as Newton-like methods and non-Taylor series methods. In general, Newton-like methods are the more efficient. This chapter outlines Newton-like methods and related features.

2.1 Taylor Series Model

Newton-like methods for solving (UO) rely upon a quadratic model which is obtained from a truncated Taylor series expansion of the objective function around $x_k$. That is,

$$f(x_k + s) = \varphi(x_k + s) \equiv f_k + g_k^T s + \frac{1}{2} s^T G_k s. \quad (2.1.1)$$

To estimate the difference between the quadratic model (2.1.1) and the objective function $f(x)$, making use of the Lipschitz condition, we have
Proposition 2.1.1 If $G \in \text{Lip}_\gamma(N_c)$, where $N_c$ containing $x$ and $x_k$, then it is a standard estimation that

$$|\varphi_k(x) - f(x)| \leq \frac{1}{6} \gamma \|x - x_k\|^2. \quad (2.1.3)$$

2.2 Newton's Method and Its Modifications.

In research in nonlinear optimization, there is a strong suspicion that if any iterative method for any problem in any field is exceptionally effective, then it is Newton's method in some appropriate context. Of course, this does not imply that Newton's method is always practical.

The basic Newton iteration can be derived by applying the first order necessary condition to the quadratic model (2.1.1). That is,

$$\begin{align*}
&\text{(a) solve } G_k s = -g_k, \text{ for } s = s_k, \\
&\text{(b) set } x_{k+1} = x_k + s_k.
\end{align*} \quad (2.2.1)$$

For the Newton iteration, the following convergence theorem is well-known:

Theorem 2.2.1 Assume that $f(x)$ is a twice continuously differentiable function, and that $G \in \text{Lip}_\gamma N(x^*, \gamma)$. If $x_k$ is sufficiently close to $x^*$ for some $k$, and if $G^*$ is positive definite, then Newton's method is well defined for all $k$, and converges with a $q$-order of two.

Theorem 2.2.1 shows that Newton's method converges to $x^*$ with a $q$-order of two if the initial point is sufficiently close to $x^*$ and $G^*$ is positive definite. However, it is not necessarily globally convergent, and it requires $o(n^3)$ operations every iterations if $G_k$ is not sparse. Another disadvantage of Newton's method is the need for the user
to supply an algorithm for evaluating and storing the second derivative matrix $G$. In addition, the main difficulty with Newton's method arises when $G_k$ is not positive definite. For this situation, the quadratic model function need not have a minimum, nor even a stationary point. Global convergence can be improved by paying more attention to the detail of finding the next iterate. For this reason, several schemes have been proposed to modify Newton's method.

Modification 2.2.2 This method consists of the iteration

$$x_{k+1} = x_k + \alpha_k d_k,$$  \hspace{1cm} (2.2.2)

where $\alpha_k$ is chosen by using one of the rules (1.2.1 - 4), and

$$d_k = -G_k^{-1} g_k.$$

This scheme overcomes a local difficulty of Newton's method, i.e., which is not necessarily global convergent when $G_k$ is positive definite for all $k$.

Modification 2.2.3 A modified Newton search direction is given by

$$(G_k + \mu_k I)d_k = -g_k.$$

(2.2.3)

This scheme causes the Newton direction to have a bias towards the steepest descent vector $-g_k$. The key is how to choose $\mu_k$ in (2.2.3) so that the modified matrix $G_k + \mu_k I$ is positive definite, and so that a good direction of search is obtained. It is possible also to regard this type of method as being a form of trust region method (trust region method is described in Section 2.7).
Modification 2.2.4 Adaptively choosing a diagonal matrix $D_k$ so that $G_k + D_k$ is positive definite. Thus, a modified Newton search direction is given by

$$\begin{align*}
(G_k + D_k)d_k &= -g_k.
\end{align*}$$

(2.2.4)

The idea is that during the Cholesky factorization process, we can detect whether $G_k$ is either nonpositive definite or nearly singular, in which case $G_k$ is replaced by a positive definite matrix $G_k + D_k$. The elements of $D_k$ are introduced sequentially during the factorization process. A good implementation of this scheme has been developed by Murray (1972).

Modification 2.2.5 When $G_k$ is indefinite, using a negative curvature direction $d_k$ instead of the Newton direction, in which $d_k$ is chosen so that $g_k^T d_k \leq 0$ and $d_k^T G_k d_k < 0$. This was first suggested by Fiacco and McCormick (1968). Since the Cholesky factorization is unstable if $G_k$ is indefinite, there are several strategies for constructing a negative curvature direction (Fletcher-Freeman 1977, McCormick 1977, Goldfarb 1980 and More–Sorrensen 1979). Here we consider the factorization (BKP) obtained by Bunch, Kaufmann and Parlett (1976)

$$B_k = L_k \bar{D}_k L_k^T,$$

(2.2.5)

where $L_k$ is a lower triangular matrix and $\bar{D}_k$ is a block diagonal matrix with one-by-one or two-by-two blocks in the diagonal. Let $e_1, e_2, \ldots, e_n$ be a set of $\bar{D}_k$ associated with its eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. For $i \in \Delta = \{i | \lambda_i < 0\}$ we define $q_k \in \mathbb{R}^n$ by
\[ L_k^T q_k = \sum_{i \in I} e_i \]  
\[ (2.2.6) \]

such that
\[ q_k^T B_k q_k = (\sum_{i \in I} e_i)^T \tilde{D}_k (\sum_{i \in I} e_i) = \sum \tilde{\lambda}_i < 0. \]

Therefore, a descent direction \( d_k \) can be obtained by
\[ d_k = \beta(-g_k) + (1 - \beta)\hat{q}_k \]  
\[ (2.2.7) \]
where
\[ \hat{q}_k = \begin{cases} q_k, & g_k^T q_k \leq 0 \\ q_k, & g_k^T q_k > 0 \end{cases} \]  
\[ (2.2.8) \]
and \( 0 \leq \beta \leq 1. \)

2.3 Quasi-Newton Methods

In order to avoid calculation of the second derivative matrix \( G \), a class of methods which give symmetric positive definite approximations to \( G \) called quasi-Newton methods have been developed. Consider

\[ \varphi(x_k + d) = f_k + g_k^T d + \frac{1}{2} d^T B_k d, \]  
\[ (2.3.1) \]

where \( B_k \) is a positive definite symmetric matrix, and is regarded as an approximation of \( G_k \). The key requirement of a quasi-Newton method is that \( B_k \) satisfies the quasi-Newton condition
\[ B_k s_{k-1} = y_{k-1}, \]  
\[ (2.3.2) \]
where \( s_{k-1} = x_k - x_{k-1}, y_{k-1} = g_k - g_{k-1}. \)
Algorithm 2.3.1 (principal quasi-Newton method)

Step 0  Given $x_k$ and the symmetric positive definite matrix $B_k$.
Step 1  If termination is achieved, then stop.
Step 2  Determine a search direction $d_k$ by using $B_k d = -g_k$.
Step 3  Find a steplenth $\alpha_k$ such that Rule 1.2.1 or Rule 1.2.4 is satisfies.
Step 4  Set $x_{k+1} = x_k + \alpha_k d_k$. Calculate $s_k$ and $y_k$.
Step 5  Update $B_k$ so that it is symmetric and positive, and so that the quasi-Newton condition (2.3.2) is satisfied for $s_k$ and $y_k$.
Step 6  Set $k = k+1$ and go to Step 1.

2.4 Broyden Family

Quasi-Newton updates have been extensively studied since Davidon (1959) first proposed them for unconstrained optimization. Broyden (1967) established an one-parameter family of quasi-Newton updates

$$
B_{k+1} = B_k + \frac{y_k y_k^T}{s_k y_k^T} - \frac{B_k s_k y_k^T}{s_k^T B_k s_k} + \phi_k (s_k^T B_k s_k) u_k u_k^T
$$

(2.4.1)

where

$$
u_k = \frac{y_k}{s_k y_k^T} - \frac{B_k s_k}{s_k^T B_k s_k}.
$$

(2.4.2)

Obviously, $\phi_k$ characterizes the different methods in the family.

For the sake of convenience, sometimes, we will use the following equivalent representative

$$
H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \rho_k (y_k^T H_k y_k) v_k v_k^T
$$

(2.4.3)

where $H_k = B_k^{-1}$ and
Moreover, $\rho_k$ is obtained from $\phi_k$ through the mapping

$$\rho_k = \frac{\phi_k - 1}{t_k \phi_k - 1},$$

(2.4.5)

where

$$t_k = 1 - \frac{(y_k^T H_k y_k) (y_k^T H_k y_k)}{(s_k^T y_k)^2}.$$  

The Broyden family possesses a number of important properties as follows.

* Positive definite

If $B_k$ is positive definite, $s_k^T y_k > 0$ and $\phi_k > \hat{\phi}_k$, then $B_{k+1}$ is also positive definite. Here $\hat{\phi}_k$ is called the degenerate value for the Broyden family (Fletcher and Sindair, 1979) and is defined as

$$\hat{\phi}_k = \frac{1}{t_k}.$$  

(2.4.6)

* Quadratic termination

For the quadratic objective functions (1.4.1), an algorithm using any updating formula from the Broyden family with exact line search terminates after $m \leq n$ iterations, and the following hold on for $i = 1, 2, ..., m$

$$B_{i+1} s_j = y_j', \quad j = 1, 2, ..., i$$

(2.4.7)

$$s_i^T G s_j = 0, \quad j = 1, 2, ..., i-1.$$  

(2.4.8)
and if \( m = n \), then \( B_{n+1} = G \).

* Independence to change in \( \phi \)

For any continuously differentiable function, all the updates in the Broyden family generate the same sequence \( \{ x_k \} \) when an exact line search is used, and the same initial point and initial matrix are chosen (Dixon 1972).

* Global convergence

Let \( f(x) \) be a strictly convex function. If \( B_1 \) is any positive definite matrix, then the sequence \( \{ x_k \} \) generated by the Broyden family using an exact line search converges to the minimum of \( f(x) \).

* Superlinear convergence

Given \( x_1 \). Assume that the hypotheses stated in global convergence hold, and that \( G \in \text{Lip}_r(\Gamma) \) where \( \Gamma \triangleq \{ x \mid f(x) \leq f(x_1) \} \). Then the sequence \( \{ x_k \} \) generated by the Broyden family using an exact line search converges to \( x^* \) q-superlinearly.

**Remark 2.4.1** The DFP update is the first member in the Broyden family which has been proved to have the superlinear convergence property (Powell 1971). According to Dixon’s result (1972), it is clear that the conclusion also holds for the Broyden updates satisfying the above assumptions.

**Remark 2.4.2** Also the result established by Dennis and More (1974) on q-superlinear convergence of general descent methods, i.e., Theorem 1.3.2, is suitable for quasi-Newton methods.
Huang (1970) proposed a large of updates class with three parameters for updating $H_k$. The Broyden family is the subclass of these, which satisfies quasi-Newton condition (2.3.2) and maintains the symmetry of $H_k$. The above outstanding properties cause it to be used widely for numerical calculation.

2.5 **BFGS, DFP and the Convex Class**

Two important members of the Broyden family are the BFGS formula

$$B_{k+1} = B_k + \frac{y_k y_k^T - B_k s_k s_k^T B_k}{s_k y_k}$$

(2.5.1)

and the DFP formula

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) y_k^T + y_k (y_k - B_k s_k)^T}{s_k y_k} - \frac{s_k^T (y_k - B_k s_k) y_k y_k^T}{(s_k y_k)^2}$$

(2.5.2)

They are obtained by setting $\phi_k = 0$ and 1 respectively in (2.4.1).

It is generally considered that BFGS is the 'best' update in terms of efficiency measured by the numbers of function evaluations and iterations. Although a perfect theoretical explanation still has not been obtained to explain why the BFGS is 'best', some interesting contributions have been made: The DFP and BFGS formulas have the "least-change" characterization of quasi-Newton updates in a weighted norm (Greenstadt 1970, Dennis and Schnabel 1979). Moreover, a strong global convergence result for the BFGS update with inexact line search satisfying Rule 1.2.4 when it is applied to a strictly convex objective function has been obtained by Powell (1976). Powell also shows that the sequence $\{x_k\}$ generated by the BFGS update converges to $x^*$ q-superlinearly given two further assumptions: $G(x^*)$ is a positive definite and $G \in \text{Lip}_\gamma(x^*, \gamma)$. BFGS is the first member in the Broyden family which has been proved to have this nice property. The result does not apply to the DFP update.
The so-called convex class consists of updates with $\phi \in [0,1]$ in the Broyden family. That is,

$$B_{k+1} = (1 - \phi)B_k^{BFGS} + \phi B_k^{DFP},$$

(2.5.3)

where $0 \leq \phi \leq 1$. Most of the research interest on the Broyden family has been focused on the convex class. Probably this is due to a theorem by Fletcher (1970), which states that, no matter how the iterative step is done, the eigenvalues of $G^{1/2}B_k^{-1}G^{1/2}$ converge to one monotonely for quadratic objective functions. Fletcher also shows that this property does not always hold outside the convex class. In addition, the convergence theory on the convex class update has been extended by Byrd, Nocedal and Yuan (1987). They show that Powell's global and $q-$superlinear convergence result for BFGS holds for all the convex class updates except for DFP.

**Remark 2.5.1** The convergence theorem on the updates in Broyden's family that use a negative parameter $\phi$ has been investigated by Zhang and Tewarson (1988). Under the same assumptions as Powell (1976), they prove the global convergence property of the sequence $\{x_k\}$ generated by the Broyden update with $(1 - \nu)\phi_k \leq \phi_k < 0$, where $0 < \nu < 1$. In addition, if $\phi_k < 0$ is suitably chosen at each iteration, the updates may give $q-$superlinear convergence.

### 2.6 Updates with Optimal Condition in the Broyden Family

It is clear that a key question in studying quasi-Newton methods concerns how to select $B_k$. Bertsekas (1981) mentions the following result:

**Theorem 2.6.1** Assume that $f(x)$ is a twice continuously differentiable function, and that a sequence $\{x_k\}$ is generated by Algorithm 2.3.1 with exact line search. If $\lim_{k \to \infty} x_k = x^*$, $g^* = 0$ and $G^*$ is positive, then for $k$ such that $x_k \neq x^*$ we have
\[
\lim_{k \to \infty} \sup_k \frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} \leq \lim_{k \to \infty} \sup_k \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{\lambda_{\text{max}} + \lambda_{\text{min}}}^2,
\]

(2.6.1)

where \(\lambda_{\text{max}}\) and \(\lambda_{\text{min}}\) are the largest and smallest eigenvalues of \(H_k^{1/2}G^*H_k^{1/2}\) respectively.

**Remark 2.6.2** A main insight from Theorem 2.6.1 is that in order to achieve fast convergence, one should try to choose the matrix \(H_k\) sufficiently close to \((G^*)^{-1}\) so that the corresponding maximum eigenvalue \(\lambda_{\text{max}}\) and minimum eigenvalue \(\lambda_{\text{min}}\) of \(H_k^{1/2}G^*H_k^{1/2}\) satisfy \(\lambda_{\text{max}} / \lambda_{\text{min}} = 1\). This comment will be important when quadratic termination properties of the main algorithm are considered in Chapter five.

Following this idea, a strategy that possesses numerical meaning directly is established by Davidon (1975). By minimizing the condition number of \(H_k^{1/2}H_{k+1}^{-1}H_k^{1/2}\), he obtained the following result: Let \(H_k\) be a symmetric positive definite matrix. Then among all positive definite matrices \(H_{k+1}\) which satisfy (2.4.3), \(\text{Cond}(H_k^{1/2}H_{k+1}^{-1}H_k^{1/2})\) is minimized by \(H_{k+1}(\rho_{oc})\), where

\[
\rho_{oc} = \begin{cases} 
\frac{s^T y (s^T B s - s^T y)}{(y^T H y) (s^T B s) - (s^T y)^2} & \text{if } s^T y \leq 2(y^T H y) (s^T B s), \\
\frac{s^T y}{s^T y - y^T H y} & \text{if } s^T y > 2(y^T H y) (s^T B s).
\end{cases}
\]

(2.6.2)

An "optimal conditioning" strategy for convex class updates is due to Schnabel (1978). If \(s_k^T y_k \leq y_k^T H_k y_k\) and \(s_k^T y_k \leq s_k^T B_k s_k\), the optimally conditioned update in the convex class is the same as Davidon's strategy. If \(y_k^T H_k y_k < s_k^T B_k s_k\) (or \(y_k^T H_k y_k > s_k^T B_k s_k\)), the BFGS (DFP) is best conditioned in the convex class, and
\[
\begin{align*}
\text{Cond}(\text{BFGS}) & < \frac{\alpha}{\beta} \text{ Cond (OC)}, \quad y_k^T H_k y_k < s_k^T B_k s_k, \\
\text{Cond (DFP)} & < \frac{\alpha}{\beta} \text{ Cond (OC)}, \quad y_k^T H_k y_k > s_k^T B_k s_k.
\end{align*}
\]

where \text{Cond (OC)} denotes the condition number with Davidon's strategy.

As an application of Davidon's criterion, a strategy for choosing the initial Hessian approximation in quasi-Newton methods is proposed by Shanno-Phua (1978). In fact, the choice of an initial approximation \( H_1 \) will influence the numerical results of a quasi-Newton algorithm. If \( H_1 = I \) is set, we do not consider the scale of \( f(x) \). For this reason, \( \|H_1\| \) may be different from \( \|G_1\| \) by many orders of magnitude. This can cause the algorithm to require additional iterations. To overcome this disadvantage, Shanno-Phua (1978) propose a pre-scaling strategy. In this scheme, a scaling factor \( \hat{\theta} \) is defined by

\[
\hat{\theta} = \frac{(s_1^T y_1)(s_1^T s_1)}{(s_1^T y_1)^2 + [(y_1^T y_1)(s_1^T s_1) - (s_1^T y_1)^2] \rho}. \tag{2.6.3}
\]

Given \( \rho \) in (2.3.3), an optimal initial value can be obtained to scale \( H(\rho) \). For example, for BFGS with \( \rho = 1 \), \( \hat{\theta} = s_1^T y_1 y_1^T y_1 \) yields the initial scaling

\[
\hat{H}_1 = (s_1^T y_1 y_1^T y_1) H_1. \tag{2.6.4}
\]

Computational experience shows that the update (2.6.4) is effective.

2.7 Trust Region Methods

The "trust region" strategy provides a way to overcome the difficulty caused by non-positive definite Hessian matrices in Newton's method. The basic idea is that the step is restricted by the region of validity of the Taylor series. Given \( x_k \in \mathbb{R}^n \), consider the subproblem
Minimize \[ \phi_k(s) = f_k + g_k^T s_k + \frac{1}{2} s_k^T B_k s_k \] \hspace{1cm} (2.7.1)

Subject to \[ \|s_k\| \leq \Delta_k. \]

where \( B_k \) is the Hessian of \( f(x_k) \) at \( x_k \) or an approximation to it, and \( \Delta_k \) is the trust radius. The iteration consists of solving (2.7.1), and then comparing the actual reduction of the objective function
\[ \text{ared}_k = f_k - f(x_k + s_k) \] \hspace{1cm} (2.7.2)

to the reduction predicted by the quadratic model
\[ \text{pred}_k = f_k - \phi(s_k). \] \hspace{1cm} (2.7.3)

If the reduction is satisfactory then the step can be taken and a larger trust region tried. Otherwise the trust region is reduced and the minor iteration is repeated. For the \( k \)-th iteration, a basic algorithm is as follows.

**Algorithm 2.7.1** (the classical one for the \( k \)-th iteration)

Step 0 Let \( 0 < \mu_1 < \mu_2 < 1 \) and \( 0 < \eta_1 < 1 < \eta_2 \) be specified. Given \( x_k \) and \( \Delta_k > 0 \).

Step 1 Calculate \( f_k \) and \( g_k \). If the condition of termination is achieved, then stop.

Step 2 Calculate the matrix \( B_k \).

Step 3 Solve the subproblem (2.7.1) and obtain \( s_k \).

Step 4 Calculate \( f(x + s_k) \) and \( t_c = \text{ared}_k / \text{pred}_k \).

Step 5 If \( t_c < \mu_1 \), \( \Delta_{k+1} = \eta_1 \Delta_k \); if \( \mu_2 < t_c \) and \( \|s_k\| = \Delta_k \), \( \Delta_{k+1} = \eta_2 \Delta_k \); otherwise \( \Delta_{k+1} = \Delta_k \).

Step 6 If \( t_c \leq 0 \), set \( x_{k+1} = x_k \); else set \( x_{k+1} = x_k + s_k \). \[ \square \]
A particular advantage of the trust region method is that strong convergence results can be established. Theorem 2.7.2 gives the main global and local convergence properties of a method based on the trust region iteration of Algorithm 2.7.1. Similar results may be found in Fletcher (1980), Gay (1981) and Sorensen (1982).

**Theorem 2.7.2** Let \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) be twice continuously differentiable and bounded below. Also, for \( x_0 \in \mathbb{R}^n \) and some \( \beta_1, \beta_2 > 0 \), let \( G(x) \) be uniformly continuous and satisfy \( \|G(x)\| \leq \beta_1 \) for all \( x \) with \( f(x) \leq f(x_0) \). Let \( \{x_k\} \) be the sequence produced by iterating Algorithm 2.7.1 starting from \( x_0 \), and using \( B_k = G(x_k) \) or any symmetric approximation with \( \|B_k\| \leq \beta_2 \) at each iteration, and the exact solution to (2.7.1) to calculate \( s_k \). Then \( \lim_{k \to \infty} \|g_k\| = 0 \). If in addition each \( B_k = G(x_k) \), then for any limit point \( x^* \) of the sequence \( \{x_k\} \), \( g^* = 0 \) and \( G(x^*) \) is at least positive semi-definite. Furthermore if each \( B_k = G(x_k) \), then if \( \{x_k\} \) converges to \( x^* \), \( G(x^*) \) is positive definite, and \( G(x) \) is Lipschitz continuous around \( x^* \), then the rate of convergence is \( q \)-order of two.

Theorem 2.7.2 shows that a trust region method which solves the subproblem (2.7.1) exactly has attractive convergence properties. However the ideal trust region step is difficult to calculate. For this reason, two general classes of efficient methods for approximately solving (2.7.1) have arisen, namely approximate optimal step methods (see Hebden 1973, More 1978, Gay 1981, Sorensen 1982 and More - Sorensen 1983) and dogleg methods (see Powell 1970 and Dennis - Mei 1979). The convergence analyses of trust region methods with these approximate optimal steps is subsumed by Theorem 2.7.3 below, due to Shultz, Schnabel and Byrd (1985).

**Theorem 2.7.3** Under the conditions of Theorem 2.7.2, If there exist \( \eta_1, \eta_2 > 0 \) such that each \( s_k \) satisfies

\[
g_k^T s_k + 1/2 s_k^T B_k s_k \leq -\eta_1 \|g_k\| \min\{\Delta_k, \eta_2 \|g_k\|/\|B_k\|\},
\]

(2.7.4)
then \( \lim_{k \to \infty} ||g_k|| = 0 \). If in addition each \( B_k = G(x_k) \) and there exists \( \eta_3 > 0 \) such that each \( s_k \) satisfies

\[
g_k^T s_k + 1/2s_k^T B_k s_k \leq -\eta_3(-\lambda_1(B_k))\Delta_k^2,
\]

(2.7.5)

where \( \lambda_1(B_k) \) denotes the smallest eigenvalue of \( B_k \), then for any limit point \( x^* \) of \( \{x_k\} \), \( g^* = 0 \) and \( G(x^*) \) is at least positive semi-definite. Also, if each \( B_k = G(x_k) \), each \( s_k \) satisfies (2.7.4), and there exists \( \eta_4 \in (0,1] \) such that \( s_k = -B_k^{-1}g_k \) whenever \( B_k \) is positive definite and \( ||B_k^{-1}g_k|| \leq \eta_4 \Delta_k \), then if \( \{x_k\} \) converges to \( x^* \), and \( G(x^*) \) is positive definite and is Lipschitz continuous around \( x^* \), then the rate of convergence has q-order of two.
CHAPTER THREE

METHODS FOR LARGE SCALE OPTIMIZATION

As a current research direction, an increasing amount of effort has been directed toward providing numerical methods for large scale optimization problems. The methods usually depend on the sparsity structure in the Hessian of the problem. If the problem has a nice enough sparsity structure, our first choice would be to use a Newton or finite difference Newton iteration. If not, we need to choose an algorithm to obtain the solution of the problem effectively. This chapter outlines a number of popular methods which are suitable in the second case.

3.1 Conjugate Gradient Methods

It is well known that the conjugate gradient methods are an important group of methods for solving large scale unconstrained optimization problems because they do not require the matrix storage of Newton-like methods which form Hessian estimates explicitly. The basic form of its search direction is given by

\[
\begin{align*}
    d_1 &= -g_1 \\
    d_k &= -g_k + \beta_{k-1} d_{k-1},
\end{align*}
\]

where

\[
\beta_{k-1} = \frac{g_k^T y_{k-1}}{g_{k-1}^T y_{k-1}},
\]

(3.1.2)

Remark 3.1.1 As alternatives to (3.1.2), Fletcher and Reeves (1964) suggest

\[
\beta_{k-1} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}},
\]

(3.1.3)

while Polak and Ribiere (1969) suggest
In fact, all of them are equivalent for the quadratic function (1.4.1) with an exact line search. Moreover (3.1.2) and (3.1.4) are equivalent for non-quadratic functions when exact line searches are done. An interesting convergence result is obtained by Al-Baali (1985).

\[ \beta_{k-1} = \frac{g_k^T y_{k-1}}{g_{k-1}^T g_{k-1}}. \]  

(3.1.4)

Theorem 3.1.1 Assume that \( f(x) \) is a twice continuously differentiable, and that the level set \( \Gamma \triangleq \{ x \mid f(x) \leq f(x_1) \} \) is bounded. If \( \{x_k\} \) are generated by the Fletcher-Reeves conjugate gradient method with Rule 1.2.4, then \( \lim_{k \to \infty} \inf ||g_k|| = 0 \).

For the quadratic objective function (1.4.1), a conjugate gradient algorithm with an exact line search has some attractive properties: (1) the successive search direction are conjugate; (2) the successive gradient vectors are orthogonal; (3) it terminates in at most \( n \) steps. Unfortunately, these are properties which hold only for quadratic functions and are not true for general functions. They are eroded by rounding errors in real computation. It follows that there is a need for techniques which can be used to improve the general convergence properties of the method.

The restarting strategy is one of the schemes which improve the classical conjugate gradient method. The success of such a strategy depends on two factors. One is the restart criterion, and the other is the restart direction. The first restarting strategy was proposed by Fletcher and Reeves (1964). They suggested beginning again every \( n \) or \( (n+1) \) steps by setting \( d_k = -g_k \). Subsequently, Powell (1977) proposed an automatic criterion for restarting that has proved to be more effective. The idea is that as long as the objective function resembles a quadratic in the neighborhood of \( x_k \), we should continue with the conjugate gradient iteration. Otherwise, we should restart. To measure the deviation from quadratic behaviour Powell uses the radio
The algorithm is restarted when $\tilde{r}_k > 0.2$. As a restart direction, Powell suggests using the scheme of Beale (1972). Let the $k$-th search direction be

$$d_k = -g_k + \beta_{k-1}d_{k-1} + \gamma_{k-1}d_t,$$

where $k > t$, $d_t$ is a descent direction, $\beta_{k-1}$ is defined by (3.1.2) and $\gamma_{k-1} = g_k^T y_t / d^T_t y_t$ ($\gamma_{k-1}$ is zero when $k = t+1$). Once a restart is requested, $d_t$ is taken as a restart direction.

To increase the effectiveness of the restarting strategy it is a reasonable strategy to try to retain more information on the objective function. This leads us to another modifications of the conjugate gradient method, in which preconditioning techniques play an important role. Let $M$ is a symmetric and positive definite matrix, a preconditioned conjugate gradient (PCG) direction is given by

$$
\begin{cases}
    d_1 = -M g_1 \\
    d_k = -M g_k + \frac{y_{k-1}^T M g_k}{y_{k-1}^T d_{k-1}^T} d_{k-1}
\end{cases}
$$

(3.1.6)

Here $M$ is called the preconditioner and should be chosen so that the condition number of $(M^{-1/2})G_k(M^{-1/2})^T$ is as small as possible.

A third scheme which has been investigated for improving on the conjugate gradient methods is the supermemory strategy. The main idea is that of combining a descent direction with the displacements generated by previous iterations to get a new search direction. The typical form of the method is shown by Wolfe and Viazminsky (1976). That is, for the $k$-th iteration, calculate $\alpha_k$, $\beta^{(i)}_k$, $s_k$ and $x_{k+1}$ from
where $p_k$ is a basic search direction and $m$ is the number of memory terms. For the quadratic function (1.4.1), the iteration (3.1.7) with exact line search terminates in a finite number of steps. Choosing different $p_k$, we obtain different supermemory descent algorithms. Numerical experience shows that they are more rapidly convergent than quasi-Newton methods, in general. The major weakness of this class of methods is the computational labour required to perform the $(m+1)$-dimensional search at each iteration.

### 3.2 Quasi-Newton Methods with Limited Memory

Limited memory quasi-Newton methods developed during the last ten years are suitable for large scale optimization problems because the amount of storage required by the algorithm can be controlled by the user. Another main feature of the methods is that they do not require knowledge of the sparsity structure of the Hessian, or knowledge of the separability of the objective function. Limited memory quasi-Newton methods consist of two types of development: one is the combined CG-QN method, the other is the limited memory BFGS method. Numerical tests have shown that these methods can be very competitive.

The combined CG-QN methods can be seen as extensions of the conjugate gradient method, in which additional storage is used to store limited prior information so that a variable metric preconditioner is formed. A typical combined CG-QN algorithm is due to Buckley and LeNir (1983), where it is called the variable storage
conjugate gradient method (VSCG). The basic idea is to combine cycles of BFGS and conjugate gradient steps. It starts by performing the usual BFGS method, but stores the corrections to the initial matrix separately to avoid using $O(n^2)$ storage. When the available storage is used up, the current BFGS matrix is used as a fixed preconditioner, and the method performs preconditioned conjugate gradient steps based on the three-term recurrence of Beale (1972). These steps are continued until the criterion of Powell (1977) indicates that a restart is desirable; all BFGS corrections are then discarded and the method performs a Beale restart. Thus a new BFGS cycle begins. For more details see Buckley and LeNir (1983). The following result is true.

**Theorem 3.2.1** Algorithm VSCG with exact line search and without restarting possesses the quadratic termination property.

The limited memory BFGS method (L-BFGS) is described by Nocedal (1980). It can be viewed as an implementation of BFGS method, in which storage is restricted. It is almost identical in its implementation to the well known BFGS method. The only difference is in the matrix update: the BFGS corrections are stored separately, and when the available storage is used up, the oldest correction is deleted to make space for the new one. The concrete algorithm is given as follows.

**Algorithm 3.2.2 (L-BFGS)**

- **Step 0** Let the amount of memory $m$ be specified. Given an initial point $x_0$ and a symmetric positive definite starting matrix $H_0$. Set $k = 0$.
- **Step 1** If the termination criterion is achieved, then stop.
- **Step 2** calculate $d_k = -H_k g_k$.
- **Step 3** Choose $\alpha_k$ so that it satisfies Rule (1.2.4).
Step 4 Let \( m_0 = \min(k, m - 1) \), \( \eta_k = 1/s_k y_k^T \), and \( P_k = I - \eta_k y_k s_k^T \). Update \( H_o \) \( m_0 + 1 \) times using the pairs \( \{y_j, s_j\}_{j=k-m_0}^k \), i.e., set

\[
H_{k+1} = (P_k^T \ldots P_{k-m_0}^T)H_0 (P_{k-m_0} \ldots P_k) \\
+ \eta_{k-m_0} (P_k^T \ldots P_{k-m_0+1}^T) s_{k-m_0} s_{k-m_0}^T (P_{k-m_0+1} \ldots P_k) \\
+ \eta_{k-m_0+1} (P_k^T \ldots P_{k-m_0+2}^T) s_{k-m_0+1} s_{k-m_0+1}^T (P_{k-m_0+2} \ldots P_k) \\
+ \ldots \\
+ \eta_k s_k s_k^T.
\]

Step 5 Set \( k = k + 1 \) and go to Step 1.

A convergence analysis of Algorithm 3.2.2 is given by Liu and Nocedal (1988).

**Theorem 3.2.3** Assume that \( f(x) \) is a twice continuously differentiable, and that the level set \( \Gamma = \{ x \mid f(x) \leq f(x_o) \} \) is convex. If \( f(x) \) is a strict convex function over \( \Gamma \), then for any positive \( H_o \), the sequence \( \{x_k\} \) generated by Algorithm 3.2.1 converges to \( x^* \) \( q \)-linearly.

**Remark 3.2.4** In Theorem 3.2.3, if \( H_o \) be scaled, i.e., \( H_k^{(o)} = (s_k^T y_k / y_k^T y_k)H_o \), then the result still holds so long as \( \|H_k^{(o)}\| \) and \( \|(H_k^{(o)})^{-1}\| \) are uniformly bounded for all \( k \).
In unconstrained optimization, a special place is occupied by objective functions which can be represented by a sum of squared terms. This is the so-called nonlinear least squares problem. There are particular ways to exploit this structure. Numerical methods for them are discussed in this chapter.

4.1 Preliminary

The nonlinear least squares problem (NLS) is

\[
\text{Minimize } f(x) = \frac{1}{2} \sum_{i=1}^{m} [r_i(x)]^2, \tag{4.1.1}
\]

where each \( r_i(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is a continuously differentiable function. The gradient and Hessian of \( f(x) \) have a special form. They are given by

\[
g(x) = J(x)^T r(x) \tag{4.1.2}
\]

and

\[
G(x) = J(x)^T J(x) + Q(x), \tag{4.1.3}
\]

where \( J(x) \) is the \( m \times n \) Jacobian matrix whose \((i,j)\)th element is \( J_{ij} = \frac{\partial r_i(x)}{\partial x_j} \), \( r(x)^T = [r_1(x), \ldots, r_m(x)] \) and \( Q(x) = \sum_{i=1}^{m} r_i(x) V^2 r_i(x) \).

Obviously, the Hessian matrix of \( f(x) \) is composed of two parts: a first order term containing only the first derivatives of \( r(x) \) and a second order one containing the second derivatives of \( r(x) \). \( r(x) \) usually is said to be the residual. A NLS problem for which \( r(x^*) = 0 \) is called a zero-residual problem.
Most methods for solving NLS problem are based on variants of Newton's method. The quadratic model of $f(x)$ around $x_k$ is

$$
\varphi_k(s) = \frac{1}{2} r_k^T r_k + (J_k^T r_k)^T s + \frac{1}{2} s^T B_k s,
$$

(4.1.4)

where $B_k$ is equal to $G_k$, or it is an approximation to $G_k$. According to the approximation for $G_k$, the methods solving (4.1.1) can be classified roughly as three classes: Gauss-Newton-type methods, all Newton-type methods and hybrid methods.

### 4.2 Gauss-Newton-type Methods

It is well-known that the Guass-Newton method (GN) is a classical algorithm for solving (4.1.1), in which

$$B_k = J_k^T J_k.
$$

(4.2.1)

The corresponding search direction $d_k$ is obtained by solving

$$(J_k^T J_k) d = -J_k^T r_k,
$$

(4.2.2)

and the new point is given by

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

(4.2.3)

This approximation is reasonable in a zero or small residual problem. The point of view has theoretical support in a result due to Meyer (1970). He proved that, for the sequence $\{x_k\}$ generated by the GN iterations (4.2.2-3) in a neighbourhood of $x^*$, it holds that
\[
\lim \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq \frac{\lambda_Q^{(\text{max})}}{\lambda_J^{(\text{min})}} = \mu_c,
\]

where \(\lambda_Q^{(\text{max})}\) is the maximum absolute eigenvalue of \(Q(x^*)\) and \(\lambda_J^{(\text{min})}\) is the smallest eigenvalue of \(J(x^*)^TJ(x^*)\). The sequence \(\{x_k\}\) is q-order of two convergent to \(x^*\) when \(r(x^*) = 0\). If \(\mu_c = 0\), it is at least superlinearly convergent. If \(\mu_c < 1\), it is locally q-linearly convergent. However, the convergence is not guaranteed when \(\mu_c > 1\). Another main disadvantage is that the GN method is not well defined if \(J_k\) does not have full column rank. In addition, it is not necessarily globally convergent.

If \(J_k\) has full column rank, \(d_k\) is well defined from (4.2.2) and it is a descent direction. This suggests two ways of improving the GN method: using it with a line search or with a trust region strategy.

Modification 4.2.1 (damped Gauss-Newton method) Let \(d_k\) be obtained by (4.2.2). Set
\[
x_{k+1} = x_k + \alpha_k d_k,
\]
where \(\alpha_k\) is determined by Rule (1.2.4).

If \(d_k\) is well defined by (4.2.2) for all \(k\), the damped GN method always takes descent steps that satisfy the line search criterion. By theorem 1.2.8, it is usually globally convergent. However, it may still be very slowly convergent on large-residual problems.

Modification 4.2.2 (Levenberg-Marquardt method) Let a descent step \(s_k\) be given by the trust region approach:
The solution to (4.2.6) is

\[ x_{k+1} = x_k - (J_k^TJ_k + \mu_k I)^{-1}J_k^Tr_k, \]  

(4.2.7)

where \( \mu_k = 0 \) if \( \Delta_k \geq \| (J_k^T J_k)^{-1} J_k^T r_k \| \) and \( \mu_k > 0 \) otherwise.

A good implementation of (4.2.7) as a trust region algorithm is due to More (1977). Two factors make the Levenberg-Marquardt method preferable to the damped GN method on many problems. One is that the Levenberg-Marquardt method is well defined even when \( J_k \) does not have full column rank. Another is that when the GN step is much too long, the Levenberg-Marquardt step is close to being in the steepest descent direction \(-J_k r_k\). Several versions of the Levenberg-Marquardt method have been proved globally convergent, for example, Powell (1975), Osborne (1976) and More (1977).

### 4.3 All Newton-type Methods

The quasi-Newton methods described in Chapter 2 also can be used to solve a NLS problem directly, and a search direction \( d_k \) is determined by

\[ B_k d = -J_k r_k, \]  

(4.3.1)

where \( B_k \) satisfies

\[ B_k s_{k-1} = y_{k-1} \]

with \( s_{k-1} = x_k - x_{k-1} \) and \( y_{k-1} = J_k^T r_k - J_{k-1}^T r_{k-1} \).
On the other hand, in order to approximate to $G_k$ more robustly, assume that $J_k$ can be stored in the computer. We only need to consider how to approximate to $Q_k$. That is,

$$G_k = B_k = J_k^T J_k + E_k,$$

(4.3.2)

where $E_k$ is an approximation to $Q_k$. Thus the corresponding search direction $d_k$ is defined by

$$(J_k^T J_k + E_k) d = -J_k^T r_k.$$

(4.3.3)

A quasi-Newton technique usually is used to form such an approximation.

**Strategy 4.3.1** (Brown-Dennis, 1971) $E_k$ is a combination of matrices which approximate to each $V_i^2 r_i(x_k)$ by a quasi-Newton approximation $E_i^{(k)}$ respectively. That is,

$$E_k = \sum_{i=1}^{m} r_i E_i^{(k)},$$

and $E_i^{(k)}$ is derived from $E_i^{(k-1)}$ using a quasi-Newton updating formula so that each $E_i^{(k)}$ satisfies

$$E_i^{(k)} s_{k-1} = y_{k-1}^{(i)},$$

where $y_{k-1}^{(i)} = V_i r_i(x_k) - V_i r_i(x_{k-1}).$

The strategy has good convergence properties (Dennis, 1973). However, this is mainly of theoretical interest since it requires the storage of $m$ matrices in which each one contains $n(n+1)/2$ elements.
Strategy 4.3.2 (Dennis, 1973) $E_k$ is derived from $E_{k-1}$ using Powell's symmetric rank 2 update (Powell, 1970), in which $E_k$ satisfies

$$E_{k_{k-1}} = \tilde{y}_{k-1},$$

where $\tilde{y}_{k-1} = y_{k-1} - J_{k_{k-1}}^T s_{k-1}.$

Strategy 4.3.3 (Betts, 1975) $E_k$ is derived from $E_{k-1}$ using Davidon's symmetric rank 1 update (Davidon, 1963), in which $E_k$ satisfies

$$E_{k_{k-1}} = \tilde{y}_{k-1},$$

where $\tilde{y}_{k-1} = y_{k-1} - J_{k_{k-1}}^T J_{k_{k-1}}^T s_{k-1}.$

Strategy 4.3.4 (Dennis–Gay–Welsch, 1981) Choosing $E_k$ to be a solution of the least-change problem

\[ \begin{align*}
\text{minimize} & \quad \|W^T(E - E_{k-1})W^{-1}\|_F \\
\text{subject to} & \quad (E - E_{k-1})^T = E - E_{k-1} \\
E_{k_{k-1}} & = \tilde{y}_{k-1},
\end{align*} \]

where $W \in \mathbb{R}^{nxn}$ is nonsingular and $\tilde{y}_{k-1} = (J_{k} - J_{k_{-1}})^T r_k.$ If $W$ is chosen so that

$$WW^T s_{k-1} = y_{k-1},$$

then

$$E_k = E_{k-1} + \frac{u_{k-1} y_{k-1}^T}{y_{k-1}^T s_{k-1}} + \frac{y_{k-1} u_{k-1}^T}{y_{k-1}^T s_{k-1}} - \frac{u_{k-1}^T s_{k-1}}{(y_{k-1}^T s_{k-1})^2} \tilde{y}_{k-1} y_{k-1}^T$$

(4.3.4)
where \( u_{k-1} = y_{k-1}^T - E_{k-1}s_{k-1} \). To make the algorithm globally convergent, a model with the trust region strategy is used; at each iteration the problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} - T_{k,k}^T r_k + (J_{k,k}^T)^T s + \frac{1}{2} s^T (J_{k,k}^T + E_k) s \\
\text{subject to} & \quad \|s\| \leq \Delta_k
\end{align*}
\]  

\tag{4.3.5}

is solved for \( s_k \), so that

\[
 s_k = -(J_{k,k}^T + E_k + \mu_k J_{k,k}^{-1})^T r_k
\]  

\tag{4.3.6}

for some \( \mu_k \geq 0 \).

**Remark 4.3.5** Strategy 4.3.4 is implemented in the code NL2SOL which is due to (Dennis–Gay–Welsch, 1981) In this code, \( E_0 \) is a zero matrix so that initially the model (4.3.5) is the same as model (4.2.6). Moreover, the algorithm is transformed from the model (4.2.6) to the model (4.3.5) adaptively by a switch that compares the predicted reductions of two models and takes the best one which matches the actual reduction from \( x_{k-1} \) to \( x_k \). Similar to the Shanno-Phua initial scaling strategy, before each update, \( E_{k-1} \) in (4.3.4) is multiplied by the scaling factor

\[
\ell_k^\# = \min \left\{ \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T E_{k-1} s_{k-1}}, 1 \right\}.
\]  

\tag{4.3.7}

**Strategy 4.3.6** (Dennis–Martinez–Tapia, 1989) Set

\[
E_k = E_{k-1} + \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} - \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}}
\]  

\tag{4.3.8}
where $\bar{y}_{k-1} = J_{k}^T J_{k} s_{k-1} + (J_{k} - J_{k-1})^T r_{k}$ and $v_{k-1} = (J_{k}^T J_{k} + E_{k-1}) s_{k-1}$.

**Remark 4.3.7** Strategy 4.3.6 is a current version of the NL2SOL code. The formula (4.3.8) is due to Al-Baali and Fletcher (1985). In fact, (4.3.4) is a "DFP" type formula, but (4.3.8) is a "BFGS" type formula. For (4.3.8), superlinear convergence can be proved.

**Theorem 4.3.8** (Daniels-Martinez-Tapia, 1989) Assume that $f(x)$ is a twice continuously differentiable function, and that $J(x) \in \text{Lip}_{\gamma_1} N(x^*, \gamma_1)$ and $G(x) \in \text{Lip}_{\gamma_2} N(x^*, \gamma_2)$. If $G(x^*)$ is positive definite, then, there exist positive constants $\varepsilon, \delta$ such that, for $x_o \in \mathbb{R}^n$ and symmetric $E_o \in \mathbb{R}^n$ satisfying $\|x_o - x^*\| < \varepsilon$ and $\|E_o - Q(x^*)\| < \delta$, the iteration sequence $\{x_k\}$ generated by (4.3.8) for problem (4.1.1) is q–superlinearly convergent to $x^*$.

### 4.4 Hybrid Methods

Numerical experiments show that the GN method is better than a quasi-Newton method for a zero–residual problem, whereas a quasi-Newton method is preferable for a large–residual problem or when $J(x)$ loses rank at the solution. For small–residual problems the GN method converges at a fast linear rate which in limited precision may be preferable to the superlinearly convergent quasi-Newton method. To have both advantages of the GN method and the quasi-Newton method at the same time, hybrid methods are developed. The key for the hybrid methods is how to choose a switch so that the size of residual is distinguished.

**Strategy 4.4.1** (Al-Baali-Fletcher 1985) An approximation to $G_k$ is made by the following principle:
Use

\[
B_k = \begin{cases} 
  B_k^{\text{BFGS}}, & \text{if } \Delta(B_{k-1}) < \Delta(J_k^T J_k), \\
  J_k^T J_k, & \text{otherwise},
\end{cases} \quad (4.4.1)
\]

and for an arbitrary positive definite matrix \( M \in \mathbb{R}^{n \times n} \), \( \Delta(M) \) is defined by

\[
\Delta(M) \triangleq (t_1^2 - 2t_2 + 1)^{1/2}, \quad (4.4.2)
\]

where

\[
t_1 = \frac{\bar{y}_{k-1}^T M^{-1} \bar{y}_{k-1}}{s_{k-1}^T \bar{y}_{k-1}} \quad \text{and} \quad t_2 = \frac{s_{k-1}^T \bar{y}_{k-1}}{s_{k-1}^T M s_{k-1}}.
\]

**Remark 4.4.2** In fact, the definition (4.4.2) denotes the minimum value of the variational problem

\[
\text{minimize} \quad \|B_k^{1/2}(B_k^{-1} - M)B_k^{1/2}\|_F
\]

subject to

\[
(B_k^{-1})^T = B_k^{-1},
\]

\[
B_k s_{k-1} = \bar{y}_{k-1}
\]

at the solution, where \( B_k \) is derived from \( M \) using the BFGS update.

A more interesting suggestion for distinguishing the size of residuals is due to Fletcher and Xu (1987).

**Strategy 4.4.3** (Fletcher–Xu, 1987) An approximation to \( G_k \) is made by the following principle:

\[
B_k = \begin{cases} 
  B_k^{\text{BFGS}}, & \text{if } (f_k - f_{k-1})/f_{k-1} < \varepsilon, \\
  J_k^T J_k, & \text{otherwise},
\end{cases} \quad (4.4.3)
\]
Strategy 4.4.3 is a simple and effective scheme for NLS problem. Also superlinear convergence of an algorithm which uses strategy 4.4.3 has been proved (Fletcher and Xu, 1987).
CHAPTER FIVE
THE SYMMETRIC RANK-ONE UPDATE

A stabilized version of the symmetric rank one updating method (SSR1) for solving the unconstrained optimization problem was developed. The study shows that the iterates generated by SSR1 converge to the true Hessian matrix under certain conditions. Specifically, if the optimality condition (CGSSR1) has a lower-approximation property for linear programming problems, then the numerical results show that SSR1 can perform well in comparison with other methods. The numerical results are based on the so-called ill-conditioned and singular optimization problems. In fact, preliminary tests show that the new method compares favorably with the BFGS method.

II History
In unconstrained optimization, the simplest modification of the matrix \( J \) satisfying the quasi-Newton condition is the rank-one formula. The first rank-one update was due to Broyden (1967). He suggested

\[
H_{k+1} = H_k + \beta_k \gamma_k \nabla \eta_k
\]

where \( \gamma_k \in \mathbb{R}^n \) and \( \beta_k \neq 0 \). Afterwards, Broyden and Johnson (1973) proposed a two-parameter class of updates which is a subset of the Hager (1972) class.

\[
H_{k+1} = H_k + \frac{\beta_k \gamma_k \nabla \eta_k \nabla \eta_k^T}{\gamma_k \nabla \eta_k^T + \beta_k \gamma_k \nabla \eta_k^T + \beta_k \gamma_k \nabla \eta_k^T}
\]
CHAPTER FIVE

THE SYMMETRIC RANK-ONE UPDATE

A stabilized version of the symmetric rank one updating method (SSR1) for solving the unconstrained optimization problem is developed. The study shows that the matrices generated by SSR1 converge to the true Hessian matrix under conditions commonly assumed, and that the new algorithm with Davidon's optimal condition (OCSSR1) has a finite termination property for inexact line searches. The numerical results show that it can perform well on the so-called ill-conditioned and singular optimization problems. In fact, preliminary tests show that the new method compares very favourably with the BFGS method.

5.1 History

In unconstrained optimization, the simplest modification of the matrix $H$ satisfying the quasi-Newton condition is the rank-one formula. The first such update was due to Broyden (1967). He suggested

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)q_k^T}{q_k^T y_k},$$

(5.1.1)

where $q_k \in \mathbb{R}^n$ and $q_k^T y_k \neq 0$. Afterwards, Broyden and Johnson (1973) proposed a two-parameter class of updates which is a subset of the Huang (1970) class:

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(\hat{\theta}_1 H_k^T y_k + \hat{\theta}_2 s_k)^T}{(\hat{\theta}_1 H_k^T y_k + \hat{\theta}_2 s_k)^T y_k},$$

(5.1.2)
where $\hat{\theta}_1$ and $\hat{\theta}_2$ are constants. Two special updates of the class (5.1.2) are attributed to Pearson with $\hat{\theta}_1 = 1$, $\hat{\theta}_2 = 0$ and to McCormick with $\hat{\theta}_1 = 0$, $\hat{\theta}_2 = 1$.

In view of the symmetry of the Hessian matrix, however, it is reasonable to choose $q$ so that $H_+$ has this property. For example, if $q_k = s_k - H_k y_k$ in (5.1.1), we obtain the classical SR1 update

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k},$$

(5.1.3)

which was suggested independently by Broyden (1967), Davidon (1968), Fiacco and McCormick (1968), Murtagh and Sargent (1969), and Wolfe (1968). It is a member of the Broyden family, too. In fact, the SR1 update is obtained if we set

$$\rho_k = s_k^T y_k / (s_k - H_k y_k)^T y_k,$$

(5.1.4)

in (2.4.3). The SR1 update is simple in structure, and requires less calculation per iteration than other quasi-Newton updating formula. In addition, when applied to a quadratic objective functions, it possesses a finite termination property which does not depend on exact line searches. However, it also has what has been regarded as the serious weakness that, even if $H_k$ is positive definite, $H_{k+1}$ can be singular, indefinite or undefined. Because it has both attractive properties and this weakness not to be ignored, the SR1 update has received considerable attention since it was first proposed.

Over the years, it seems that the investigation of the symmetric rank-one update was concentrated mainly on how to overcome the weakness of SR1. In general, there are two ways to do this: one is to propose a modified version of SR1 based on the theory of the SR1 formula (Norris and Gerken 1977), the other is to use a scaling factor for controlling the positive definiteness of $H$ (Kleinmichel 1981, Spedicato 1983 and
Numerical experiments show that these methods give algorithms possessing more stable numerical behaviour, but they have not proved as effective as the standard quasi-Newton methods. This state provides a motivation to study SR1 more deeply.

5.2 SR1 Formula

When the SR1 update is used to minimize the quadratic function (1.4.1), it has some attractive properties. Lemma 5.2.1 and Theorem 5.2.2 are well-known.

Lemma 5.2.1 Given an initial point $x_1$, let us move successively along $n$ directions $d_j$ ($1 \leq j \leq n$), and update $H_i$ after each step using the SR1 formula. If the SR1 update is well defined at each step, then the hereditary property holds for the quadratic function (1.4.1). That is,

$$H_i y_j = s_j, \quad j = 1, 2, ..., i - 1. \quad (5.2.1)$$

It should be pointed out that the hereditary property of the SR1 update does not require a quasi-Newton direction of search, and does not depend on step length $\alpha$. This is an important property on the SR1 update.

Theorem 5.2.2 Let the conditions of Lemma 5.2.1 hold. If $d_1, d_2, ..., d_n$ are linearly independent, then the SR1 algorithm terminates for the quadratic function (1.4.1) in at most $n + 1$ steps, with $H_{n+1} = A^{-1}$.

Remark 5.2.3 Theorem 5.2.2 shows that we have complete freedom in choosing the direction of search, as long as these directions are linearly independent. In fact, if the directions are linearly dependent, the SR1 update will not be defined at some step.
Moreover, for the non-quadratic objective function (1.1.1), the matrices generated by the SR1 update converges to the true Hessian matrix under commonly assumed conditions. But not always regarded as realistic. This result is due to Conn, Gould and Toint (1987):

**Theorem 5.2.4** Assume that $f(x)$ be twice continuously differentiable, and that $G(x)$ is bounded and $G(x)$ is Lipschitz continuous. Let $\{x_k\}$ be the sequence produced by using the SR1 update with the condition

$$\|y_k - H_k^{-1}s_k\| s_k \geq \beta \|y_k - H_k^{-1}s_k\| s_k,$$  \hspace{1cm} (5.2.2)

where $\beta \in (0,1)$. and let the sequence $\{s_k\}$ is uniformly linearly independent, that is there exist a $\bar{c} > 0$, a $K$ and an $m \geq n$ such that, for $k \geq K$, one can choose $n$ distinct indices: $k \leq k_1 < \cdots < k_n \leq k + m$ with $\mu_{\min}(S_k) \geq \bar{c}$, where $\mu_{\min}(S_k)$ is the minimum singular value of the matrix

$$S_k \overset{\Delta}{=} \begin{bmatrix} s_{k_1} \\ \|s_{k_1}\| \\ \vdots \\ s_{k_n} \\ \|s_{k_n}\| \\ \end{bmatrix}.$$  \hspace{1cm} (5.2.3)

If the sequence $\{x_k\}$ converges to $x^*$, then $\lim_{k \to \infty} H_k^{-1} = G(x^*)$.

### 5.3 SSR1 update formula

Introducing a scaling factor into the SR1 formula, we obtain the scaling symmetric rank one update (SSR1). This is,

$$H_{k+1} = \theta_k H_k + \frac{(s_k - \theta_k H_k y_k) (s_k - \theta_k H_k y_k)^T}{(s_k - \theta_k H_k y_k)^T y_k},$$  \hspace{1cm} (5.3.1)
Let $H_k$ be positive definite and choose $\theta_k$ such that $(s_k - \theta_k H_k y_k)^T y_k \neq 0$. Then the SSR1 update has some attractive properties:

**Theorem 5.3.1** Let the update (5.3.1) be written as $H_{k+1} = \theta_k H_k + E_k$. If the symmetric positive definite matrix $W_k$ satisfies $(W_k^{-1} + \theta_k H_k y_k) = s_k$, then $E_k$ is the solution of the problem:

\[
\begin{align*}
\text{minimize} & \quad \|W_k^{1/2} E_k W_k^{1/2}\|_F \\
\text{subject to} & \quad H_{k+1} y_k = s_k \\
& \quad E_k = E_k^T.
\end{align*}
\]

(5.3.2)

**Proof.** Greenstadt (1970) shows that the solution of (5.3.2) possesses the following form (here we drop $k$):

\[
E = \mu \{ y^T W^{-1} + W^{-1} y y^T - \theta H y y^T W^{-1} - \theta W^{-1} y y^T W^{-1} W^{-1} \}
\]

where $\mu = 1/y^T W^{-1} y$. Substituting $W_k^{-1} y_k = s_k - \theta_k H_k y_k$ into the above formula, we obtain

\[
E_k = \frac{(s_k - \theta_k H_k y_k)(s_k - \theta_k H_k y_k)^T}{(s_k - \theta_k H_k y_k)^T y_k}
\]

\[
E_k = \frac{(s_k - \theta_k H_k y_k)(s_k - \theta_k H_k y_k)^T}{(s_k - \theta_k H_k y_k)^T y_k}
\]

\[\square\]

**Theorem 5.3.2** The corresponding expression for the inverse of the updating formula (5.3.1) is

\[
B_{k+1} = \omega_k B_k + \frac{(y_k - \omega_k B_k s_k)(y_k - \omega_k B_k s_k)^T}{(y_k - \omega_k B_k s_k)^T s_k}
\]

(5.3.3)

where $\omega_k = \theta_k^{-1}$.  


Theorem 5.3.3 (Sun 1984) Let the update (5.3.1) be used to minimize the quadratic function (1.4.1) with exact line search. Then for $1 \leq k \leq n$,

$$H_{k+j} = \begin{cases} s_j, & j = k - 1 \\ \left( \prod_{p=1}^{j-1} \theta_p \right) s_j, & j < k - 1 \end{cases}$$

and

$$s_i^T A s_j = 0, \quad \text{for } i \neq j \text{ and } i, j = 1, \ldots, k. \quad (5.3.5)$$

Corollary 5.3.4 The iterations using the SSR1 update with exact line search possesses the quadratic termination property.

Theorem 5.3.5 In (5.3.1), if $\theta_k > 0$ and $s_k^T y_k > 0$, then $H_{k+1}$ is positive definite if and only if $\theta_k$ is outside the interval $[s_k^T y_k, s_k^T B_k s_k^T y_k]$.

Proof. For a symmetric rank one correction of a symmetric positive definite matrix, it is positive definite if and only if the determinant of the matrix which is obtained by correction is positive. So we only need to prove the following conclusion: under the assumptions of the theorem, $\det(H_{k+1}) > 0$ if and only if $\theta_k$ is outside the interval $[s_k^T y_k, s_k^T B_k s_k^T y_k]$.

It follows from (5.3.1) that

$$H_{k+1} = \theta_k H_k \left\{ I + \frac{(\theta_k H_k)^{-1} s_k - y_k)(s_k - \theta_k H_k y_k)^T}{(s_k - \theta_k H_k y_k)^T y_k} \right\}$$

$$\det(H_{k+1}) = \det(H_k) \theta_k^{n-1} \frac{s_k^T B_k s_k - \theta_k s_k^T y_k}{s_k^T y_k - \theta_k y_k^T H_k y_k}. \quad (5.3.6)$$
Corollary 5.3.6 If $\omega_k > 0$ and $s_k^T y_k > 0$, then $B_{k+1}$ is positive definite if and only if $\omega_k$ is outside the interval $[s_k B_k s_k, s_k^T y_k / s_k^T y_k]$.
Algorithm 5.3.7 (SSR1)

Step 0 Let \( k \) be specified. Given \( x_k \in \mathbb{R}^n \) and an symmetric positive definite matrix \( H_k \).

Step 1 If the convergence criterion is achieved, then stop.

Step 2 Calculate the search direction \( d_k \) using the equation \( d_k = -H_k g_k \).

Step 3 Select a step-length \( \alpha_k > 0 \) such that Rule (1.2.4) is satisfied.

Step 4 Set \( s_k = \alpha_k d_k \) and \( x_{k+1} = x_k + s_k \); calculate \( g_{k+1} \) and \( y_k \).

Step 5 Choose a scaling factor \( \theta_k \) so that it is outside the interval

\[
[T_k y_k \quad T_k H_k y_k \quad T_k B_k T_k y_k | T_k y_k].
\]

Step 6 Update \( H_k \) using (5.3.1). Set \( k = k+1 \) and go to Step 1.

\[ \square \]

Remark 5.3.8 the condition (1.2.4b) in Rule (1.2.4) implies that there exists \( \epsilon_0 > 0 \) which is independent of \( k \) so that

\[
s_k^T y_k > \epsilon_0 \| s_k \| \| y_k \|. \tag{5.3.7}
\]

5.4 Finite Termination Property on Algorithm SSR1

For analysing the properties of a quasi-Newton method on the quadratic function (1.4.1), sometimes it is convenient to introduce the following transformations which is due to Broyden (1970):

\[
\begin{align*}
T_k &= A^{1/2} H_k A^{1/2} \\
T_{k+1} &= A^{1/2} H_{k+1} A^{1/2},
\end{align*}
\tag{5.4.1}
\]

and

\[
\begin{align*}
e_k &= A^{-1/2} g_k \\
e_{k+1} &= A^{-1/2} g_{k+1}.
\end{align*}
\tag{5.4.2}
\]
For a quasi-Newton iteration on the quadratic function (1.4.1), it is easy to derive the following equations:

\[ e_{k+1} = e_k - \alpha_k T_k e_k, \quad (5.4.3) \]
\[ s_k^T B_k s_k = \alpha_k^2 e_k^T T_k e_k, \quad (5.4.4) \]
\[ s_k^T y_k = \alpha_k^2 e_k^T T_k^2 e_k, \quad (5.4.5) \]

and

\[ y_k^T H_k y_k = \alpha_k^2 e_k^T T_k^3 e_k. \quad (5.4.6) \]

Applying the transformation (5.4.1) and (5.4.2) to the SSR1 update, we have

\[ T_{k+1} = \theta_k^T T_k + \frac{[(I - \theta_k T_k)T_k e_k][(I - \theta_k T_k)T_k e_k]^T}{(T_k e_k)^T (I - \theta_k T_k) (T_k e_k)}. \quad (5.4.7) \]

Following Powell's (1972) idea, we outline the following definitions and results:

**Lemma 5.4.1** Let \( \Lambda_k \) and \( \Omega_k \) denote the space spanned by vectors \( (AH_k)^j g_k \) and \( T_k^j e_k \) (\( j = 1, 2, ... \)) respectively, then

\[ \dim(\Lambda_k) = \dim(\Omega_k). \quad (5.4.8) \]

**Definition 5.4.2** An iteration is called a "good" iteration if the equation

\[ \dim(\Lambda_{k+1}) = \dim(\Lambda_k) - 1 \quad (5.4.9) \]

is obtained and the step length \( \alpha_k \) satisfies Rule (1.2.4).
Theorem 5.4.3 Assume that a quasi-Newton algorithm is used to minimize the quadratic function (1.4.1), starting at point \( x = x_1 \) with the positive definite or positive semi-definite matrix \( H_1 \), and that on each iteration the steplenth \( \alpha_k \) satisfies Rule 1.2.4. If \( \hat{r} = \text{dim } (A_1) \), then after exactly \( \hat{r} \) "good" iterations the algorithm terminates.

Lemma 5.4.4 Let \( \text{dim}(\Omega_k) = \sigma_k \), then the vectors \( T^j_k e_k (j = 1, 2, \ldots, \sigma_k) \) are linearly independent.

After making the above preparations, we can discuss the quadratic termination property on Algorithm SSR1 now.

Theorem 5.4.5 Let the SSR1 update be used to minimize the quadratic function (1.4.1), then

\[
\text{dim}(\Omega_k) - 1 \leq \text{dim}(\Omega_{k+1}) \leq \text{dim}(\Omega_k). 
\] (5.4.10)

Proof. Let \( \text{dim}(\Omega_k) = \sigma_k \) and \( \text{dim}(\Omega_{k+1}) = \sigma_{k+1} \). Consider

\[
T_{k+1} = \theta_k T_k + Z_k, 
\]

where

\[
Z_k = w_k w_k^T / w_k^T T_k e_k 
\] (5.4.11)

and

\[
w_k = (I - \theta_k T_k) T_k e_k. 
\] (5.4.12)

For \( j \geq 1 \),

\[
T^j_{k+1} = \theta_k^j T_k + \theta_k^{j-1} (T_{k+1}^{j-1} T_k + T_{k+1}^{j-2} Z_k + \ldots + T_{k+1} Z_k T_k) + \ldots + \theta_k Z_k T_k^{j-1} + \theta_k^{j-2} Z_k T_k^{j-2} + \ldots + Z_k T_k^{j-2} + Z_k^j. 
\]

From (5.4.3) and (5.4.11), we have
\[ T_{k}^{j}e_{k+1} = T_{k}^{j}e_{k} - \alpha_{k}T_{k}^{j+1}e_{k}; \]

\[ (T_{k}^{j-1}Z_{k})e_{k+1} = \left( \frac{w_{k}^{T}e_{k}}{w_{k}^{T}T_{k}^{j}e_{k}} - \alpha_{k}\right)(T_{k}^{j}e_{k} - \theta_{k}T_{k}^{j+1}e_{k}); \]

for \( 1 \leq t \leq j - 2 \),

\[ (T_{k}^{1}Z_{k}T_{k}^{j-1})e_{k+1} = \left( \frac{w_{k}^{T}T_{k}^{j-1}e_{k}}{w_{k}^{T}T_{k}^{j}e_{k}} - \alpha_{k}w_{k}^{T}T_{k}^{j}e_{k}\right)(T_{k}^{1}e_{k} - \theta_{k}T_{k}^{2}e_{k}), \]

especially, when \( t = j - 2 \),

\[ (T_{k}^{j-2}Z_{k}T_{k})e_{k+1} = \left( \frac{w_{k}^{T}T_{k}^{j-1}e_{k}}{w_{k}^{T}T_{k}^{j}e_{k}} - \alpha_{k}w_{k}^{T}T_{k}^{j}e_{k}\right)(T_{k}^{j-1}e_{k} - \theta_{k}T_{k}^{j}e_{k}), \]

and when \( t = 1 \),

\[ (T_{k}Z_{k}^{j-2})e_{k+1} = \left( \frac{w_{k}^{T}T_{k}^{j-1}e_{k}}{w_{k}^{T}T_{k}^{j}e_{k}} - \alpha_{k}w_{k}^{T}T_{k}^{j-1}e_{k}\right)(T_{k}^{2}e_{k} - \theta_{k}T_{k}^{3}e_{k}). \]

Moreover for \( t = 1, 2, ..., \)

\[ Z_{k}^{1} = (w_{k}^{T}w_{k}/w_{k}^{T}T_{k}^{j}e_{k})^{j-1}Z_{k}, \quad \text{(5.4.13)} \]

so

\[ (T_{k}Z_{k}^{j-1})e_{k+1} = \left( \frac{(w_{k}^{T}w_{k})^{j-2}(w_{k}^{T}e_{k+1})}{(w_{k}^{T}T_{k}^{j}e_{k})^{j-1}} \right)(T_{k}^{2}e_{k} - \theta_{k}T_{k}^{3}e_{k}). \]

Similarly, we have that \( Z_{k}^{j}, (Z_{k}^{j-1}T_{k})e_{k+1}, (Z_{k}^{j-1}T_{k})e_{k+1}, \) and \( (Z_{k}^{j}T_{k})Z_{k}^{j-1})e_{k+1} \) where \( 1 \leq t \leq j-2 \) are linearly combination of \( T_{k}^{j}e_{k} \) and \( T_{k}^{2}e_{k} \). Hence, the vector \( T_{k}^{j+1}e_{k+1} \) is a linearly combination of the vectors \( T_{k}^{j}e_{k} \) (\( t = 1, 2, ..., j+1 \)). This shows that \( \Omega_{k+1} \) is a subspace of \( \Omega_{k} \). That is,
If $\sigma_k = 1$, Theorem 5.4.5 holds. When $\sigma_k \geq 2$, there exist $\beta_j$ ($j = 1, 2, \ldots, \sigma_{k+1}$) such that

$$\sigma_{k+1}^{+1} \sum_{j=1}^{\sigma_{k+1}^{+1}} \beta_j T^j e_{k+1} = 0.$$  \hfill (5.4.15)

Thus

$$\sigma_{k+1}^{+2} \sum_{j=1}^{\sigma_{k+1}^{+2}} \mu_j T^j e_k = 0,$$  \hfill (5.4.16)

where

$$\mu \sigma_{k+1}^{+2} = -\beta \sigma_{k+1}^{+1} \theta \sigma_{k+1}^{+1} (w^T e_k / w^T T e_k).$$

The term $w^T e_k / w^T T e_k$ is zero, only if $\theta_k$ is equal to $s^T k e_k / s^T k e_k$, which is not allowed by Theorem 5.3.5. Moreover, if $\beta \sigma_{k+1}^{+1}$ is zero, then equation (5.4.15) implies that the vectors $T^j e_{k+1}$, $j = 1, 2, \ldots, \sigma_{k+1}$ are linear dependent, which contradicts Lemma 5.4.4. Thus we deduce that $\mu \sigma_{k+1}^{+2}$ is non-zero.

It follows, from equation (5.4.16) and Lemma 5.4.4, that the value of $\sigma_{k+1}^{+2}$ is at least $\sigma_k + 1$. That is,

$$\dim(\Omega_{k+1}) \geq \dim(\Omega_k) - 1.$$  \hfill (5.4.17)

By the inequalities (5.4.14) and (5.4.17), Theorem 5.4.5 is proved.

$$\Box$$

By using Powell's result (1972) and Theorem 5.4.5 we obtain the following conclusion directly:
Theorem 5.4.6  Assume that the SSR1 update with an inexact line search satisfying Rule (1.2.4) is used to minimize the quadratic function (1.4.1), and that $s_k^T y_k > y_k^T H_k y_k$ if $\theta_k = 1$ is chosen from $k$th to $(k+1)$th iteration, Then

$$\dim(\Omega_{k+1}) = \dim(\Omega_k) - 1.$$  \hspace{1cm} (5.4.18)

Corollary 5.4.7  Under stated conditions Algorithm 5.3.7 (SSR1) possesses the finite termination property, namely it will terminate at the minimum $x^*$ of the positive definite quadratic function (1.4.1) after at most $n$ iterations in which $\theta = 1$ is chosen.

This result is clear, because each choice of $\theta_k = 1$ always makes the equation (5.4.18) hold. It is well known that the SR1 update possesses quadratic termination property without exact line search if it is well defined in every iteration. The key point is that $\dim(\Omega_k)$ is nonincreasing and decreases each time the SR1 update ($\theta = 1$) is used.
5.5 Convergence of the matrices generated by the SSR1 update

There are few results about the convergence of the sequence of Hessian estimates $H_k$. The oldest result is that, given a positive definite quadratic objective function, a sequence of $n$ linearly independent steps and exact line searches, then the quasi-Newton formula determines the exact Hessian matrix at the $(n+1)$th iteration (see Fiacco and McCormick 1968). Powell (1970) proved the convergence of the matrices of an algorithm that employs the PSB update. Schuller (1974) considered the convergence of the matrices generated by the Broyden family under strong assumptions. Ge and Powell (1983) point out that, if $f(x)$ is twice continuously differentiable, $G(x)$ is Lipschitz continuous and $G(x^*)$ is positive, and if $x_1$ and $B_1$ are close to $x^*$ and $G(x^*)$, then the sequence of $\{B_k; k = 1, 2, \ldots\}$ generated by both the DFP and the BFGS update is convergent. Conn, Gould and Toint (1987) prove a result closely related to that given here. They prove the convergence of the matrices generated by the SR1 update under commonly assumed conditions, i.e., Theorem 5.2.4.

Here we consider the convergence of the matrices $B_k$, i.e. $H_k^{-1}$ generated by SSR1 update under the following assumptions which essentially follow Conn, Gould and Toint.

1. (AS.1) $f(x)$ is twice continuously differentiable.
2. (AS.2) $V^2f(x)$ is bounded and Lipschitz continuous. That is there exist constants $c_1 > 0$ and $c_2 > 0$ such that, for all $x$ and $\bar{x} \in \mathbb{R}^n$, 
\[
\|\nabla^2 f(x)\| \leq c_1 \tag{5.5.1}
\]

and
\[
\|\nabla^2 f(x) - \nabla^2 f(\bar{x})\| \leq c_2\|x - \bar{x}\|. \tag{5.5.2}
\]

(AS.3) The sequence of iterates \(\{x_k\}\) generated by Algorithm 5.3.7 converges to \(x^*\).

(AS.4) In every iteration,
\[
|u_k^T s_k| \geq c_3\|u_k\|\|s_k\| \tag{5.5.3}
\]

where \(u_k = y_k - \omega_k B_k s_k\) and \(c_3 \in (0,1)\).

**Lemma 5.5.1** Assume that (AS.1) and (AS.2) hold, and also that \(\{x_k\}\) is a sequence of iterates generated by the Algorithm 5.3.7. Then, for all \(j\) and \(i \geq j+2\),
\[
|u_i^T s_j| \leq c_1 \xi_{ij} \|s_i\|\|s_j\| + c_2 \eta_{ij} \|s_i\|\|s_j\| + \rho_{ij} \|s_i\|\|y_j - B_i s_j\|,
\]

where
\[
\rho_{ij} \triangleq \max\{\omega_p |j+1 \leq p \leq i\},
\]
\[
\xi_{ij} \triangleq \max\{1 - \omega_p |j+1 \leq p \leq i\},
\]
\[
\eta_{ij} \triangleq \max\{\|x_s - x_t\| | j \leq t \leq s \leq i\}.
\]

**Proof.** Using the mean value theorem, we have that, for all \(j\),
\[
y_j = G_j s_j
\]

where
\[
G_j = \int_0^1 \nabla^2 f(x_{j-1} + \tau s_j) d\tau. \tag{5.5.4}
\]
From (AS.1) and (AS.2) we obtain that

\[
|u^T_{j} s_j| \leq |(1 - \omega_j) s^T_{j} y_j| + |y^T_{j} s_j - s^T_{j} y_j| + |\omega_j s^T_{j} (y_j - B_j s_j)|
\]

\[
\leq |(1 - \omega_j) s^T_{j} G_j s_j| + |s^T_{j} (G_j - G_j) s_j| + |\omega_j s^T_{j} (y_j - B_j s_j)|
\]

\[
\leq c_1 \xi_{ij} ||s_i|| ||s_j|| + c_2 \eta_{ij} ||s_i|| ||s_j|| + \rho_{ij} ||s_i|| ||y_j - B_j s_j||.
\]

\[\square\]

Lemma 5.5.2 If the assumption of Lemma 5.5.1 and (AS.4) hold, then for all j and i ≥ j + 2

\[
\begin{cases}
\|y_j - B_{j+1} s_j\| = 0 \\
\|y_j - B_i s_j\| \leq [c_1 (1+1/c_3) \xi_{i-1,j} + (c_2/c_3) \eta_{i-1,j}][1+(1+1/c_3)\rho_{i-1,j}]^{i-j-2} ||s_j||.
\end{cases}
\]

Proof. The first equality is just a statement of the quasi-Newton condition. The proof of the second inequality is by induction. Let it hold for i = k, so that

\[
\|y_j - B_k s_j\| \leq [c_1 (1+1/c_3) \xi_{k-1,j} + (c_2/c_3) \eta_{k-1,j}][1+(1+1/c_3)\rho_{k-1,j}]^{k-1} ||s_j||. 
\] (5.5.5)

We now consider the case i = k+1. By means of (AS.1), (AS.4) and Lemma 5.5.1, we obtain

\[
\|y_j - B_{k+1} s_j\| = \|y_j - (\omega_k B_k + (u_k u^T_k s_k) s_j)\| 
\]

\[
\leq |1 - \omega_k| ||y_j|| + \omega_k ||y_j - B_k s_j|| + |u^T_k s_j|/(c_3 ||s_k||) 
\]

\[
\leq c_1 (1+1/c_3) \xi_{k+1,j} ||s_j|| + (c_2/c_3) \eta_{k+1,j} ||s_j|| + (1+c_3) \rho_{k+1,j} ||y_j - B_k s_j||. 
\]

Thus we replace \(\|y_j - B_k s_j\|\) by (5.5.5) and obtain the required result.

\[\square\]

In order to prove \(B_k\) converges to \(\nabla^2 f(x^*)\) we make the further assumptions:
The sequence \( \{ s_k \} \) is uniformly linearly independent. That is there exist \( c_4 > 0 \), \( \hat{k}_1 \), and \( m \geq n \) such that, for each \( \kappa \geq \hat{k}_1 \), we can choose \( n \) distinct indices \( \kappa_1 \leq \ldots \leq \kappa_n \leq \kappa + m \) with \( \sigma_{\min}(S_\kappa) \geq c_4 \), where \( \sigma_{\min}(S_\kappa) \) is the minimum singular value of the matrix

\[
S_\kappa \triangleq \begin{bmatrix} s_{\kappa_1} & \cdots & s_{\kappa_n} \\ \| s_{\kappa_1} \| & \cdots & \| s_{\kappa_n} \| \\ \end{bmatrix}.
\]  

The limit of the sequence \( \{ \omega_\kappa \} \) is one. That is for any \( \delta > 0 \), there exists \( \hat{k}_2 \) such that, when \( \kappa \geq \hat{k}_2 \) we have

\[
|\omega_\kappa - 1| < \delta.
\]  

\textit{Theorem 5.5.3} Let \( \{ x_k \} \) be a sequence of iterates generated by the Algorithm (5.3.7), and assume that (AS.1) --- (AS.6) hold, then

\[
\lim_{k \to \infty} \| B_k - \nabla^2 f(x^*) \| = 0.
\]  

\textit{Proof.} Consider

\[
\| (B_{k+m+1} - \nabla^2 f(x^*)) s_k \| \leq \| y_k - B_{k+m+1} s_k \| + \| y_k - \nabla^2 f(x^*) s_k \|. 
\]  

Here

\[
\| y_k - \nabla^2 f(x^*) s_k \| = \| (G_k - \nabla^2 f(x^*)) s_k \| \leq c_2 \varepsilon_k \| s_k \|, 
\]  

where

\[
\varepsilon_k \triangleq \max \{ \| x_t - x^* \| : \kappa \leq t \leq \kappa + m+1 \}
\]  

and

\[
\| y_k - B_{k+m+1} s_k \| \leq \left[ c_1 \left( 1 + 1/c_3 \right) \varepsilon_{(k+m)_k} + c_2 \varepsilon_{(k+m)} \right] \eta_{(k+m)_k} \\
\cdot \left[ 1 + (1/c_3) \rho_{(k+m)_k} \right]^{m-1} \| s_k \|. 
\]
We simplify this second inequality by using that

\[ \|x_s - x_t\| \leq \|x - x^*\| + \|x_t - x^*\| \]

implies

\[ \eta_{(k+m)k} \leq 2\epsilon_k. \]

Set \( \hat{k}_0 = \max\{\hat{k}_1, \hat{k}_2\} \) and \( \delta = \epsilon_k \). For \( \kappa \geq \hat{k}_0 \), (5.5.11) can be written as

\[ \|y_{k+m} - B_{k+m+1}s_k\| \leq [c_1(1+1/c_3) + 2c_2/c_3][1+(1+\epsilon_k)(1 + \epsilon_k)]^{-m}\epsilon_k\|s_k\|. \]

Hence for any \( \kappa \leq j \leq \kappa+m \),

\[ \|B_{k+m+1}^{-\nabla^2 f(x^*)}s_j\| \leq \tau_k\epsilon_k, \]

where

\[ \tau_k = c_2 + [c_1(1+1/c_3) + 2c_2/c_3][1+(1+\epsilon_k)(1 + \epsilon_k)]^{-m}. \]

On the other hand, from (AS.5) we have

\[ \|B_{k+m+1}^{-\nabla^2 f(x^*)}\| \leq 1/c_4\|B_{k+m+1}^{-\nabla^2 f(x^*)}s_k\| \leq (\sqrt{n}/c_4)\tau_k\epsilon_k. \]

Then, since

\[ \lim_{k \to \infty} \tau_k = c_5 \]

where

\[ c_5 = c_2 + [c_1(1+1/c_3) + 2c_2/c_3][2+1/c_3]^{-m}, \]

it follows that

\[ \lim_{k \to \infty} \|B_k^{-\nabla^2 f(x^*)}\| = 0. \]
By Theorem 5.5.3 and Theorem 1.3.2, we obtain the following result directly:

**Corollary 5.5.4** If the assumptions of Theorem 5.5.3 hold, and if \( G^* \) is positive definite, then for \( x_k \) sufficiently close to \( x^* \), the choice \( \alpha = 1 \) satisfies (1.2.4), and if this choice is made at each iteration when it is acceptable, then \( \{x_k\} \) generated by algorithm SSR1 converges q-superlinearly to \( x^* \).

**Remark 5.5.5** The condition that \( \{\omega_k\} \rightarrow 1 \) is somewhat unsatisfactory from a theoretical point of view. However, it can be tested in practice, and numerical evidence is presented in figure 5.2.

### 5.6. The optimal choice of \( \theta \)

How to choose \( \theta \) to maximize the effectiveness of SSR1 is an important question. Spedicato (1985) also discusses this problem for the two scaling parameters in his updating formula, but he does not give a definitive recommendation. Here we choose \( \theta \) following Davidon’s idea (see Davidon (1975)) of minimizing the spectral condition of the diagonal plus rank one matrix

\[
C_k = H_k^{-1/2} H_{k+1}^{-1/2}
\]

\[
= \theta_k I + \frac{(H_k^{-1/2} s_k - \theta_k H_k^{1/2} y_k)(H_k^{-1/2} s_k - \theta_k H_k^{1/2} y_k)^T}{(H_k^{-1/2} s_k - \theta_k H_k^{1/2} y_k)^T(H_k^{-1/2} y_k)}.
\]

and this approach proves worthwhile. Let

\[
\psi(\theta_k) \Delta = -\frac{(s_k^T B_k s_k) - (s_k^T y_k) \theta_k}{(s_k^T y_k)^2 \theta_k}.
\]
Then, when \( 0 < \theta_k < s_k^T y_k / y_k^T H_k y_k \) or \( \theta_k > s_k^T B_k s_k / y_k^T H_k y_k \), we have

\[
\text{cond}(C_k) = \begin{cases} 
\psi(\theta_k), & \theta_k < \frac{s_k^T B_k s_k / y_k^T H_k y_k}{s_k^T y_k / y_k^T H_k y_k} \\
1/\psi(\theta_k), & \theta_k > \frac{s_k^T B_k s_k / y_k^T H_k y_k}{s_k^T y_k / y_k^T H_k y_k}
\end{cases}
\] (5.6.3a)

(5.6.3b)

A typical plot of \( \psi(\theta_k) \) is given in Figure 5.1.

The salient features are readily verified. For example, the turning points \( \theta_1 \) and \( \theta_2 \) are given by

\[
\theta_k^{(1)} = \frac{s_k^T B_k s_k}{s_k^T y_k} - \frac{(s_k^T B_k s_k)^2}{(s_k^T y_k)^2} - \frac{s_k^T B_k s_k}{y_k^T H_k y_k}
\] (5.6.4)

and

\[
\theta_k^{(2)} = \frac{s_k^T B_k s_k}{s_k^T y_k} + \frac{(s_k^T B_k s_k)^2}{(s_k^T y_k)^2} - \frac{s_k^T B_k s_k}{y_k^T H_k y_k}
\] (5.6.5)

They are the minimum points of (5.6.3a) and (5.6.3b) respectively, which satisfy the required conditions. In addition,

\[
\psi(\theta_k^{(1)}) = \theta_k^{(2)} / \theta_k^{(1)} = 1/\psi(\theta_k^{(2)})
\] (5.6.6)

holds generally. This shows that both \( \theta_k^{(1)} \) and \( \theta_k^{(2)} \) are optimal scaling factors in the sense of Davidon, and raises the question of which value to choose in practice? Two possible strategies are suggested:

**Strategy 5.6.1** Always take \( \theta_k = \theta_k^{(1)} \). In this case \( 0 < \theta_k < b/a \), so \((s_k^T - \theta_k H_k y_k)^T y_k > 0\). This strategy provides the greater control over the size of the elements of \( H_{k+1} \).
Strategy 5.6.2 Choose the value of $\theta_k$ which makes $\text{cond}(H_{k+1})$ smaller.

This does not prove an easy condition to implement directly so it has been necessary to employ heuristics. Conditions both on the determinant and the trace have been investigated. The latter has proved superior and is derived here. From (5.6.6) and

$$\det(H_{k+1}) = \theta_k^{n-1} \psi(\theta_k) \det(H_k), \quad (5.6.7)$$

we have

$$\frac{\det(H_{k+1}^{\theta_1})}{\det(H_{k+1}^{\theta_2})} = (\theta_k^{1}/\theta_k^{2})^{n-2} < 1, \quad (5.6.8)$$

which implies

$$\prod_{i=1}^{n} \lambda_i^{\theta_1} < \prod_{i=1}^{n} \lambda_i^{\theta_2}, \quad (5.6.9)$$

where $H_{k+1}^{\theta}$ denotes (5.3.1) at $\theta_k = \theta_k^{(j)}$ (j = 1, 2) and $\lambda_i^{\theta}$ are its eigenvalues with $\lambda_1^{\theta} \leq \lambda_2^{\theta} \leq \ldots \leq \lambda_n^{\theta}.$ In addition,

$$\text{Tr}(H_{k+1}^{\theta}) = \sum_{i=1}^{n} \lambda_i^{\theta}, \quad (5.6.10)$$

where

$$\text{Tr}(H_{k+1}^{\theta}) = \theta_k \text{Tr}(H_k) + \frac{\text{Tr}(T_k v_k)}{\text{Tr}(v_k^T v_k)} \quad (5.6.11)$$

with $v_k = s_k - \theta_k H_k v_k.$ On the other hand,

$$\text{Tr}(B_{k+1}^{\theta}) = \sum_{i=1}^{n} 1/\lambda_i^{\theta}, \quad (5.6.12)$$

and

$$\text{Tr}(B_{k+1}^{\theta}) = \frac{1}{\theta_k} \text{Tr}(B_k) + \frac{\text{Tr}(u_k^T u_k)}{\text{Tr}(u_k^T s_k)} \quad (5.6.13)$$
Thus, $\text{cond}(H_{k+1}^{\theta_2})$ probably is less than $\text{cond}(H_{k+1}^{\theta_1})$, when

$$\text{Tr}(H_{k+1}^{\theta_1}) \geq \text{Tr}(H_{k+1}^{\theta_2})$$  \hspace{1cm} (5.6.14)

and

$$\text{Tr}(B_{k+1}^{\theta_1}) \geq \text{Tr}(B_{k+1}^{\theta_2}).$$  \hspace{1cm} (5.6.15)

In fact, it is not difficult to calculate the criteria (5.6.14) and (5.6.15) by using the recurrence formulae (5.6.11) and (5.6.13) with the initial matrix $H_1 = I$.

**Remark 5.6.3** Inspection of Figure 5.1 shows the possibility of degenerate cases corresponding to $b/a$ very close to $c/b$. This case, namely $ac = b^2$, occurs if $H_k y_k = s_k$. Since it causes (5.6.2) to fail, it is necessary to consider how to deal with this trouble. In this case, (5.3.1) and (5.6.1) are changed into the following forms:

$$H_{k+1} = \theta_k H_k + \frac{(1 - \theta_k)s_k^T s_k}{s_k^T y_k}$$  \hspace{1cm} (5.6.16)

and

$$C_k = \theta_k Z(\theta_k),$$  \hspace{1cm} (5.6.17)

where

$$Z(\theta_k) = I + \frac{(1 - \theta_k)(H_k^{1/2} y_k)(H_k^{1/2} y_k)^T}{(y_k^T H_k y_k)\theta_k}.$$  \hspace{1cm} (5.6.18)

Since

$$\text{cond}(C_k) = \begin{cases} 1/\theta_k, & 0 < \theta_k < 1 \\ \theta_k, & \theta_k > 1, \end{cases}$$  \hspace{1cm} (5.6.19)

taking $\theta_k = 1$ is a good strategy. That is, set
\[ H_{k+1} = H_k. \] (5.6.20)

**Remark 5.6.4** Both the formula (5.6.4) and (5.6.5) are consistent in the sense that if the iteration is started from \( x^* + \delta \) with the correct Hessian then \( \omega = 1 + o(\delta) \) and the correction to the Hessian is \( O(\delta) \).

We can establish the optimally conditioned algorithm based on (5.3.1).

**Algorithm 5.6.5 (OCSSR1)**

**Step 0** Let \( k \) be specified. Given \( \varepsilon_1 > 0, \varepsilon_2 > 0, x_k \in \mathbb{R}^n \) and a symmetric positive definite matrix \( H_k \).

**Step 1** If the convergence criterion is achieved, then stop.

**Step 2** Calculate the search direction \( d_k \) using the equation

\[
d_k = -H_k g_k
\]

where \( H_k \) satisfies the updating (5.3.1).

**Step 3** Select \( \alpha_k > 0 \) such that Rule (1.2.4) is satisfied.

**Step 4** Set \( s_k = \alpha_k d_k \) and \( x_{k+1} = x_k + s_k \); calculate \( g_{k+1} \) and \( y_k \).

**Step 5** If

\[
(s_k - H_k y_k)^T y_k > \varepsilon_1 \| s_k - H_k y_k \| \| y_k \|
\]

set \( \theta_k = 1 \) and go to Step 8. Else continue to Step 6.

**Step 6** If

\[
\| H_k y_k - s_k \| \leq \varepsilon_2,
\]

set \( H_{k+1} = H_k \) and go to Step 9. Else calculate \( \theta_k^{(1)} \) and \( \theta_k^{(2)} \) by using (5.6.4) and (2.5.5) respectively.

**Step 7** Determine \( \theta_k \) using one of the Strategies 5.6.1 or 5.6.2.

**Step 8** Update \( H_k \) by using (5.3.1).

**Step 9** Set \( k = k + 1 \) and go to Step 1.
Theorem 5.6.6  Algorithm 5.6.5 (OCSSR1) possesses the finite termination property, namely it will terminate at the minimum $x^*$ of the positive definite quadratic function (1.4.1) after at most $n$ iterations in which $\theta = 1$ is chosen.

Proof. OCSSR1 is really a special case of The SSR1 update. To show quadratic termination it is thus necessary show that the strategy for selecting $\theta$ ensure that $\theta \in \{s_k^T y_k / y_k^T H_k y_k, s_k^T B_k s_k / s_k^T y_k\}$ and this is verified readily by direct calculation. □

Remark 5.6.7 Forcing $\theta_k = 1$ is a choice which is favourable for Theorem 5.5.3 and implies that the SR1 update is used under the condition (5.6.21) which is one of the assumptions set by Conn, Gould and Toint (1987) to prove the convergence of the matrices generated by SR1. The choice is stipulated in the finite termination property Theorem 5.4.7. Numerical results conform its superiority.

5.7 Numerical results

An attempt has been made to give a fairly thorough testing of OCSSR1 by comparing its performance against what appears to be the most popular current methods on a range of frequently used test problems. Tests have been conducted using the following algorithms: BFGS, SBFGS, OCSSR1 with strategy 5.6.2 and OCSSR1N (a variance of OCSSR1 described below). Here $\varepsilon_1 = 10^{-8}$ and $\varepsilon_2 = 10^{-2}$ are used in the numerical performance of the OCSSR1 algorithm.

Table 5.1 gives numerical comparisons of the algorithms BFGS, SBFGS, OCSSR1. To study more completely the robustness of the methods and their implementations, we use not only the standard initial point $x_0$ but also the more difficult initial point $10x_0$. Here TF.A6c and TF.A6s etc. denote using $x_0$ and $10x_0$ respectively. In addition, (*) means $N_t > 500$. The termination criterion is that
Table 5.2 compares the results of calculation for OCSSR1 with the results given by Shanno and Phua (1978) for their testing of BFGS18, OCON, OCON10 and OSS2. Here the test functions, the initial points and the convergence criterion are the same as those used by them. That is,

Table 5.3 gives the numerical results obtained using Algorithm OCSSR1N, and the corresponding \( w_k \) values at the termination points of each iteration. Here the standard initial points \( x_0 \) are used, and the convergence criterion is (5.7.1). The table does not record results for TF.A9 because OCSSR1N failed to reduce \( \|g_k\| \) sufficiently. Our numerical experience indicates that the forcing strategy is always superior.
Figure 5.2 gives a comparison of $\omega_k$ values for OCSSR1 and OCSSR1N for the particular case TF.A14 with $n = 4$ which is fairly typical. However, on occasion, the forcing strategy tends to cause oscillations in the value of $\omega$ while the natural choice tends to 1 more smoothly. Despite this, as noted above, the forcing strategy leads to more rapid convergence in all our tests. In the case of TF.A6 which has a singular Hessian at the optimum, the forced choice was taken in the great majority of iterations and the difference in convergence rate between the two algorithms was very marked.

In summary, OCSSR1 compares favourably with BFGS. In addition, it seems that OCSSR1 is very effective in the so-called ill-conditioned and singular optimization problems with advantages that are more obvious when $n$ is large.
Table 5.1
(the comparison with BFGS and SBFGS)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>n</th>
<th>N_t</th>
<th>BFGS N_f</th>
<th>N_g</th>
<th>SBFGS N_f</th>
<th>N_g</th>
<th>OCSSR1 N_t</th>
<th>N_f</th>
<th>N_g</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF.A1</td>
<td>2</td>
<td>21</td>
<td>24</td>
<td>22</td>
<td>17</td>
<td>19</td>
<td>18</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>TF.A2</td>
<td>4</td>
<td>32</td>
<td>35</td>
<td>33</td>
<td>42</td>
<td>44</td>
<td>43</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>TF.A4</td>
<td>2</td>
<td>12</td>
<td>46</td>
<td>13</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>TF.A5</td>
<td>10</td>
<td>25</td>
<td>37</td>
<td>26</td>
<td>32</td>
<td>34</td>
<td>33</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>TF.A6c</td>
<td>4</td>
<td>53</td>
<td>64</td>
<td>54</td>
<td>80</td>
<td>85</td>
<td>81</td>
<td>41</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>160</td>
<td>184</td>
<td>161</td>
<td>81</td>
<td>86</td>
<td>82</td>
<td>48</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>(*)</td>
<td></td>
<td></td>
<td>83</td>
<td>88</td>
<td>84</td>
<td>52</td>
<td>59</td>
</tr>
<tr>
<td>TF.A6s</td>
<td>4</td>
<td>66</td>
<td>89</td>
<td>67</td>
<td>89</td>
<td>103</td>
<td>90</td>
<td>74</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>329</td>
<td>401</td>
<td>330</td>
<td>103</td>
<td>118</td>
<td>104</td>
<td>75</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>(*)</td>
<td></td>
<td></td>
<td>106</td>
<td>120</td>
<td>107</td>
<td>77</td>
<td>92</td>
</tr>
<tr>
<td>TF.A7c</td>
<td>2</td>
<td>21</td>
<td>33</td>
<td>22</td>
<td>22</td>
<td>27</td>
<td>23</td>
<td>26</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>57</td>
<td>89</td>
<td>58</td>
<td>33</td>
<td>42</td>
<td>34</td>
<td>26</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>109</td>
<td>219</td>
<td>110</td>
<td>41</td>
<td>53</td>
<td>42</td>
<td>26</td>
<td>33</td>
</tr>
<tr>
<td>TF.A7s</td>
<td>2</td>
<td>65</td>
<td>110</td>
<td>66</td>
<td>55</td>
<td>76</td>
<td>56</td>
<td>61</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>133</td>
<td>185</td>
<td>134</td>
<td>59</td>
<td>80</td>
<td>60</td>
<td>61</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>(*)</td>
<td></td>
<td></td>
<td>58</td>
<td>79</td>
<td>59</td>
<td>62</td>
<td>79</td>
</tr>
<tr>
<td>TF.A9</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>24</td>
<td>25</td>
<td>25</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>26</td>
<td>27</td>
<td>27</td>
<td>36</td>
<td>37</td>
<td>37</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>TF.A11</td>
<td>4</td>
<td>83</td>
<td>87</td>
<td>84</td>
<td>75</td>
<td>80</td>
<td>76</td>
<td>62</td>
<td>66</td>
</tr>
<tr>
<td>TF.A13c</td>
<td>5</td>
<td>23</td>
<td>25</td>
<td>24</td>
<td>24</td>
<td>27</td>
<td>25</td>
<td>28</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>29</td>
<td>29</td>
<td>30</td>
<td>32</td>
<td>33</td>
<td>33</td>
<td>39</td>
<td>46</td>
</tr>
<tr>
<td>TF.A13s</td>
<td>5</td>
<td>28</td>
<td>36</td>
<td>29</td>
<td>44</td>
<td>55</td>
<td>45</td>
<td>42</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>50</td>
<td>59</td>
<td>51</td>
<td>86</td>
<td>94</td>
<td>87</td>
<td>46</td>
<td>60</td>
</tr>
<tr>
<td>TF.A14c</td>
<td>4</td>
<td>48</td>
<td>78</td>
<td>49</td>
<td>38</td>
<td>47</td>
<td>39</td>
<td>35</td>
<td>47</td>
</tr>
<tr>
<td>TF.A14s</td>
<td>4</td>
<td>65</td>
<td>99</td>
<td>66</td>
<td>90</td>
<td>108</td>
<td>91</td>
<td>56</td>
<td>75</td>
</tr>
</tbody>
</table>
Table 5.2
(the comparison with Shanno's results)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>n</th>
<th>BFGS18</th>
<th>OCON</th>
<th>OCON10</th>
<th>OSS2</th>
<th>OCSSR1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>N_t</td>
<td>N_f</td>
<td>H_t</td>
<td>N_t</td>
<td>N_f</td>
</tr>
<tr>
<td>TF.A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_o(a)</td>
<td>4</td>
<td>59</td>
<td>61</td>
<td>32</td>
<td>36</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
<td>59</td>
<td>61</td>
<td>51</td>
<td>63</td>
</tr>
<tr>
<td>x_o(b)</td>
<td>2</td>
<td>38</td>
<td>47</td>
<td>15</td>
<td>17</td>
<td>38</td>
</tr>
<tr>
<td>x_o(c)</td>
<td></td>
<td>50</td>
<td>62</td>
<td>55</td>
<td>63</td>
<td>61</td>
</tr>
<tr>
<td>x_o(d)</td>
<td></td>
<td>29</td>
<td>34</td>
<td>53</td>
<td>64</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>35</td>
<td>23</td>
<td>26</td>
<td>32</td>
</tr>
<tr>
<td>x_o(e)</td>
<td></td>
<td>38</td>
<td>47</td>
<td>29</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>x_o(f)</td>
<td></td>
<td>50</td>
<td>62</td>
<td>155</td>
<td>168</td>
<td>59</td>
</tr>
<tr>
<td>x_o(g)</td>
<td></td>
<td>29</td>
<td>34</td>
<td>158</td>
<td>176</td>
<td>29</td>
</tr>
<tr>
<td>x_o(h)</td>
<td></td>
<td>29</td>
<td>35</td>
<td>56</td>
<td>67</td>
<td>32</td>
</tr>
<tr>
<td>TF.A7</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_o(a)</td>
<td></td>
<td>38</td>
<td>47</td>
<td>29</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>x_o(b)</td>
<td></td>
<td>50</td>
<td>62</td>
<td>155</td>
<td>168</td>
<td>59</td>
</tr>
<tr>
<td>x_o(c)</td>
<td></td>
<td>29</td>
<td>34</td>
<td>158</td>
<td>176</td>
<td>29</td>
</tr>
<tr>
<td>x_o(d)</td>
<td></td>
<td>29</td>
<td>35</td>
<td>56</td>
<td>67</td>
<td>32</td>
</tr>
<tr>
<td>TF.A10</td>
<td>10</td>
<td>5</td>
<td>8</td>
<td>15</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>x_o</td>
<td></td>
<td>20</td>
<td>10</td>
<td>21</td>
<td>34</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>8</td>
<td>10</td>
<td>23</td>
<td>37</td>
</tr>
<tr>
<td>TF.A12</td>
<td>20</td>
<td>232</td>
<td>234</td>
<td>69</td>
<td>80</td>
<td>212</td>
</tr>
<tr>
<td>x_o</td>
<td></td>
<td>20</td>
<td>232</td>
<td>234</td>
<td>69</td>
<td>80</td>
</tr>
<tr>
<td>TF.A14</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x_o(a)</td>
<td></td>
<td>95</td>
<td>112</td>
<td>81</td>
<td>96</td>
<td>101</td>
</tr>
<tr>
<td>x_o(b)</td>
<td></td>
<td>85</td>
<td>111</td>
<td>71</td>
<td>86</td>
<td>97</td>
</tr>
<tr>
<td>x_o(c)</td>
<td></td>
<td>44</td>
<td>51</td>
<td>37</td>
<td>47</td>
<td>47</td>
</tr>
</tbody>
</table>
Table 5.3

(numerical results and ω value at termination point for OCSSR1N)

\(x_0\) is standard initial point and \(\|g\| < 10^{-8}\)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>n</th>
<th>(N_t)</th>
<th>(N_f)</th>
<th>(N_g)</th>
<th>(ω)</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF.A1</td>
<td>2</td>
<td>18</td>
<td>21</td>
<td>19</td>
<td>1.01256117192175</td>
<td>0.81x10^{-23}</td>
</tr>
<tr>
<td>TF.A2</td>
<td>4</td>
<td>68</td>
<td>71</td>
<td>69</td>
<td>0.93640357332493</td>
<td>0.52x10^{-19}</td>
</tr>
<tr>
<td>TF.A4</td>
<td>2</td>
<td>17</td>
<td>23</td>
<td>19</td>
<td>0.99999889956843</td>
<td>0.25x10^{-28}</td>
</tr>
<tr>
<td>TF.A5</td>
<td>10</td>
<td>50</td>
<td>63</td>
<td>52</td>
<td>1.14330727122093</td>
<td>0.88x10^{-20}</td>
</tr>
<tr>
<td>TF.A6</td>
<td>4</td>
<td>87</td>
<td>95</td>
<td>89</td>
<td>1.16799409213793</td>
<td>0.12x10^{-18}</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>87</td>
<td>95</td>
<td>89</td>
<td>1.17175712222292</td>
<td>0.41x10^{-18}</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>87</td>
<td>95</td>
<td>89</td>
<td>1.16093473026684</td>
<td>0.35x10^{-18}</td>
</tr>
<tr>
<td>TF.A7</td>
<td>2</td>
<td>22</td>
<td>30</td>
<td>24</td>
<td>1.00349837718278</td>
<td>0.88x10^{-25}</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>28</td>
<td>39</td>
<td>32</td>
<td>0.98058137761503</td>
<td>0.46x10^{-20}</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>30</td>
<td>41</td>
<td>33</td>
<td>1.00445769989559</td>
<td>0.37x10^{-23}</td>
</tr>
<tr>
<td>TF.A11</td>
<td>4</td>
<td>40</td>
<td>47</td>
<td>41</td>
<td>0.84515410933047</td>
<td>0.14x10^{-11}</td>
</tr>
<tr>
<td>TF.A13</td>
<td>5</td>
<td>32</td>
<td>34</td>
<td>33</td>
<td>1.07510743472465</td>
<td>0.84x10^{-19}</td>
</tr>
<tr>
<td>TF.A14</td>
<td>4</td>
<td>41</td>
<td>59</td>
<td>42</td>
<td>0.95013144181145</td>
<td>0.57x10^{-20}</td>
</tr>
</tbody>
</table>
Figure 5.1. $\phi(\theta)$ for representative $a$, $b$, $c$.

\[ \phi(\theta) = \frac{c-b\theta}{b\theta-a\theta^2} \]

- $a = 0.15707582$
- $b = 0.26602886$
- $c = 0.61161656$
Figure 5.2, OCSSR1 versus OCSSR1N

![Graph showing OCSSR1 and OCSSR1N over Iteration Number](image-url)
CHAPTER SIX

AN APPLICATION OF THE OCSSR1 UPDATE FOR TRUST REGION METHODS

The motivation for the idea of this chapter is to find a means whereby the potential of Algorithm OCSSR1 is exploited. The scheme suggested is one in which the descent step is sought by using trust region steps within restricted subspaces. Because each subspace can be specified to include information about previous steps, the method is also related to a supermemory descent method but avoids the need for performing multiple dimensional searches. Information of this kind may be useful in providing local geometry information, for example, when the Hessian at the current point is indefinite so that directions of negative curvature could be good descent directions while quasi-Newton methods which force $B_k$ to be definite are ineffective. Trust region methods have attractive global convergence properties, while supermemory information has good scale independence properties. Since the method possesses the characteristics of both the trust region method and the supermemory descent method, one can anticipate good numerical performance. Numerical tests illustrate this point.

6.1 Trust Region Methods on a Subspace

In Algorithm 2.7.1, if the trust region steps are restricted within a sequence of subspaces, the kth step is generated by solving the problem

$$
\text{Minimize} \quad \varphi_k(s) = f_k + g_k^Ts + \frac{1}{2}s^TB_k s \\
\text{Subject to} \quad s \in S_k \\
\|s\| \leq \Delta_k.
$$

(6.1.1)

Assume that $Z_k$ is a $n \times m$ matrix such that $Z_k^T Z_k = I$ and such that the column space of $Z_k$ spans $S_k$. Then the subspace constraint can be satisfied by setting $s_k = Z_k s_x$. 
Substituting this in (6.1.1) gives the problem

\[
\text{Minimize} \quad \psi_k(s_z) = f_k + g_z^T z + \frac{1}{2} s_z^T B_z s_z
\]

(6.1.2)

Subject to \( \|s_z\| \leq \Delta_k \).

where \( g_z = Z_k^T g_k, B_z = Z_k^T B_k Z_k \) and \( \|Z_k s_z\| = \|s_z\| \). If \( m_o << n \), the subproblem (6.1.2) is a lower-dimensional version of the general trust region model (2.7.1). Obviously, the trust region step can be obtained by solving (6.1.2) since \( s_k = Z_k s_z \).

Here we propose another strategy in the trust region method. The constraint

\[
\|s_z\| \leq \Delta_k
\]

(6.1.3)

implies the trust region is an \( m_o \)-dimensional ball. If the trust region is the outer tangent regular polyhedron of the ball instead of the constraint (6.1.3), the corresponding subproblem is written as

\[
\text{Minimize} \quad \psi_k(s_z) = f_k + g_z^T z + \frac{1}{2} s_z^T B_z s_z
\]

(6.1.4)

Subject to \( \|s_z\|_\infty \leq \Delta_k \).

Remark 6.1.1 The feature of (6.1.4) is that it contains linear constraints only so that finite algorithms are available for its solution. The subproblem (6.1.4) is not strictly equivalent to the subproblem (6.1.2). In particular, it does not have the same transformation invariance properties. However, this does not seem to translate into a serious disadvantage in practice, and questions of numerical convenience would seem to be the most important.
A trust region algorithm with restricted subspace is given below.

Algorithm 6.1.2 (TS-OCSSR1)

Step 0  Let $k$ be specified. Given $\Delta_k > 0$, $x_k \in \mathbb{R}^n$ and a symmetric positive definite matrix $B_k$.

Step 1  Calculate $f_k$ and $g_k$. If the condition for termination is achieved, then stop.

Step 2  Update the matrix $B_k$ by using the OCSSR1 update.

Step 3  Construct the matrix $Z_k$ such that $Z_k^T Z_k = I$.

Step 4  Calculate $g_z = Z_k^T g_k$, $B_z = Z_k^T B_k Z_k$.

Step 5  If $\|B_z^{-1} g_z\|_\infty \leq \Delta_k$, then $s_z = -B_z^{-1} g_z$ and go to Step 7.

Step 6  Solve the subproblem (6.1.4) and obtain $s_z$.

Step 7  Calculate $s_k = Z_k s_z$, $f(x_k + s_k)$ and $t_c = \text{arcd} / \text{pred}_k$.

Step 8  If $t_c < 0.25$, set $\Delta_{k+1} = \|s\|_\infty / 4$; if $t_c > 0.75$ and $\|s\|_\infty = \Delta_k$, set $\Delta_{k+1} = 2 \Delta_k$; otherwise set $\Delta_{k+1} = \Delta_k$.

Step 9  If $t_c \leq 0$, set $x_{k+1} = x_k$; else $x_{k+1} = x_k + s_k$.

Step 10  Set $k = k + 1$ and go to Step 1.

6.2 Choice of Subspace

The first reported use of the subproblem (6.1.1) appears to be due to Bulteau and Vial (1985) who proposed a restricted trust region algorithm by constructing $S_k$ using the steepest descent direction and a quasi-Newton direction.

In choosing the subspace $S_k$, we make use of the result (Cullum and Brayton 1979), that the algorithm has the quadratic termination property if, at each iteration, an exact line search is done and the direction of search is

$$d_k = \mu H_k e_k + \sum \beta_j s_j,$$
where \( \mu \neq 0 \), \( H_k \) belongs to Broyden's family and \( s_j \) are previous steps. Thus, it seems that the subspace should be spanned by a basic descent direction and some linearly independent displacements of \( x_k \) to achieve fast asymptotic convergence. The selection of the basic descent direction depends on the positive definiteness of \( B_k \). Here \( B_k \) is constructed by the update (5.3.3). Since \( B_k \) is always positive definite, the direction

\[
d_k = -B_k^{-1} g_k
\]  

(6.2.1)

is taken as the basic descent direction. A rule to compute the matrix \( Z \) is given below.

**Algorithm 6.2.1** (an additional condition on Step 3 of Algorithm 6.1.2)

Step 3.1 Calculate the basic descent direction \( d_k \) by (6.2.1).

Step 3.2 Select the linearly independent vectors \( d_k, s_1, s_2, \ldots, s_{m-1} \) from \( d_k \), \( s_{k-1}, s_{k-2}, \ldots, s_{k-m} \).

Step 3.3 Using \( d_k, s_1, s_2, \ldots, s_{m-1} \), construct \( m \) column vectors of \( Z_k \) by the Gram-Schmidt orthogonalization procedure.

\[ \square \]

**Remark 6.2.2** If \( m = 1 \), namely \( d_k \) and every \( s_j \), \( 1 \leq j \leq k-m \) are linearly dependent, find a vector \( \hat{s} \) which is linearly independent to \( d_k \). In general, set \( \hat{s} = -g_k \) if it is possible.

### 6.3 Convergence Analysis

To study the convergence of Algorithm 6.1.2, we need the following lemma:

**Lemma 6.3.1** In Algorithm 6.1.2, if there is a constant \( c_1 \in (0, 1] \) and \( c_2 \in (0, 1] \) such that
\[ d_k^T \tilde{g}_k \leq -c_1 \|d_k\| \|g_k\| \]  

(6.3.1)

and

\[ s_z^T g_z \leq -c_2 \|s_z\| \|g_z\|, \]  

(6.3.2)

then

\[ f_k - \psi_k(s) \geq \frac{1}{2} c_0 \|g_k\| \min\{\Lambda_k, c_0 \|g_k\|/\|B_k\|\}. \]  

(6.3.3)

where \( c_0 = c_1 c_2 \).

**Proof.** Define

\[ p(x_z) = -g_z^T x_z - \frac{1}{2} x_z^T B_z x_z, \]

\[ \Gamma = \{v_z | g_z^T v_z \leq -c_1 \|g_z\| \|v_z\|\}, \]

\[ v_k = Z_k v_z \]

and

\[ h(\gamma) = \gamma^T g_z v_z + \frac{1}{2} \gamma^T B_z v_z. \]

Since

\[ \|g_z\| = \|Z_k^T g_k\| = \sqrt{\sum_{j=1}^m (\gamma_j^T c_k)^2} \]

and

\[ z_1 = d_k/\|d_k\|, \]

where \( z_j \) (\( j = 1, \ldots, m \)) are the columns of \( Z_k \). Hence

\[ \|g_z\| \geq \|d_k^T g_k\|/\|d_k\| \geq c_1 \|g_k\|. \]  

(6.3.4)

If \( \gamma^* \) solves the problem

\[ \text{minimize } h(\gamma) \]

subject to \( \|\gamma v_z\|_\infty \leq \Delta_k \)

then
Consider the problem

\[
\begin{align*}
\text{Minimize} & \quad g_z^T w_z + \frac{1}{2} w_z^T B_z w_z \\
\text{Subject to} & \quad \|w_z\|_\infty \leq \Delta_k, \\
& \quad w_z \in \Gamma.
\end{align*}
\]

Let its solution be \( w_z^* = \gamma^* v_z \). In the case of (6.3.5), by (6.3.4) we have

\[
p(w_z^*) = p(\gamma^* v_z) = \frac{1}{2} [(g_z^T v_z)^2/v_z^T B_z v_z] \\
\geq \frac{1}{2} [c_1^2 \|g_z\|^2/\|B_z\|] \\
\geq \frac{1}{2} c_1^2 \|g_z\|^2/\|B_z\|.
\]

In the case of (6.3.6), since

\[
\Delta_k/\|v_z\|_\infty < -g_z^T v_z/\|v_z\|_\infty,
\]

so

\[
p(w_z^*) = -\Delta_k/\|v_z\|_\infty (g_z^T v_z) - \frac{1}{2} \Delta_k/\|v_z\|_\infty^2 (v_z^T B_z v_z) \\
\geq -\frac{1}{2} \Delta_k/\|v_z\|_\infty (g_z^T v_z) \\
\geq -\frac{1}{2} c_1^2 c_2 \Delta_k \|g_z\|.
\]

Thus,

\[
p(w_z^*) \geq \frac{1}{2} c_1^2 c_2 \|g_z\| \min \{\Delta_k, c_2 \|g_z\|/\|B_z\|\}. \tag{6.3.7}
\]

Since

\[
f_k - \psi_k(s) = f_k - \left[ f_k + g_z^T s_z + \frac{1}{2} s_z^T B_z s_z \right]
\]
and \( s_z \) is the solution of (6.1.4), by (6.3.7) and the assumption (6.3.1) and (6.3.2), we obtain (6.3.3).

\[ \square \]

Remark 6.3.2 The introduction of two constants \( c_1, c_2 \) is somewhat unsatisfactory. Ideally the downhill condition in the subspace should be easy to satisfy if it holds in \( \mathbb{R}^n \).

In fact we can take \( c_1 = c_2 \) if \( g_k \in S_k \) for then \( \| g_z \| = \| g_k \| \). There is not much more of an assumption made if \( g_k \) is close to \( S_k \). But the design of \( S_k \) is intended to achieve this result because it aims to adapt to the direction of descent for \( f(x) \). In this sense the assumptions of the theorem are reasonable.

Lemma 6.3.1 shows that the inequality (2.7.4) can be derived from Algorithm 6.1.2 with the condition (6.3.1). Therefore, we can establish directly the following convergence result by using Theorem 2.7.3:

**Theorem 6.3.3** Let \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) be twice continuously differentiable and bounded below. for \( x_0 \in \mathbb{R}^n \) and some \( \beta_1, \beta_2 > 0 \), let \( G(x) \) be uniformly continuous and satisfy \( \| G(x) \| \leq \beta_1 \) for all \( x \) with \( f(x) \leq f(x_0) \). Let \( \{ x_k \} \) be the sequence produced by iterating Algorithm 6.1.2 under condition (6.3.1) and starting from \( x_1 \), using \( B_k = G(x_k) \) or any symmetric approximation with \( \| B_k \| \leq \beta_2 \) at each iteration. Then \( \lim_{k \to \infty} \| g_k \| = 0 \).

6.4 Calculation of the Trust Region Step

In Algorithm 6.1.2, an important step involves solving the subproblem (6.1.4). By the criterion for choosing \( \theta_k \) in Algorithm OCSSR1, we know that the projected matrix \( B_z \) is always positive definite. Thus calculating the trust region step is a
quadratic programming problem. There are several good methods which are suitable for (6.1.4) (Fletcher 1972, Bertsekas 1982 and Clark-Osborne 1988). Here the basic idea used is common to a number of standard active set algorithms, and the subproblem with equality constraints is solved by the method of Lagrange multipliers.

For any feasible \( s_z \) in (6.1.4), we define the index set

\[
I_1(s_z) \triangleq \{ i | s_z(i) = \Delta_k \text{ or } s_z(i) = -\Delta_k \} \quad (6.4.1)
\]

\[
I_2(s_z) \triangleq \{ i | -\Delta_k < s_z(i) < \Delta_k \} \quad (6.4.2)
\]

corresponding to bound variables and free variables respectively. Thus it suffices to consider the problem

Minimize

\[
f_k + g_{z,z}^T s_z + \frac{1}{2}s_z^T B_z s_z
\]

Subject to

\[
s_z(i) = \Delta_z, \ i \in I_1
\]

\[-\Delta_k < s_z(i) < \Delta_k, \ i \in I_2
\]

where \( \Delta_z = \Delta_k \) or \( \Delta_z = -\Delta_k \) for each \( i \in I_1 \). For the sake of convenience, we assume that the last \((m_0-t)\) constraints are active. By the active set strategy, we only have to solve a set of subproblems with equality constraints as follows.

Minimize

\[
f_k + g_{z,z}^T s_z + \frac{1}{2}s_z^T B_z s_z
\]

Subject to

\[
E_z s_z = \zeta,
\]

where \( E_z \in \mathbb{R}^{(m_0-t) \times m_0} \) and \( E_z = [0 \ I_{m_0-t}] \), \( \zeta \in \mathbb{R}^{(m_0-t)} \) and \( \zeta(i) = \Delta_z \) and \( t \) is the number of free variables.
Let \( s_z \) be a feasible point in (6.4.4) and \( s = s_z + h_z \), then (6.4.4) becomes

\[
\begin{align*}
\text{Minimize} & \quad f_z + b_z^T h_z + 1/2 h_z^T B_z h_z \\
\text{Subject to} & \quad E_z h_z = 0,
\end{align*}
\]

where

\[
f_z = f_k + g_z^T s_z + 1/2 (s_z^T B_z s_z
\]

and

\[
b_z = g_z + B_z s_z.
\]

The Lagrange equations of (6.4.5) are

\[
\begin{align*}
B_z h_z - E_z^T e_z &= -b_z \\
E_z h_z &= 0,
\end{align*}
\]

where \( e_z \in \mathbb{R}^{m \times t} \) is the Lagrange multiplier vector. Let

\[
B_z = \begin{pmatrix} B_z^{(1)} & B_z^{(12)} \\ B_z^{(12)^T} & B_z^{(m \times t)} \end{pmatrix} \quad \text{and} \quad b_z = \begin{pmatrix} b_t \\ b_{m \times t} \end{pmatrix},
\]

where \( B_z^{(1)} \in \mathbb{R}^{t \times t} \) and \( b_t \in \mathbb{R}^t \). Then the solution to (6.4.6) are

\[
e_z = b_{m \times t} + B_z^{(12)^T} h_t
\]

and

\[
h_z = \begin{pmatrix} h_t \\ 0 \end{pmatrix},
\]
where \( h \) is defined by

\[
B^{(t)}_{z} h_{t} = -b_{t} \tag{6.4.9}
\]

Since \( B^{(t)}_{z} \) is positive definite, \( h_{t} \) is obtained by using the Choleski factorization of \( B^{(t)}_{z} \).

After obtaining the search direction, the calculations have been carried out using the homotopy algorithm described in Clark and Osborne (1988).

### 6.5 Numerical results

In this section, we report some numerical experiments that were performed using our algorithm. The number of terms with memory is decided by the following criterion:

\[
m = \begin{cases} 
  2, & \text{if } 2 \leq n \leq 10 \\
  3, & \text{if } n > 10.
\end{cases} \tag{6.5.1}
\]

The number of terms is not too critical, but there is some advantage in increasing it as the dimension of the problem increases. The convergence criterion is

\[
\|g\| < 10^{-8} \tag{6.5.2}
\]

The table 6.1 gives results on several test functions for a range of different dimensions of the parameter vector. For example, the function of Rosenbrock is considered for dimensions ranging from 2 to 90, etc. Numerical tests show that the algorithm is suitable for medium-sized unconstrained optimization problems. Also it will be seen that the results compare favourably with those published by other authors (Bulteau-Vial 1985, Dennis-Mei 1979 and Wolfe-Viazminsky 1976).
Table 6.1

(numerical results for TS-OCSSR1)

($x_0$ is standard initial point and $\|g\| < 10^{-8}$)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>n</th>
<th>$N_t$</th>
<th>$N_f$</th>
<th>$N_g$</th>
<th>f</th>
<th>$|g|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF.A2</td>
<td>4</td>
<td>30</td>
<td>31</td>
<td>31</td>
<td>$0.33 \times 10^{-23}$</td>
<td>$0.29 \times 10^{-11}$</td>
</tr>
<tr>
<td>TF.A5</td>
<td>10</td>
<td>25</td>
<td>26</td>
<td>26</td>
<td>$0.10 \times 10^{-21}$</td>
<td>$0.46 \times 10^{-10}$</td>
</tr>
<tr>
<td>TF.A6</td>
<td>64</td>
<td>54</td>
<td>55</td>
<td>55</td>
<td>$0.18 \times 10^{-16}$</td>
<td>$0.44 \times 10^{-10}$</td>
</tr>
<tr>
<td>TF.A7</td>
<td>2</td>
<td>22</td>
<td>23</td>
<td>23</td>
<td>$0.47 \times 10^{-27}$</td>
<td>$0.23 \times 10^{-12}$</td>
</tr>
<tr>
<td>TF.A11</td>
<td>4</td>
<td>55</td>
<td>56</td>
<td>56</td>
<td>$0.24 \times 10^{-15}$</td>
<td>$0.81 \times 10^{-9}$</td>
</tr>
<tr>
<td>TF.A12</td>
<td>20</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>$0.47 \times 10^{-24}$</td>
<td>$0.21 \times 10^{-9}$</td>
</tr>
<tr>
<td>TF.A13</td>
<td>20</td>
<td>52</td>
<td>53</td>
<td>53</td>
<td>$0.55 \times 10^{-16}$</td>
<td>$0.48 \times 10^{-9}$</td>
</tr>
<tr>
<td>TF.A14</td>
<td>4</td>
<td>46</td>
<td>47</td>
<td>47</td>
<td>$0.41 \times 10^{-24}$</td>
<td>$0.20 \times 10^{-10}$</td>
</tr>
</tbody>
</table>
CHAPTER SEVEN
AN APPLICATION OF THE OCSSR1 UPDATE
FOR
LARGE SCALE OPTIMIZATION PROBLEMS

Here we study how to use the OCSSR1 update to construct a hybrid QN algorithm which is suitable for large scale optimization problems. One problem with QN methods is that the successive $H_k$ can fill in in large scale problems. One possibility is to cycle several steps of a QN formula when their information can be held in a few vectors. Nocedal (1980) gives an example of this kind. Here a two-step QN updating formula without storage of any matrices is presented. Numerical results show that this algorithm compares very favourably with the improved CG algorithms, and is competitive with the limited memory QN algorithm.

7.1 A Scaling Conjugate Gradient Direction

Shanno (1978) proposed a scaling conjugate gradient (SCG) algorithm for solving (1.1.1), in which the search direction is given by

$$
\begin{align*}
q_1 &= -g_1 \\
q_k &= \left[ \begin{array}{c}
\frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \bar{g}_k - \left( 2 \frac{s_{k-1}^T \bar{g}_k}{s_{k-1}^T y_{k-1}} \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \right) s_{k-1} + \frac{s_{k-1}^T \bar{g}_k}{y_{k-1}^T y_{k-1}} y_{k-1}
\end{array} \right]
\end{align*}
$$

(7.1.1)

The direction $q_k$ possesses the following properties:

Property 7.1.1 $q_k$ is a scaling conjugate gradient direction when it is applied to the quadratic function (1.4.1) with exact line search. That is,
\[ q_k = \left( \frac{T}{s_{k-1}^T} \right) d_{cg}^{(k)} \]

(7.1.2)

where \( d_{cg}^{(k)} \) is a classical conjugate gradient direction which is defined by (3.1.1).

**Property 7.1.2** \( q_k \) is a quasi-Newton direction. In fact,

\[ q_k = -H_{\text{shanno}}^{(k)} s_k \]

(7.1.3)

where

\[ H_{\text{shanno}}^{(k)} = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} I - \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} + \frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} + 2 \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}} \]

(7.1.4)

and

\[ H_{\text{shanno}}^{(k)} y_{k-1} = s_{k-1} \]

(7.1.5)

**Property 7.1.3** (7.1.1) always gives a descent direction if \( s_{k-1}^T y_{k-1} > 0 \).

**Property 7.1.4** The inverse of the matrix (7.1.4) is

\[ B_{\text{shanno}}^{(k)} = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} \left( I - \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T s_{k-1}} + \frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \right) \]

(7.1.6)

### 7.2 A Quasi-Newton Direction

Replacing \( H_k \) in the OCSSR1 update (5.3.1) with \( H_{\text{shanno}}^{(k)} \) we have

\[ H_{\text{srcgl}}^{(k+1)} = \theta_k H_{\text{shanno}}^{(k)} + \frac{(s_k - \theta_k H_{\text{shanno}}^{(k)} y_k)(s_k - \theta_k H_{\text{shanno}}^{(k)} y_k)^T}{s_k - \theta_k H_{\text{shanno}}^{(k)} y_k} \]

(7.2.1)
The quasi-Newton direction which follows (7.2.1) is

\[
d_{(k+1)}^{(srcg1)} = -\theta_h p_{k+1} - \frac{(s_k - \theta_h H_{sshanno} y_k)^T g_{k+1}}{(s_k - \theta_h H_{sshanno} y_k)^T y_k} (s_k - \theta_h H_{sshanno} y_k),
\]

where

\[
p_{k+1} = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} g_{k+1} + \left(2 \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} g_{k+1} \right)^T s_{k-1} - \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} y_{k-1}.
\]

The formula (7.2.2) provides a quasi-Newton direction, which is available as an alternative to the conjugate gradient method.

7.3 One-Step Quasi-Newton iteration

We use \( q_k \) of (7.1.1) as the basic search direction. If Powell's criterion (3.1.5) holds at \( x_k \), the scaling conjugate gradient iteration will be continued. Otherwise, the quasi-Newton search will be done along the direction \( d_{(k)}^{(srcg1)} \). The resulting algorithm is as follows.

**Algorithm 7.3.1 (SRCG1)**

1. **Step 0** Let \( k \) be specified. Given \( x_k \in \mathbb{R}^n \), calculate \( g_k \) and set \( key = 0 \).
2. **Step 1** If the convergence criterion is achieved, then stop.
3. **Step 2** Compute the scaling conjugate gradient direction \( q_k \) by using (7.1.1). If \( key = 0 \), then set \( key = 1 \), \( \cdot_k = q_k \) and go to step 6.
4. **Step 3** If \( g_k^T g_{k-1} < 0.2 \), then go to step 6.
5. **Step 4** Calculate \( \theta_k \) by using the criterion in Algorithm OCSSR1.
6. **Step 5** Set \( key = 0 \). Calculate the quasi-Newton direction \( d_{(k)}^{(srcg1)} \) by using (7.2.2), and go to step 7.
7. **Step 6** Set \( s_{k-1} = s_k \), \( y_{k-1} = y_k \).
Step 7 Select $\alpha_k > 0$ such that Rule (1.2.4) is satisfied.

Step 8 Set $s_k = \alpha_k d_k$, $x_{k+1} = x_k + s_k$, calculate $g_{k+1}$ and $y_k$.

Step 9 Set $k = k + 1$, go to step 1.

Remark 7.3.2 We set $d_1 = -g_1$ when $k = 1$.

Remark 7.3.3 To obtain a reliable scaling conjugate gradient search direction, if

$$ q_k^T g_k \geq -0.04\|q_k\|\|g_k\|, $$

then the algorithm is restarted with $d_k = -g_k$.

7.4 Two-step Quasi-Newton Iterations

In Algorithm 7.3.1, no matter what happens, whether or not Powell’s criterion is satisfied after the QN step, we always perform the scaling conjugate gradient step. Presumably this is not an optimal strategy. In fact, we can repeat the quasi-Newton iteration by using the OCSSR1 formula again with only a modest increase in cost and complexity. By using

$$ H_{\text{srcg2}}^{(k+1)} = \theta_k H^{(k)}_{\text{srcg1}} + \frac{v_k v_k^T}{v_k v_k^T} + \frac{v_k v_k^T}{v_k v_k^T}, $$

(7.4.1)

where

$$ v_k = s_k - \theta_k H^{(k)} \text{shanno} v_k, $$

(7.4.2a)

and
The relevant quasi-Newton direction is

$$d_{\text{srcg2}}^{(k+1)} = -\theta_k \theta_{k-1} P_{k+1}^* - \frac{V_{k-1}^T g_{k+1}}{V_{k-1}^T V_{k-1}} v_{k-1}^* - \frac{V_k^T g_{k+1}}{V_k^T V_k} \hat{v}'$$

(7.4.3)

where

$$P_{k+1}^* = H^{(k-1)}_{\text{shanno}} g_{k+1}$$

$$= \frac{s_{k-2}^T y_{k-2}}{y_{k-2}^T y_{k-2}} g_{k+1} + \left( 2 \frac{s_{k-2}^T y_{k-2}}{s_{k-2}^T y_{k-2}} - \frac{y_{k-2}^T s_{k-2} y_{k-2}}{y_{k-2}^T y_{k-2}} \right) s_{k-2} - \frac{y_{k-2}^T s_{k-2} y_{k-2}}{y_{k-2}^T y_{k-2}} y_{k-2}$$

Thus the variation of Algorithm 7.3.1 is obtained.

Algorithm 7.4.1 (SRCG2)

It is the same as Algorithm 7.3.1 except Step 4 and Step 5.

Step 4 If key = 1, then calculate $\theta_{k-1}$; else calculate $\theta_k$ by using the criterion in OCSSR1 Algorithm.

Step 5 Calculate the quasi-Newton step.

5.1 Calculate the quasi-Newton direction $d_{k+1}^*$ by using (7.2.2) or (7.4.3) corresponding to key = 1 or key = 2.

5.2 Set key = key + 1. If key $\geq 3$, set key = 0, and go to step 7.

7.5 Quadratic Termination Property

In this section, we discuss only the quadratic function (1.4.1). At the k-th iteration we have
\[ g_k = A x_k + b, \]  
\[ d_k = -H g_k, \]  
\[ x_{k+1} = x_k + \alpha_k d_k, \]  
\[ y_k = A s_k, \]

(7.5.1a)  
(7.5.1b)  
(7.5.1c)  
(7.5.1d)

where \( \alpha_k \) is calculated by an exact line search, that is

\[ \alpha_k = -g_k^T d_k / d_k^T A d_k \]  

(7.5.2)

and

\[ g_{k+1}^T d_k = 0. \]  

(7.5.3)

The proof of quadratic termination is based on an inductive argument which is used to verify the propositions

\[
\begin{aligned}
\{ & s_i^T A s_j = 0, & 1 \leq i \leq n, \\
& g_i^T g_j = 0, & 1 \leq i \leq n+1,
\end{aligned}
\]

(QT)

Because SCG is used in the first step, (QT) is an immediate consequence of (7.1.2) when \( n = 2 \). Let (QT) be satisfied with \( n = k-1 \), then the following lemmas are necessary to prove (QT) holds when \( n = k \).

**Lemma 7.5.1** Assume that (QT) holds with \( n = k-1 \) and set \( t_j = s_j^T y_j / y_j^T y_j \). Then the following expressions can be obtained by direct calculation:

for \( j \leq k-3 \) we have

\[ g_k^T H^{(k)}_{\text{shanno}} y_j = 0 \]  

(7.5.4)

and
\[ y_k^{(k)} = 0; \]  
\[
\forall j < k-2 \text{ we have } \quad v_{j-1}^T y_{k-1} = - \theta_{j-1} T_{j-1} y_{k-1}, \quad (7.5.6)
\]

\[ v_{k-1}^T y_j = 0 \]  
\[
\forall j < k-1 \text{ we have } \quad v_{j-1}^T y_{k-1} = - \theta_{j-1} T_{j-1} y_{j-1} + \theta_{j-2} T_{j-2} y_{j-2}^T y_{k-1} v_{j-2}; \quad (7.5.8)
\]

\[ H_{\text{shanno}}^{(j-1)} y_{j-1} = t_{j-2} y_{j-1} - \frac{y_{j-2}^T y_{j-1}}{y_{j-2}^T y_{j-2}} s_{j-2}^T \]  
\[
H_{\text{shanno}}^{(j-2)} y_{j-1} = t_{j-3} y_{j-1}, \quad (7.5.10)
\]

\[ v_{j-1}^T g_j = - \theta_{j-1} T_{j-1} g_{j-1}, \quad (7.5.11)\]

\[ d^{(j)}_{\text{srcg1}} = \theta_{j-1} t_{j-2} \left( g_j + \frac{y_{j-1}^T g_j}{y_{j-1}^T y_{j-1}} v_{j-1} \right), \quad (7.5.12)\]

\[ v_{j-1}^T A d^{(j)}_{\text{srcg1}} = - \theta_{j-1} t_{j-2} g_{j-1} A d^{(j)}_{\text{srcg1}} \quad (7.5.13)\]

\[ v_{j-1}^T g_j = - \theta_{j-1} T_{j-1} \theta_{j-2} T_{j-3} y_{j-1}^T g_j. \quad (7.5.14)\]
\[
d^{(j)}_{\text{srcg2}} = \theta_{j-1} \theta_{j-2} t_{j-3} \left( -g_j^T + \frac{y_{j-1}^T g_j}{\hat{v}^T_{j-1} y_{j-1}} \right) \tag{7.5.15}
\]

\[
v_{j-2}^T A^{-1} d^{(j)}_{\text{srcg2}} = 0 \tag{7.5.16}
\]

\[
\hat{v}_{j-1}^T A^{-1} d^{(j)}_{\text{srcg2}} = -\theta_{j-1} \theta_{j-2} t_{j-3} g_j^T A^{-1} d^{(j)}_{\text{srcg1}} \tag{7.5.17}
\]

\[\Box\]

**Lemma 7.5.2** Let the iteration satisfy

\[
\begin{cases}
s_i^T A s_j = 0, \\
i \leq k-1 \text{ and } j < i.
\end{cases}
\tag{7.5.18}
\]

and \(s_j\) is obtained by Algorithm SRCG1. Then, for \(j < k-1\)

\[
y_{k-1}^T g_j = c_1 y_{k-1}^T g_{j-1} \tag{7.5.19}
\]

where \(c_1\) is a constant.

**Proof.** Since \(y_{k-1}^T s_j = 0\), from (7.5.6) and (7.5.12) we have

\[
y_{k-1}^T g_j = c_1 y_{k-1}^T g_{j-1},
\]

where

\[
\begin{align*}
c_1 &= \frac{\hat{c}_1}{1 + \hat{c}_1} \quad \text{and} \quad \hat{c}_1 = \theta_{j-1} t_{j-2} \frac{y_{j-1}^T g_j}{\hat{v}_{j-1}^T y_{j-1}}.
\end{align*}
\]

\[\Box\]
Lemma 7.5.3 Assume that the same conditions as for Lemma 7.5.2, and let $s_j$ be given by Algorithm SRCG2. Then, for $j < k-1$

$$y_{k-1}^T g_j = c_2 y_{k-1}^T g_{j-2}$$  \hfill (7.5.20)

where $c_2$ is a constant.

Proof. Since $y_{k-1}^T s_j = 0$, from (7.5.8) and (7.5.15) we have

$$y_{k-1}^T g_j = c_2 y_{k-1}^T g_{j-2}$$

where

$$c_2 = \frac{\hat{c}_2 (c_1 (1-\hat{c}_3) + \hat{c}_3)}{1 + \hat{c}_2},$$

$$\hat{c}_2 = \theta_{j-1} \theta_{j-2} \frac{y_{j-1}^T g_j}{y_{j-1}^T y_{j-1}}$$ \text{ and } $$\hat{c}_3 = \theta_{j-2} \frac{y_{j-2}^T y_{j-1}}{y_{j-2}^T y_{j-2}}.$$

\[\square\]

Corollary 7.5.4 Let the condition (7.5.18) holds, then there is a constant $c_r$ so that

$$y_{k-1}^T g_j = c_r y_{k-1}^T g_r$$

where $g_r$ is the gradient which was used on the last occasion a SCG step was made.

Lemma 7.5.5 Assume that (QT) is satisfied with $n = k-1$, and $x_{k+1}$ produced by an iteration of SCG. Then (QT) holds when $n = k$. 

Proof. From (7.1.5), (7.5.1) and (7.5.3) we know that

\[
{s_k^T T^k A_j s_{k-1} = - \alpha g^T H^{(k)}_{s h a n n o} y_j = - \alpha g^T s_{k-1} = 0.}
\]  
(7.5.21)

For \( j < k-1 \), by (7.1.2) and (7.5.4) we have

\[
{s_k^T T^k A_j s_{j-1} = - \alpha g^T H^{(k)}_{s h a n n o} y_j = 0.}
\]  
(7.5.22)

Secondly, from (7.1.2) we have

\[
g_k = \frac{1}{s_{k-1} y_{k-1}} \left( (y^T_{k-1} y_{k-1}) s_{k-1} - (y^T_{k-1} y_{k-1} d_k) \right). \]
(7.5.23)

For \( j = 1, 2, \ldots, k \) consider

\[
g_{k+1}^T g_j = y_{k+1}^T g_j + g_{k+1}^T g_j.
\]

If \( j < k \), then

\[
g_{k+1}^T g_j = 0
\]

and by (7.5.21 – 22), Corollary 7.5.4 and (7.5.23) we know that there is a \( \tau \) so that

\[
y_{k+1}^T g_j = c_{\tau} \frac{1}{s_{\tau-1} y_{\tau-1}} \left( (y^T_{\tau-1} y_{\tau-1}) y_{k+1}^T s_{k-1} - (y^T_{\tau-1} y_{\tau-1} d_k) y_{k+1}^T d_\tau \right) = 0.
\]

If \( j = k \), from (7.1.2), (7.5.2) and (7.5.23) we have

\[
y_{k+1}^T g_j = - g_{k+1}^T g_j,
\]

i.e.,
Lemma 7.5.6 Assume that (QT) is satisfied with \( n = k - 1 \), \( x_{k+1} \) produced by an iteration of OCSSR1. Then (QT) holds when \( n = k \).

Proof. Since

\[
H_k A_k s_{k-1} = s_{k-1}^	op
\]

so

\[
s_{k-1}^T A_k s_{k-1} - \alpha_k g_k^T H_k A_k s_{k-1} = 0.
\]

Secondly, from (5.3.4), (7.5.1d) we have

\[
H_k y_{k-2} = \theta_{k-1} s_{k-2}
\]

and

\[
s_{k-2}^T A_{k-2} - \alpha_k g_k^T H_k y_{k-2} = -\alpha_k \theta_{k-1} (y_{k-1} + g_{k-1})^T s_{k-2} = 0.
\]

For \( j < k-2 \), consider \( s_{k-2}^T A_{j} \). There are two possibilities for \( s_k \):

\[
s_k = \alpha_k d^{(k)}_s \quad \text{and} \quad s_k = \alpha_k d^{(k)}_{s2} \tag{7.5.24, 7.5.25}
\]

By using (7.5.5), (7.5.7), (7.5.10), (7.5.12) and (7.5.15) for (7.5.24) and (7.5.25) we have

\[
s_{k}^T A_{j} s_{j} = 0. \tag{7.5.26}
\]
On the other hand, consider

\[ g_{k+1}^T g_j = y_{k}^T g_j + g_k^T g_j. \]

If \( j < k \), by using (7.5.26) and Corollary 7.5.4 we obtain

\[ g_{k+1}^T g_j = 0. \]

If \( j = k \) and \( d_k = d^{(k)}_{\text{srcgl}} \), by using (7.5.2) and (7.5.11 – 13) we have

\[
g_k^T H_{\text{srcgl}}^T g_k = \theta_{k-1}^T k-2g_k^T g_k \left( 1 + \frac{\theta_{k-1}^T k-2g_k^T g_k}{v_{k-1}^T v_{k-1}} \right), \tag{7.5.27}
\]

\[
d^{(k)}_{\text{srcgl}}^T \text{Ad}^{(k)}_{\text{srcgl}} = - \theta_{k-1}^T k-2g_k^T \text{Ad}^{(k)}_{\text{srcgl}} \left( 1 + \frac{\theta_{k-1}^T k-2g_k^T g_k}{v_{k-1}^T v_{k-1}} \right) \tag{7.5.28}
\]

and

\[ \alpha_k = \frac{g_k^T H_k^T g_k}{d^{(k)}_{\text{srcgl}}^T \text{Ad}^{(k)}_{\text{srcgl}}} \tag{7.5.29} \]

So

\[ y_k^T g_k = \alpha_k g_k^T \text{Ad}^{(k)}_{\text{srcgl}} = - g_k^T g_k, \]

i.e.,

\[ g_{k+1}^T g_k = 0. \]

Similarly, if \( d_k = d^{(k)}_{\text{srcgl2}} \), by using (7.5.14 – 17) we have
\[ Td(k) = Ad(k) = -a^T \begin{pmatrix} \theta_{k-1} & \theta_{k-2}T_{k-3} & g_k^T g_k \\ T & \gamma_{k-1}^T \gamma_{k-1} \end{pmatrix}, \quad (7.5.30) \]

\[ d_{srcg2}^T Ad_{srcg2}^k = -\theta_{k-1}^T \theta_{k-2}T_{k-3}g_k^T Ad_{srcg2}^k \left( 1 + \frac{\theta_{k-1} \theta_{k-2}T_{k-3}g_k^T g_k}{\gamma_{k-1}^T \gamma_{k-1}} \right), \quad (7.5.31) \]

And

\[ \alpha_k^* = \frac{g_k^T H g_k}{d_{srcg2}^T Ad_{srcg2}^k^T g_k}. \quad (7.5.32) \]

So

\[ g_{k+1}^T g_k = 0. \]

The following theorem is a consequence of Lemma 7.5.5 and lemma 7.5.6 which show that SCG and OCSSR1 can be mixed without destroying quadratic termination property.

**Theorem 7.5.7** The Algorithm SRCG1 and SRCG2 possesses quadratic termination with exact line search, and (QT) holds.

### 7.6 Numerical results

This section we report some numerical experiments performed using our algorithm. The report consists of two parts. First, we give numerical comparisons of SRCG2 and some popular improved CG methods on a range of frequently used test problems. Then we will mention numerical results obtained with the two new algorithms when they are used for solving large scale optimization problems. The test functions are taken from group A and group B.
Table 7.1 compares the results of calculation for SRCG2 with the results given by Shanno (1978) for his testing of SCONBP and PRCN. The initial points and the convergence criterion are the same as those used by him. Most starting points are standard $x_0$ except for TF.A6 and TF.A14, in which

<table>
<thead>
<tr>
<th>TF.A6</th>
<th>$x_0^{(a)} = (-3, -1, 0, 1);$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF.A14</td>
<td>$x_0^{(a)} = (-3, -3, 1),$</td>
</tr>
<tr>
<td></td>
<td>$x_0^{(b)} = (-1.2, 1, -1.2, 1),$</td>
</tr>
<tr>
<td></td>
<td>$x_0^{(c)} = (-1.2, 1, 1.2, 1).$</td>
</tr>
</tbody>
</table>

The convergence criterion is

$$\max_i |g_k(i)| < 10^{-5}. \quad (7.6.1)$$

Table 7.2 compares the results of calculation for SRCG2 with the results given by Nocedal (1980) for his testing of SQN, PCG and SCONBP. The initial points and the convergence criterion are the same as that used by him, That is, the standard initial points $x_0$ are used and the convergence criterion is

$$\|g_k\| < 10^{-8} \quad (7.6.2)$$

except for the Powell function, where $\|g_k\| < 10^{-6}$.

Table 7.3 gives numerical comparisons of the algorithms SRCG1 and SRCG2 for solving large scale optimization problems. $f(x)$ denotes the value of the objective function at the termination point. Here the standard initial points $x_0$ are used and the convergence criterion is

$$\|g_k\| < 10^{-5} \times \max(1,\|x_k\|). \quad (7.6.3)$$
Table 7.1
(the comparison with Shanno's results)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>SCONBP</th>
<th></th>
<th></th>
<th>PRCON</th>
<th></th>
<th></th>
<th>SRCG2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N&lt;sub&gt;t&lt;/sub&gt;</td>
<td>N&lt;sub&gt;f&lt;/sub&gt;</td>
<td>N&lt;sub&gt;g&lt;/sub&gt;</td>
<td>N&lt;sub&gt;t&lt;/sub&gt;</td>
<td>N&lt;sub&gt;f&lt;/sub&gt;</td>
<td>N&lt;sub&gt;g&lt;/sub&gt;</td>
<td>N&lt;sub&gt;t&lt;/sub&gt;</td>
<td>N&lt;sub&gt;f&lt;/sub&gt;</td>
<td>N&lt;sub&gt;g&lt;/sub&gt;</td>
</tr>
<tr>
<td>TF.A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>(n=4)</td>
<td>64</td>
<td>160</td>
<td>160</td>
<td>50</td>
<td>188</td>
<td>188</td>
<td>74</td>
<td>106</td>
</tr>
<tr>
<td>TF.A10</td>
<td>(n=10)</td>
<td>9</td>
<td>19</td>
<td>19</td>
<td>12</td>
<td>77</td>
<td>77</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>(n=20)</td>
<td>10</td>
<td>22</td>
<td>22</td>
<td></td>
<td>F</td>
<td></td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>(n=30)</td>
<td>11</td>
<td>26</td>
<td>26</td>
<td></td>
<td>F</td>
<td></td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>TF.A12</td>
<td>(n=20)</td>
<td>18</td>
<td>38</td>
<td>38</td>
<td>14</td>
<td>45</td>
<td>45</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>(n=50)</td>
<td>26</td>
<td>53</td>
<td>53</td>
<td>24</td>
<td>61</td>
<td>61</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>TF.A13</td>
<td>(n=5)</td>
<td>22</td>
<td>50</td>
<td>50</td>
<td>28</td>
<td>75</td>
<td>75</td>
<td>32</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>(n=10)</td>
<td>33</td>
<td>69</td>
<td>69</td>
<td>43</td>
<td>114</td>
<td>114</td>
<td>34</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>(n=15)</td>
<td>94</td>
<td>190</td>
<td>190</td>
<td></td>
<td>F</td>
<td></td>
<td>35</td>
<td>47</td>
</tr>
<tr>
<td>TF.A14</td>
<td>(n=4)</td>
<td>101</td>
<td>235</td>
<td>235</td>
<td>144</td>
<td>529</td>
<td>529</td>
<td>54</td>
<td>85</td>
</tr>
<tr>
<td>x&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td></td>
<td>113</td>
<td>195</td>
<td>195</td>
<td>174</td>
<td>654</td>
<td>654</td>
<td>64</td>
<td>90</td>
</tr>
<tr>
<td>x&lt;sup&gt;(b)&lt;/sup&gt;</td>
<td></td>
<td>93</td>
<td>219</td>
<td>219</td>
<td>156</td>
<td>567</td>
<td>567</td>
<td>214</td>
<td>296</td>
</tr>
<tr>
<td>x&lt;sup&gt;(c)&lt;/sup&gt;</td>
<td></td>
<td>48</td>
<td>118</td>
<td>118</td>
<td>68</td>
<td>241</td>
<td>241</td>
<td>78</td>
<td>106</td>
</tr>
<tr>
<td>TF.B3</td>
<td>(n=5)</td>
<td>124</td>
<td>274</td>
<td>274</td>
<td>232</td>
<td>855</td>
<td>855</td>
<td>31</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>(n=10)</td>
<td></td>
<td>F</td>
<td></td>
<td>31</td>
<td>41</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7.2
(the comparison with Nocedal's results)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>PCG</th>
<th>SQN</th>
<th>SHANNO</th>
<th>SRCG2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_f$</td>
<td>$N_g$</td>
<td>$N_f$</td>
<td>$N_g$</td>
</tr>
<tr>
<td>TF.A3</td>
<td>(n=6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>60</td>
<td>60</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>49</td>
<td>49</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>46</td>
<td>46</td>
<td>68</td>
</tr>
<tr>
<td>TF.A6</td>
<td>(n=4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>82</td>
<td>82</td>
<td>122</td>
<td>122</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>76</td>
<td>76</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>68</td>
<td>68</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>(n=8)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>115</td>
<td>115</td>
<td>116</td>
<td>116</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>93</td>
<td>93</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>79</td>
<td>79</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>(n=16)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>113</td>
<td>113</td>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>99</td>
<td>99</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>92</td>
<td>92</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>(n=20)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>106</td>
<td>106</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>105</td>
<td>105</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>98</td>
<td>98</td>
<td>92</td>
</tr>
<tr>
<td>TF.A8</td>
<td>(n=3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>59</td>
<td>59</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>53</td>
<td>53</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>51</td>
<td>51</td>
<td>44</td>
</tr>
<tr>
<td>TF.A14</td>
<td>(n=4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1=2$,</td>
<td>146</td>
<td>146</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>$M_2=3$</td>
<td>(M=4)</td>
<td>181</td>
<td>181</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>(M=8)</td>
<td>155</td>
<td>155</td>
<td>56</td>
</tr>
</tbody>
</table>

Remark 7.6.1 M denotes the number of corrections stored and only is used for Algorithms PCG and SQN, where $M_1$ for SCG and $M_2$ for SQN.
Table 7.3
(numerical results for SRCG1 and SRCG2)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>SIZE (n)</th>
<th>SRCG1</th>
<th>SRCG2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$N_t$</td>
<td>$N_f$</td>
</tr>
<tr>
<td>TF.A6</td>
<td>100</td>
<td>101</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>144</td>
<td>181</td>
</tr>
<tr>
<td>TF.A7</td>
<td>100</td>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td>TF.A13</td>
<td>100</td>
<td>66</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>74</td>
<td>112</td>
</tr>
<tr>
<td>TF.B1</td>
<td>100</td>
<td>53</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>70</td>
<td>116</td>
</tr>
<tr>
<td>TF.B2</td>
<td>100</td>
<td>142</td>
<td>194</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>103</td>
<td>154</td>
</tr>
<tr>
<td>TF.B3</td>
<td>100</td>
<td>21</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>19</td>
<td>27</td>
</tr>
<tr>
<td>TF.B4</td>
<td>100</td>
<td>123</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>454</td>
<td>515</td>
</tr>
<tr>
<td>TF.B5</td>
<td>100</td>
<td>34</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>39</td>
<td>98</td>
</tr>
</tbody>
</table>
7.7 Conclusion

A two-step QN updating formula on the OCSSR1 without storage of any matrix is presented. Numerical experiments show that SRCG2 generally outperforms Shanno’s method and is somewhat faster than Nocedal’s PCG algorithm with \( M = 2,4 \) except for the Powell function. It should be noted that the algorithm SRCG2 proves very stable for the extended Powell function. It also compares favourably with PCG (\( M = 8 \)) and SQN (\( M = 3 \)). Thus, it is very efficient and appears suitable for large scale optimization problems.
CHAPTER EIGHT
AN APPLICATION OF THE OCSSR1 UPDATE
IN
NONLINEAR LEAST SQUARES PROBLEM

The GN algorithm (see Chapter 4) has been widely used to minimize objective functions having the form of a sum of squares. This method is ubiquitous in the case of statistical estimation (Osborne 1987). However, in other problems it is necessary to distinguish between problems in which the sum of squares is very small or zero at the minimum and those in which it is not. In the latter case the GN algorithm can be very slowly convergent. Two main kinds of approach have been employed to improve the convergence properties of the GN algorithm on large-residual problems: (1) Quasi-Newton style updates can be used to estimate the second derivative terms ignored in displaying the Hessian of $J^T J$. This is the approach of Biggs (1977), Dennis-Gay-Welsch (1981) and Dennis-Martinez-Tapia (1989), etc. (2) Hybrid algorithms can be employed which switch between GN and a quasi-Newton method depending on which appears to have the faster rate of convergence. It will be shown that the SR1 style of updating is useful in both approaches.

8.1 An Idea for Improving the Hessian Estimate

In NLS problem (4.1.1), as mentioned in Chapter 4, it is a development of the GN method that a quasi-Newton update is used to approximate to $\sum_{i=1}^{m} r_i^2 r_i(x)$. That is, we calculate an estimate $E_k$ of $\sum_{i=1}^{m} r_i^2 r_i(x)$, which is symmetric and satisfies the quasi-Newton equation

$$E_{k,s}^k = \bar{y}_{k-1}$$  \hspace{1cm} (8.1.1)
where \( \hat{y}_{k-1} \) is suitably defined. Several alternative definitions of the vector \( \hat{y}_{k-1} \) and the updating schemes for the matrix \( E_k \) have been introduced in Section 4.3. Here we suggest using
\[
\hat{y}_{k-1} = y^\#_{k-1} \triangleq (J_k - J_{k-1})^T r_k,
\]
which is due to Biggs (1977), and Choosing \( E_k \) to be a solution of the least-change problem
\[
\begin{align*}
\text{minimize} & \quad \|W^{1/2}(E - \omega_{k-1} E_{k-1})W^{1/2}\|_F \\
\text{subject to} & \quad (E - \omega_{k-1} E_{k-1})^T = E - \omega_{k-1} E_{k-1} \\
& \quad E_{s_{k-1}} = y^\#_{k-1},
\end{align*}
\]
where \( W \in \mathbb{R}^{n \times n} \) is the symmetric positive definite matrix satisfying
\[
(W^{-1} + \omega_{k-1} E_{k-1}) s_{k-1} = y_{k-1}, \text{ and } y^\#_{k-1} = (J_k - J_{k-1})^T r_k. \text{ That is (formally just the same as SSR1),}
\]
\[
E_k = \omega_{k-1} E_{k-1} + \frac{(y^\#_{k-1} - \omega_{k-1} E_{k-1} s_{k-1}) (y^\#_{k-1} - \omega_{k-1} E_{k-1} s_{k-1})^T}{(y^\#_{k-1} - \omega_{k-1} E_{k-1} s_{k-1})^T s_{k-1}}.
\]
Thus an approximation to \( G_k \) is given by
\[
B_k = J_k^T J_k + E_k.
\]
To make the algorithm globally convergent, a model with the trust region strategy is used; at each iteration the problem
\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} r_k^T r_k + (J_k^T r_k)^T s + \frac{1}{2} s^T (J_k^T J_k + E_k) s \\
\text{subject to} & \quad \|s\| \leq \Delta_k
\end{align*}
\]
is solved for \( s_k \), so that

\[
s_k = -(J_k^T J_k + E_k + \mu_k D)^{-1} J_k^T r_k
\]

for some \( \mu_k \geq 0 \).

Remark 8.1.1 Except for the form of \( E_k \), this idea is similar to Strategy 4.3.4 and 4.3.6 which is suggested by Dennis—Gay—Welsch (1981) and Dennis—Martinez—Tapia (1989) respectively. They report encouraging results. For reasons of time this suggestion has not been tested numerically.

8.2 A Switch for the hybrid approach

We are interested in developing a hybrid method which asymptotically takes an OCSSR1 step for a large-residual problem and the GN steps for a zero-residual problems. The key doing this is lies in distinguishing clearly the size of residuals in the numerical calculation.

If the subfunction \( r_i(x) \) is quadratic, then

\[
(V_i^2 r_i(x))s_{k-1} = V_i r_i(x_k) - V_i r_i(x_{k-1}),
\]

hence

\[
(\sum_{i=1}^{m} r_i(x_k) V_i^2 r_i(x_k)) s_{k-1} = (J_k - J_{k-1})^T r_k.
\]

Let

\[
r_k^{(1)} = \frac{|s_{k-1}^T (\sum_{i=1}^{m} r_i(x_k) V_i^2 r_i(x_k)) s_{k-1}|}{\|s_{k-1}\|^2}
\]

\[
= \frac{|r_k^T (J_k - J_{k-1}) s_{k-1}|}{\|s_{k-1}\|^2},
\]
the expression (8.2.2) is getting small when (4.1.1) is a zero-residual problem. On the other hand, when \( r_i(x) \) is quadratic, we also have

\[
\left( J_{k_k}^T J_{k_k} + \sum_{i=1}^{m} r_i(x_i) \nabla^2 r_i(x_i) \right) s_{k-1} = J_{k_k}^T r_{k_k} - J_{k-1_k}^T r_{k-1}.
\]

so

\[
(J_{k_k}^T J_{k_k}) s_{k-1} = J_{k_k}^T (r_{k_k} - r_{k-1})
\]

and the measure

\[
\tau_k^{(2)} = \| (J_{k_k} - J_{k-1_k}) r_{k_k} \| / \| J_{k-1_k}^T (r_{k_k} - r_{k-1}) \| \]

is also getting small for a zero-residual problem. (8.2.2) and (8.2.4) can be used to estimate whether NLS is a zero residual or not in numerical calculation.

**Switch 8.2.1** A current estimate \( B_k \) of the Hessian matrix is defined by

\[
B_k = \begin{cases} 
J_{k_k}^T J_{k_k}, & \text{if } (\tau_{k-1}^{(1)}/\tau_{k-1}^{(1)}) > \varepsilon_1 \text{ and } (\tau_{k-1}^{(2)}/\tau_{k-1}^{(2)}) > \varepsilon_2, \\
\text{OCSSR1}, & \text{otherwise},
\end{cases}
\]

where \( \varepsilon_1 \) and \( \varepsilon_2 \in (0, 1) \) is two preset parameters. The test for Switch (8.2.1) is simple and negligible extra operations are needed.

**8.3 A Hybrid Algorithm**

A hybrid algorithm can be established by using Switch 8.2.1.

**Algorithm 8.3.1 (SSR1GN)**

Step 0 Let \( k \) be specified. Given \( x_k \in \mathbb{R}^n \) and a symmetric and positive definite matrix \( H_k \). Set \( \varepsilon_1 \) and \( \varepsilon_2 \in (0, 1) \), and calculate \( J_k \).
Step 1: If the convergence criterion is satisfied, then stop.

Step 2: Calculate $r_k^{(1)}$ and $r_k^{(2)}$ by using (8.2.2) and (8.2.4) respectively. If (8.2.5a) is satisfied, then go to step 4.

Step 3: Calculate the QN direction $d_k$ by $d_k = -H_k g_k$ and go to step 6.

Step 4: Calculate the GN direction $d_k$ by solving the overdetermined linear equations

$$J_k d_k = - r_k^*.$$  \hspace{2cm} (8.3.1)

Step 5: Set

$$H_k = (J_k^T J_k)^{-1}.$$  \hspace{2cm} (8.3.2)

Step 6: Select $\alpha_k$ such that it satisfies Rule 1.2.4.

Step 7: Set $s_k = \alpha_k d_k$ and $x_{k+1} = x_k + s_k$. Calculate $g_{k+1}$ and $y_k = J_{k+1}^T r_{k+1} - J_k^T r_k$.

Step 8: Update $H_k$ by using (5.3.1) with Strategy 5.6.2.

Step 9: Set $k = k+1$, and go to step 1.

Remark 8.3.2: We set $d_1 = -g_1$ when $k = 1$.

Remark 8.3.3: In step 4, to obtain a GN search direction, we use the QR orthogonal decomposition to $J(x_k)$, that is,

$$J_k = Q_k R_k$$  \hspace{2cm} (8.3.3)

where $Q_k \in \mathbb{R}^{m \times m}$ is an orthogonal matrix and $R_k \in \mathbb{R}^{m \times n}$. If $\text{rank}(J_k) = n$, then

$$R_k = \begin{pmatrix} U_k \\ 0 \end{pmatrix}$$  \hspace{2cm} (8.3.4)

where $U_k \in \mathbb{R}^{m \times n}$ is an upper triangular matrix. Thus a GN direction $d_k$ is defined by
\[ U_k d_k = z_k \]  \hspace{1cm} (8.3.5)

where \( z_k \in \mathbb{R}^n \) and is composed of the first \( n \) components of \( Q_k^T r_k \). If \( \text{rank}(J_k) < n \), we take a quasi-Newton step.

**Remark 8.3.4** We do not need to calculate the inverse of \( J_k^T J_k \). When \( \text{rank}(J_k) = n \), we have

\[ (J_k^T J_k)^{-1} = (U_k^{-1})(U_k^{-1})^T, \]  \hspace{1cm} (8.3.6)

and it is easy to obtain the inverse of an upper triangular matrix.

8.4 **Numerical results**

In this section we report some numerical experiences performed using Algorithm HOSQNGN. The report consists of two parts. Table 8.1 gives results on several test functions for zero–residual problems. The others which belong to nonzero–residual problems are given in Table 8.2. The termination criterion is

\[ \|g_k\| < 10^{-8}. \]  \hspace{1cm} (8.4.1)

The parameters in Switch 8.2.1 are taken as \( \epsilon_1 = 0.2 \) and \( \epsilon_2 = 0.25 \). A numerical implementation of Fletcher-Xu’s switch (4.4.3) with the OCSSR1 update is used to compare with Algorithm SSR1GN. Numerical tests show that Algorithm SSR1GN is suitable for NLS problems.
Table 8.1

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>n/m</th>
<th>OCSSR1</th>
<th>H-XU</th>
<th>SSR1GN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$N_t$</td>
<td>$N_f$</td>
<td>$N_g$</td>
</tr>
<tr>
<td>TF.A1</td>
<td>2/3</td>
<td>15</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>TF.A4</td>
<td>2/3</td>
<td>17</td>
<td>23</td>
<td>18</td>
</tr>
<tr>
<td>TF.A6</td>
<td>4/4</td>
<td>41</td>
<td>47</td>
<td>42</td>
</tr>
<tr>
<td>TF.A7</td>
<td>10/10</td>
<td>26</td>
<td>33</td>
<td>27</td>
</tr>
<tr>
<td>TF.A8</td>
<td>3/3</td>
<td>33</td>
<td>41</td>
<td>34</td>
</tr>
<tr>
<td>TF.A13</td>
<td>5/5</td>
<td>28</td>
<td>32</td>
<td>29</td>
</tr>
<tr>
<td>TF.A14</td>
<td>4/6</td>
<td>35</td>
<td>47</td>
<td>36</td>
</tr>
<tr>
<td>TF.C_1</td>
<td>3/10</td>
<td>35</td>
<td>39</td>
<td>36</td>
</tr>
<tr>
<td>TF.C_2</td>
<td>4/5</td>
<td>61</td>
<td>66</td>
<td>62</td>
</tr>
<tr>
<td>TF.C_3</td>
<td>10/10</td>
<td>21</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>n/m</td>
<td>OCSSR1</td>
<td>H-XU</td>
<td>SSR1GN</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N_t</td>
<td>N_f</td>
<td>N_g</td>
</tr>
<tr>
<td>TF.C_1</td>
<td>3/15</td>
<td>37</td>
<td>43</td>
<td>38</td>
</tr>
<tr>
<td>TF.C_2</td>
<td>8/8</td>
<td>45</td>
<td>52</td>
<td>46</td>
</tr>
<tr>
<td>TF.C_3</td>
<td>4/11</td>
<td>74</td>
<td>84</td>
<td>75</td>
</tr>
<tr>
<td>TF.C_4</td>
<td>2/3</td>
<td>11</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>TF.C_5</td>
<td>5/33</td>
<td>28</td>
<td>43</td>
<td>29</td>
</tr>
<tr>
<td>TF.C_6</td>
<td>11/65</td>
<td>22</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>TF.C_7</td>
<td>9/31</td>
<td>52</td>
<td>81</td>
<td>53</td>
</tr>
<tr>
<td>TF.C_L1</td>
<td>4/20</td>
<td>21</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>TF.C_L2</td>
<td>2/2</td>
<td>6</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>TF.C_L3</td>
<td>2/10</td>
<td>18</td>
<td>24</td>
<td>19</td>
</tr>
</tbody>
</table>
CHAPTER NINE
NUMERICAL EXPERIMENTS

9.1 Test Functions

The test functions are divided into three groups: group A contains the classical test functions; group B contains the functions which are used to test behaviour when the number of variables is very large; group C consists of the functions that are used to test the ability of an algorithm to solve nonlinear least squares problems, in which subclass $C_z$ denotes functions with zero-residual, $C_s$ denotes functions with small-residual and $C_L$ denotes functions with Large-residual. They are outlined as follows:

Group A

TF.A1 Beale

$$f(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$$

$x_0 = (1, 1)$

$x^* = (3, 0.5)$

$f(x^*) = 0$

TF.A2 Biggs (Exp4)

$$f(x) = \sum_{i=1}^{m} \left( x_3 e^{-t_i x_1} - x_4 e^{-t_i x_2} - e^{-t_i} + 5 e^{-10 t_i} \right)^2$$

$t_i = 0.1i.$

$x_0 = (1, 2, 1, 1)$

$x^* = (1, 10, 1, 5)$, if $m = 10.$

$f(x^*) = 0$
TF.A3 Biggs (Exp6)

\[ f(x) = \sum_{i=1}^{m} \left( x_3 e^{-t_i x_1} - x_4 e^{-t_i x_2} + x_6 e^{-t_i x_5} - (e^{-t_i} - 5e^{-10t_i} + 3e^{-4t_i})^2 \right) \]

\[ t_i = 0.1i. \]

\[ x_0 = (1, 2, 1, 1, 1, 1) \]

\[ x^* = (1, 10, 1, 5, 4, 3), \text{ if } m = 13. \]

\[ f(x^*) = 0 \]

TF.A4 Brown Badly Scaled

\[ f(x) = (x_1 - 10^6)^2 + (x_2 - 2 \cdot 10^{-6})^2 + (x_1 x_2 - 2)^2 \]

\[ x_0 = (1, 1) \]

\[ x^* = (10^6, 2 \cdot 10^{-6}) \]

\[ f(x^*) = 0 \]

TF.A5 Dixon

\[ f(x) = (1 - x_1)^2 + (1 - x_1 10^2 + \sum_{i=1}^{9} (x_i - x_{i+1})^2 \]

\[ x_0 = (-2, ..., -2) \]

\[ x^* = (1, ..., 1) \]

\[ f(x^*) = 0 \]
TF.A6 Extended Powell

\[ f(x) = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \]

\( x_0 = (3, -1, 0, 1, ..., -1, 0, 1) \)

\( x^* : \text{origin} \)

\[ f(x^*) = 0 \]

TF.A7 Extended Rosenbrock

\[ f(x) = \sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \]

\( x_0 = (-1.2, 1, ..., -1.2, 1) \)

\( x^* = (1, ..., 1) \)

\[ f(x^*) = 0 \]

TF.A8 Helical

\[ f(x) = 100[x_3 - 10\theta(x_1, x_2)]^2 + 100[(x_1^2 + x_2^2) - 1]^2 + x_3^2 \]

\[ \theta(x_1, x_2) = \begin{cases} 
(1/2\pi)\arctan(x_2/x_1), & \text{if } x_1 > 0 \\
(1/2\pi)\arctan(x_2/x_1) + 0.5, & \text{if } x_1 < 0 
\end{cases} \]

\( x_0 = (-1, 0, 0) \)

\( x^* = (1, 0, 0) \)

\[ f(x^*) = 0 \]
TF.A9 Hilbert

\[ f(x) = x^T A_n x, \text{ where } A_n(i, j) = \frac{1}{i+j-1} \text{ and } n \text{ is even.} \]

\[ x_0 = \begin{cases} 
(-4, -2), & \text{if } n = 2 \\
(-4, -2, -1.333, -1), & \text{if } n = 4 \\
(-4, -2, -1.333, -1, -0.8, -0.6667), & \text{if } n = 6.
\end{cases} \]

\[ x^* : \text{origin} \]

\[ f(x^*) = 0 \]

TF.A10 Mancino

\[
f(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} \frac{1}{5}(\sin^5 \log(x^2_j + i/j)^2 + \cos^5 \log(x^2_j + i/j)^2) + 14nx_i + (i - n/2)^3 \right)^2
\]

\[ x_0 = (c_{o1} f_1(0), \ldots, c_{on} f_n(0)), \text{ where } c_o = 7n/(18-36n-80n^2) \]

\[ f(x^*) = 0 \]

TF.A11 Miele

\[ f(x) = (x_1^4 - x_2^4)^4 + 100(x_2 - x_3)^6 + [\arctan(x_3 - x_4)]^4 + x_1^8 \]

\[ x_0 = (1, 2, 2, 2) \]

\[ x^* = (0, 1, 1, 1) \]

\[ f(x^*) = 0 \]
TF.A12 Power

\[ f(x) = \sum_{i=1}^{n} ix_i^2 \]

\( x_0 = (1, \ldots, 1) \)

\( x^* : \) origin

\( f(x^*) = 0 \)

TF.A13 Trigonometric

\[ f(x) = \sum_{i=1}^{n} \left( n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i \right)^2 \]

\( x_0 = (1/n, \ldots, 1/n) \)

\( f(x^*) = 0 \)

TF.A14 Wood

\[ f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3)^2 + (1 - x_3)^2 + 10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2 \]

\( x_0 = (-3, -1, -3, -1) \)

\( x^* = (1, 1, 1, 1) \)

\( f(x^*) = 0 \)

Group B

TF.B1 Nondia

\[ f(x) = \sum_{i=2}^{n} 100(x_i - x_i^2)^2 + (1 - x_i)^2 \]

\( x_0 = (-1, \ldots, -1) \)

\( x^* = (1, \ldots, 1) \)

\( f(x^*) = 0 \)
TF.B2 Penalty I

\[ f(x) = \sum_{i=1}^{n} 10^{-5} (x_i - 1)^2 + \left[ \sum_{j=1}^{n} x_j^2 \right] - 1/4 \]

\[ x_0 = 1, 2, ..., n \]

\[ f(x^*) = \begin{cases} 
2.24997 \times 10^{-5}, & \text{if } n = 4 \\
7.08765 \times 10^{-5}, & \text{if } n = 10
\end{cases} \]

TF.B3 Shanno's E–Rosen

\[ f(x) = \sum_{i=2}^{n} 100 (x_i - x_{i-1})^2 + (1 - x_i)^2 \]

\[ x_0 = (-1.2, 1, ..., 1) \]

TF.B4 Tridia

\[ f(x) = \sum_{i=1}^{n} i (2x_i - x_{i-1})^2 \]

\[ x_0 = (1, ..., 1) \]

TF.B5 Variably Dimensioned

\[ f(x) = \sum_{i=1}^{n} (x_i - 1)^2 + \left[ \sum_{j=1}^{n} j(x_j - 1) \right]^2 + \left[ \sum_{j=1}^{n} j(x_j - 1) \right]^4 \]

\[ x_0 = (1-1/n, ..., 1-1/n, ..., 0) \]

\[ x^* = (1, ..., 1) \]

\[ f(x^*) = 0 \]
**Group C**

**TF.C.1 Box three-dimensional**

\[
f(x) = \sum_{i=1}^{m} \left( e^{-t_i x_1} - e^{-t_i x_2} - x_3(e^{-t_i} - e^{-10t_i}) \right)^2
\]

\[t_i = 0.1i.
\]

\[x_0 = (0, 10, 20)
\]

\[x^* : (1, 10, 1), (10, 1, -1) \text{ and wherever } (x_1 = x_2 \text{ and } x_3 = 0)
\]

\[f(x^*) = 0
\]

**TF.C.2 Modified Cragg**

\[
f(x) = (e^{x_1} - x_2)^4 + 100(x_2 - x_3)^6 + [\arctan(x_3 - x_4)]^8 + x_1^8 + (x_4 - 1)^2
\]

\[x_0 = (1, 2, 2, 2)
\]

\[x^* = (0, 1, 1, 1)
\]

\[f(x^*) = 0
\]

**TF.C.3 Broyden Tridiagonal**

\[
f(x) = \sum_{i=1}^{n} \left( (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1 \right)^2, \text{ where } x_0 = x_{n+1} = 0.
\]

\[x_0 = (-1,...,-1)
\]

\[f(x^*) = 0
\]
TF.C.1 Bard

\[ f(x) = \sum_{i=1}^{m} \left( \beta_i - \frac{x_1 + i}{(16 - i)x_2 + \gamma_i x_3} \right)^2 \]

\[ m = 15 \]
\[ \gamma_i = \min(i, 16-i) \]

<table>
<thead>
<tr>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.14</td>
<td>4</td>
<td>0.25</td>
<td>7</td>
<td>0.35</td>
<td>10</td>
<td>0.58</td>
<td>13</td>
<td>1.34</td>
</tr>
<tr>
<td>2</td>
<td>0.18</td>
<td>5</td>
<td>0.29</td>
<td>8</td>
<td>0.39</td>
<td>11</td>
<td>0.73</td>
<td>14</td>
<td>2.10</td>
</tr>
<tr>
<td>3</td>
<td>0.22</td>
<td>6</td>
<td>0.32</td>
<td>9</td>
<td>0.37</td>
<td>12</td>
<td>0.96</td>
<td>15</td>
<td>4.39</td>
</tr>
</tbody>
</table>

\[ x_0 = (1, 1, 1) \]
\[ f(x^*) = 8.21487\ldots \cdot 10^{-3} \]

TF.C.2 Chebyquad

\[ f(x) = \sum_{i=1}^{m} \left( \frac{1}{n} \sum_{j=1}^{n} T_i(x_j) - \int_{0}^{1} T_i(x)dx \right)^2 \]

\[ T_i: \text{the } i-\text{th Cheby} \text{ } \text{shev polynomial shifted to the interval } [0, 1] \text{ and hence} \]

\[ \int_{0}^{1} T_i(x)dx = 0 \quad \text{for } i \text{ odd,} \]
\[ \int_{0}^{1} T_i(x)dx = -\frac{1}{(i^2 - 1)} \quad \text{for } i \text{ even.} \]

\[ x_0 = (1/(n+1), \ldots, j/(n+1), \ldots, n/(n+1)) \]
\[ f(x^*) = \begin{cases} 3.51687\ldots \cdot 10^{-3}, & \text{for } m = n = 8, \\ 6.50395\ldots \cdot 10^{-3}, & \text{for } m = n = 10. \end{cases} \]
TF.C.3 Kowalik and Osborne

\[ f(x) = \sum_{i=1}^{m} \left( \beta_i - \frac{x_i (\gamma_i^2 + \gamma_i x_{2i})}{\gamma_i^2 + \gamma_i x_3 + x_4} \right)^2, \] where \( m = 11 \) and

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \beta_i )</th>
<th>( \gamma_i )</th>
<th>( i )</th>
<th>( \beta_i )</th>
<th>( \gamma_i )</th>
<th>( i )</th>
<th>( \beta_i )</th>
<th>( \gamma_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1957</td>
<td>4.0000</td>
<td>5</td>
<td>0.0844</td>
<td>0.2500</td>
<td>9</td>
<td>0.0323</td>
<td>0.0833</td>
</tr>
<tr>
<td>2</td>
<td>0.1947</td>
<td>2.0000</td>
<td>6</td>
<td>0.0627</td>
<td>0.1670</td>
<td>10</td>
<td>0.0235</td>
<td>0.0714</td>
</tr>
<tr>
<td>3</td>
<td>0.1735</td>
<td>1.0000</td>
<td>7</td>
<td>0.0456</td>
<td>0.1250</td>
<td>11</td>
<td>0.0246</td>
<td>0.0625</td>
</tr>
<tr>
<td>4</td>
<td>0.1600</td>
<td>0.5000</td>
<td>8</td>
<td>0.0342</td>
<td>0.1000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( x_0 = (0.25, 0.39, 0.415, 0.39) \)

\( f(x^*) = \begin{cases} 
3.07505 \ldots \cdot 10^{-4}, & \text{at } (0.1928, 0.1913, 0.1231, 0.1361) \\
1.02734 \ldots \cdot 10^{-3}, & \text{at } (+\infty, -14.07 \ldots, -\infty, -\infty). 
\end{cases} \)
**TF.C.4 Madsen**

\[
f(x) = (x_1^2 + x_2^2 + x_1x_2)^2 + \sin^2 x_1 + \cos^2 x_2
\]

\[
x_0 = (3, 1)
\]

\[
f(x^*) = 0.616
\]

**TF.C.5 Osborne 1**

\[
f(x) = \sum_{i=1}^{m} \left( \beta_1 - (x_1 + x_2 e^{-t_4 x_1} + x_3 e^{-t_5 x_2}) \right)^2
\]

\[
m = 33
\]

\[
t_i = 10(i - 1)
\]

and

<table>
<thead>
<tr>
<th>i</th>
<th>( \beta_1 )</th>
<th>i</th>
<th>( \beta_1 )</th>
<th>i</th>
<th>( \beta_1 )</th>
<th>i</th>
<th>( \beta_1 )</th>
<th>i</th>
<th>( \beta_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.844</td>
<td>8</td>
<td>0.850</td>
<td>15</td>
<td>0.628</td>
<td>22</td>
<td>0.490</td>
<td>29</td>
<td>0.424</td>
</tr>
<tr>
<td>2</td>
<td>0.908</td>
<td>9</td>
<td>0.818</td>
<td>16</td>
<td>0.603</td>
<td>23</td>
<td>0.478</td>
<td>30</td>
<td>0.420</td>
</tr>
<tr>
<td>3</td>
<td>0.932</td>
<td>10</td>
<td>0.784</td>
<td>17</td>
<td>0.580</td>
<td>24</td>
<td>0.467</td>
<td>31</td>
<td>0.414</td>
</tr>
<tr>
<td>4</td>
<td>0.936</td>
<td>11</td>
<td>0.751</td>
<td>18</td>
<td>0.558</td>
<td>25</td>
<td>0.457</td>
<td>32</td>
<td>0.411</td>
</tr>
<tr>
<td>5</td>
<td>0.925</td>
<td>12</td>
<td>0.718</td>
<td>19</td>
<td>0.538</td>
<td>26</td>
<td>0.448</td>
<td>33</td>
<td>0.406</td>
</tr>
<tr>
<td>6</td>
<td>0.908</td>
<td>13</td>
<td>0.685</td>
<td>20</td>
<td>0.522</td>
<td>27</td>
<td>0.438</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.881</td>
<td>14</td>
<td>0.658</td>
<td>21</td>
<td>0.506</td>
<td>28</td>
<td>0.431</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
x_0 = (0.5, 1.5, -1, 0.01, 0.02)
\]

\[
x^* = (0.3754, 1.936, -1.465, 0.01287, 0.02212)
\]

\[
f(x^*) = 5.46489\ldots \cdot 10^{-5}
\]
TF.C 6 Osborne 2

\[ f(x) = \sum_{i=1}^{m} \left( \beta_i - \left( x_1 e^{-t_i x_5} + x_2 e^{-(t_i - x_9)^2 x_6} + x_3 e^{-(t_i - x_{10})^2 x_7} + x_4 e^{-(t_i - x_{11})^2 x_8} \right)^2 \right), \]

\[ m = 65 \]

\[ t_i = (i - 1)/10 \]

and

<table>
<thead>
<tr>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
<th>i</th>
<th>( \beta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.366</td>
<td>14</td>
<td>0.655</td>
<td>27</td>
<td>0.612</td>
<td>40</td>
<td>0.429</td>
<td>53</td>
<td>0.597</td>
</tr>
<tr>
<td>2</td>
<td>1.191</td>
<td>15</td>
<td>0.616</td>
<td>28</td>
<td>0.558</td>
<td>41</td>
<td>0.523</td>
<td>54</td>
<td>0.625</td>
</tr>
<tr>
<td>3</td>
<td>1.112</td>
<td>16</td>
<td>0.606</td>
<td>29</td>
<td>0.533</td>
<td>42</td>
<td>0.562</td>
<td>55</td>
<td>0.739</td>
</tr>
<tr>
<td>4</td>
<td>1.013</td>
<td>17</td>
<td>0.602</td>
<td>30</td>
<td>0.495</td>
<td>43</td>
<td>0.607</td>
<td>56</td>
<td>0.710</td>
</tr>
<tr>
<td>5</td>
<td>0.991</td>
<td>18</td>
<td>0.626</td>
<td>31</td>
<td>0.500</td>
<td>44</td>
<td>0.653</td>
<td>57</td>
<td>0.729</td>
</tr>
<tr>
<td>6</td>
<td>0.885</td>
<td>19</td>
<td>0.651</td>
<td>32</td>
<td>0.423</td>
<td>45</td>
<td>0.672</td>
<td>58</td>
<td>0.720</td>
</tr>
<tr>
<td>7</td>
<td>0.831</td>
<td>20</td>
<td>0.724</td>
<td>33</td>
<td>0.395</td>
<td>46</td>
<td>0.708</td>
<td>59</td>
<td>0.636</td>
</tr>
<tr>
<td>8</td>
<td>0.847</td>
<td>21</td>
<td>0.649</td>
<td>34</td>
<td>0.375</td>
<td>47</td>
<td>0.633</td>
<td>60</td>
<td>0.581</td>
</tr>
<tr>
<td>9</td>
<td>0.786</td>
<td>22</td>
<td>0.649</td>
<td>35</td>
<td>0.372</td>
<td>48</td>
<td>0.668</td>
<td>61</td>
<td>0.428</td>
</tr>
<tr>
<td>10</td>
<td>0.725</td>
<td>23</td>
<td>0.694</td>
<td>36</td>
<td>0.391</td>
<td>49</td>
<td>0.645</td>
<td>62</td>
<td>0.292</td>
</tr>
<tr>
<td>11</td>
<td>0.746</td>
<td>24</td>
<td>0.644</td>
<td>37</td>
<td>0.396</td>
<td>50</td>
<td>0.632</td>
<td>63</td>
<td>0.162</td>
</tr>
<tr>
<td>12</td>
<td>0.679</td>
<td>25</td>
<td>0.624</td>
<td>38</td>
<td>0.405</td>
<td>51</td>
<td>0.591</td>
<td>64</td>
<td>0.098</td>
</tr>
<tr>
<td>13</td>
<td>0.608</td>
<td>26</td>
<td>0.661</td>
<td>39</td>
<td>0.428</td>
<td>52</td>
<td>0.559</td>
<td>65</td>
<td>0.054</td>
</tr>
</tbody>
</table>

\[ x_0 = (1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5) \]

\[ x^* = (1.3099, 0.4315, 0.6336, 0.5994, 0.7542, 0.9042, 1.3658, 4.8237, 2.3987, 4.5689, 5.6753) \]

\[ f(x^*) = 4.01377 \ldots 10^{-2} \]
\[ f(x) = x_1^2 + (x_2 - x_1^2 - 1)^2 + \sum_{i=1}^{m} \left( \sum_{j=2}^{n} \left( \sum_{i=1}^{m} (j - 1)x_i^j \right)^2 - \left( \sum_{j=1}^{n} x_i^{j-1} \right)^2 - 1 \right)^2 \]

\[ \begin{align*}
\text{m} = 29 \quad \text{and} \quad 2 \leq n \leq 31 \\
t_i = i/29 \\
x_0 = (0, \ldots, 0) \\
x^* = \begin{cases} 
(0.01572, 1.012, -0.2329, -1.26, -1.514, 0.993), & \text{if } n = 6 \\
(0.000015, 0.1, -0.0147, -0.146, 1, -2.62, 4.1, -2.14, 1.05) & \text{if } n = 9
\end{cases}
\end{align*} \]

\[ f(x^*) = \begin{cases} 
2.28767 \ldots \cdot 10^{-3}, & \text{if } n = 6 \\
1.39976 \ldots \cdot 10^{-6} & \text{if } n = 9 \\
4.72238 \ldots \cdot 10^{-10} & \text{if } n = 12.
\end{cases} \]

\[ f(x) = \sum_{i=1}^{m} \left( (x_1 + t_i x_2 - e^{t_i})^2 + (x_3 + x_4 \sin(t_i) - \cos(t_i))^2 \right)^2, \text{ where } t_i = i/5. \]

\[ x_0 = (25, 5, -5, -1) \]

\[ f(x^*) = 85822.2\ldots, \text{ if } m = 20 \]

\[ f(x) = (x_2^3 - 5x_2^2 + 2x_2 - x_1 + 13)^2 + (x_2^3 + x_2^2 - 14x_2 + x_1 - 29)^2 \]

\[ x_0 = (0.5, -2) \]

\[ f(x^*) = \begin{cases} 
0, & \text{at } (5, 4) \\
48.9842\ldots, & \text{at } (11.41\ldots, -0.8968\ldots)
\end{cases} \]
Jennrich and Sampson

\[ f(x) = \sum_{i=1}^{m} \left( 2 + 2i - (e^{ix_1} + e^{ix_2}) \right)^2 \]

\[ x_0 = (0.3, 0.4) \]

\[ f(x^*) = 124.362... \text{ at } x_1 = x_2 = 0.2578... \text{ for } m = 10. \]

where TF.A1, TF.A3, TF.A4, TF.A6, TF.A7, TF.A8, TF.A13, TF.A14, TF.B2, TF.B5, C.1, C.3, TF.C.1, TF.C.2, TF.C.3, TF.C.5, TF.C.6, TF.C.7, TF.C.1, TF.C.2 and TF.C.3 appear in More, Carbow and Hillstrom (1981); TF.A2, TF.A5 and TF.A11 appear in Wolfe (1976); TF.A9 appears in Schittkowski (1987); TF.A10 appears in Oren (1974); TF.A12 appears in Spedicato (1975); TF.B1 and TF.B4 appear in Shanno (1978a); TF.B3 appears in Shanno (1978b); TF.C.2 appears in Gill and Maturity (1976); TF.C.4 appears in Madson (1975).

9.2 Some Popular Algorithms

To look at the efficiency of the designed algorithms from all sides, some numerical results obtained by using other popular algorithms are used to make comparisons under the same termination criterion. They are listed as follows.

* BFGS.
* SBFGS: BFGS with Shanno–Phua's initial scaling.
* BFGS18: Shanno–Phua's code.
* OCON: Davidon's optimal condition algorithm.
* OCON10: Davidon's optimal condition algorithm with Shanno's initial scaling.
* OSS2: Oren-Spedicato's algorithm.
* SCONBP (Shanno): Scaling CG algorithm by using Beale's restart direction with Powell's criterion.
* PRCON: Polark-Ribiere conjugate gradient algorithm with \( d_k = - \frac{\|s_{k-1}\|}{\|g_k\|}g_k \)
restart direction every n iterations.

* SQN: A QN algorithm with special BFGS update.

* PCG: A combined CG-QN algorithm, in which special BFGS update is used as the preconditional matrix.

* H-Xu: A hybrid algorithm using Fletcher-Xu's switch with the OCSSR1 update.

9.3 Statements on Numerical tests

With every test problem the computation is carried out in double precision on an IBM PC/AT clone. The corresponding machine precision is of the order of \(10^{-16}\).

An iterate \(x_k\) is accepted as a close approximation to a stationary point if the following criteria are satisfied:

\[
\|g_k\| < \varepsilon_c \quad (9.3.1)
\]

or

\[
\max_i |g_k(i)| < \varepsilon_s. \quad (9.3.2)
\]

In the line search, Rule (1.2.4) with \(\tau = 10^{-4}\) and \(\beta = 0.9\) is used as a termination criterion of calculating steplength, and the initial value of \(\alpha_k\) is always taken as 1.

For each test function, the Tables report the dimension of the objective function argument (n), the number of iteration (\(N_t\)), the number of function evaluations (\(N_f\)) and the number of gradient evaluations (\(N_g\)). F designates failure to converge.
References


[77] Spedicato, E. (1985), "Computational Experience with Rank-One Positive

pp.15-25.


Updates Form the Preconvex Part of Broyden's Family", IMA J. Numer. Anal.,
Vol.8, pp.487-509.
## ERRATA

<table>
<thead>
<tr>
<th>Page</th>
<th>Line</th>
<th>Old Word</th>
<th>New Word</th>
<th>Page</th>
<th>Line</th>
<th>Old Word</th>
<th>New Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>ii</td>
<td>4</td>
<td>unless</td>
<td>unless</td>
<td>20</td>
<td>20</td>
<td>assumptions</td>
<td>assumptions</td>
</tr>
<tr>
<td>8</td>
<td>nversity</td>
<td>University</td>
<td>positive</td>
<td>11</td>
<td>nversity</td>
<td>University</td>
<td>parameter</td>
</tr>
<tr>
<td>14</td>
<td>nversity</td>
<td>University</td>
<td>sequence</td>
<td>17</td>
<td>nversity</td>
<td>University</td>
<td>corresponding</td>
</tr>
<tr>
<td>v</td>
<td>6</td>
<td>chosen</td>
<td>maximum</td>
<td>23</td>
<td>chapters</td>
<td>chapters</td>
<td>Taylor</td>
</tr>
<tr>
<td>15</td>
<td>chapters</td>
<td>contains</td>
<td>classical</td>
<td>19</td>
<td>contains</td>
<td>contains</td>
<td>classical</td>
</tr>
<tr>
<td>20</td>
<td>chapters</td>
<td>chapters</td>
<td>attractive</td>
<td>v</td>
<td>chosen</td>
<td>maximum</td>
<td>maximum</td>
</tr>
<tr>
<td>vii</td>
<td>20</td>
<td>Chioce</td>
<td>Choice</td>
<td>23</td>
<td>conditons</td>
<td>conditons</td>
<td>conditions</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>sequence</td>
<td>sequence</td>
<td>33</td>
<td>17</td>
<td>Hassian</td>
<td>Hessian</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>chosen</td>
<td>maximum</td>
<td>35</td>
<td>7</td>
<td>necessarily</td>
<td>necessarily</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>satisfies</td>
<td>satisfies</td>
<td>36</td>
<td>11</td>
<td>direction</td>
<td>direction</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>sequence</td>
<td>sequence</td>
<td>37</td>
<td>9</td>
<td>combination</td>
<td>combination</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>sequence</td>
<td>sequence</td>
<td>40</td>
<td>11</td>
<td>convergent</td>
<td>convergent</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>Lipschits</td>
<td>Lipschitz</td>
<td>20</td>
<td>13</td>
<td>distinguished</td>
<td>distinguished</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>minimum</td>
<td>minimum</td>
<td>22</td>
<td>princlple</td>
<td>princlple</td>
<td>principle</td>
</tr>
<tr>
<td>15</td>
<td>towards</td>
<td>towards</td>
<td>14</td>
<td>14</td>
<td>distinguished</td>
<td>distinguished</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>summetric</td>
<td>symmetric</td>
<td>17</td>
<td>princlple</td>
<td>princlple</td>
<td>principle</td>
</tr>
<tr>
<td>20</td>
<td>17</td>
<td>satisfying</td>
<td>satisfying</td>
<td>42</td>
<td>1</td>
<td>simplie</td>
<td>simple</td>
</tr>
<tr>
<td>Page</td>
<td>Line</td>
<td>Old Word</td>
<td>New Word</td>
<td>Page</td>
<td>Line</td>
<td>Old Word</td>
<td>New Word</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------------</td>
<td>------------</td>
<td>------</td>
<td>------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>45</td>
<td>16</td>
<td>attractive</td>
<td>attractive</td>
<td>83</td>
<td>11</td>
<td>multiplier</td>
<td>multiplier</td>
</tr>
<tr>
<td>46</td>
<td>6</td>
<td>attractive</td>
<td>attractive</td>
<td>86</td>
<td>15</td>
<td>search</td>
<td>search</td>
</tr>
<tr>
<td>47</td>
<td>6</td>
<td>sequence</td>
<td>sequence</td>
<td>91</td>
<td>16</td>
<td>Assume</td>
<td>Assume</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>condition</td>
<td>condition</td>
<td>105</td>
<td>1</td>
<td>alternative</td>
<td>alternative</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>sequence</td>
<td>sequence</td>
<td>10</td>
<td>7</td>
<td>satisfying</td>
<td>satisfying</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>sequence</td>
<td>sequence</td>
<td>106</td>
<td>11</td>
<td>distinguishing</td>
<td>distinguishing</td>
</tr>
<tr>
<td>49</td>
<td>11</td>
<td>symmetric</td>
<td>symmetric</td>
<td>125</td>
<td>7</td>
<td>Statements</td>
<td>Statements</td>
</tr>
<tr>
<td>51</td>
<td>6</td>
<td>length</td>
<td>length</td>
<td>52</td>
<td>8</td>
<td>Applying</td>
<td>Applying</td>
</tr>
<tr>
<td>52</td>
<td>16</td>
<td>length</td>
<td>length</td>
<td>57</td>
<td>9</td>
<td>continuously</td>
<td>continuously</td>
</tr>
<tr>
<td>62</td>
<td>11</td>
<td>parameters</td>
<td>parameters</td>
<td>64</td>
<td>11</td>
<td>addition</td>
<td>addition</td>
</tr>
<tr>
<td>65</td>
<td>5</td>
<td>caculate</td>
<td>calculate</td>
<td>6</td>
<td>5</td>
<td>recurrence</td>
<td>recurrence</td>
</tr>
<tr>
<td>75</td>
<td>3</td>
<td>whereby</td>
<td>whereby</td>
<td>8</td>
<td>9</td>
<td>multiple</td>
<td>multiple</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>Substituting</td>
<td>Substituting</td>
<td>12</td>
<td>9</td>
<td>attractive</td>
<td>attractive</td>
</tr>
<tr>
<td>77</td>
<td>13</td>
<td>Chioce</td>
<td>Choice</td>
<td>78</td>
<td>17</td>
<td>columns</td>
<td>columns</td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>unsatisfact</td>
<td>unsatisfactory</td>
<td>4</td>
<td>4</td>
<td>ideally</td>
<td>ideally</td>
</tr>
</tbody>
</table>