1988

A new modified Newton's method for minimizing factorable functions.

Leung Chow. Kwok
University of Windsor

A New Modified Newton's Method for Minimizing Factorable Functions

by

Kwok Leung Chow

A Thesis
submitted to the
Faculty of Graduate Studies and Research
through the Department of
Mathematics and Statistics in Partial Fulfillment
of the requirements for the Degree
of Master of Science at
the University of Windsor

Windsor, Ontario, Canada

1988

ISBN 0-315-48136-6

Abstract

Most of the objective functions that arise in nonlinear programming are complicated compositions of transformed sums and products of a single variable, that is, they are factorable. If a factorable function is twice differentiable, then its Hessian matrix can be written in dyadic form. That is, it can be expressed as a sum of rank one matrices, i.e. outer products or dyads, and pairs of rank one matrices.

This thesis presents a new modification of Newton's method for minimizing factorable functions which exploits the fact that most of the dyads appear in pairs. This results in an improvement over the method given by Sisser [1982] in that it avoids the unnecessary transformations required at each iteration.

In addition, the modification presented in this thesis does not require an ordering of the dyads at each iteration, nor does it require the calculation of an expensive perturbation parameter in iterations having an indefinite Hessian matrix.

Contents

List of Tables

Chapter 1

Introduction

1.1 Introduction

The purpose of this thesis is to present a new modification of Newton's method for the solution of the unconstrained minimization problem. The unconstrained minimization problem is one of determining the values of several variables that will minimize a function of those variables. A well known technique for solving this problem is Newton's method. However, Newton's method involves the calculation and inversion of the Hessian, i.e. second derivative, matrix. This drawback has led to the development of Quasi-Newton, or Variable metric, methods. These methods use gradient, i.e. first derivative, information to approximate the Hessian matrix. Recently, the concept of a factorable function has been used to overcome the difficulties associated with calculating and inverting the Hessian matrix. This has given rise to Sisser's modified Newton method, which is competitive with the Quasi-Newton methods. Our new modification of Newton's method, which can be thought of as an extension of Sisser's method, takes further advantage of factorable function to overcome difficulties associated with the Hessian.

1.2 Survey

The unconstrained minimization problem is to obtain a set of values for several variables which minimize a function of those variables. Practical applications of this problem include nonlinear regression [Dennis and Schnabel, 1983], engineering design [Fratta et al., 1973], and mathematical programming [Fiacco and McCormick, 1968].

The nonlinear regression problem is important to fields such as physics, chemistry, and medicine, etc. Dennis and Schnabel [1983, p.6] describe a problem arising in physics that involves fitting a bell shaped curve to twenty pieces of solar spectroscopy data.

In Fratta et al. [1973, p.97], a nonlinear, unconstrained, multicommodity flow problem is described. This problem has application to the design of the ARPA computer network.

The mathematical programming problem of minimizing a nonlinear function subject to nonlinear equality constraints also has many practical applications. Fiacco and McCormick [1968] show how this problem can be solved by solving a sequence of unconstrained minimization problems.

Due to the practical applications of the unconstrained minimization problem, a considerable amount of effort has been expended on the development of efficient algorithms for their solution. One of the best known methods is Newton's method. It has the advantage that if it converges, it does so at a superlinear, or even quadratic, rate. Also, since Newton's method requires the calculation of the Hessian matrix, a further benefit is that the sufficient conditions for optimality can be verified.

Although Newton's method is theoretically attractive, it has at least three drawbacks. The first is that to guarantee convergence, the initial estimate of the solution must be "close" to the actual solution. The second is that if the Hessian matrix is not positive definite, the Newton iteration may be undefined. The third drawback is that the calculation and inversion of the Hessian can be computationally expensive. These drawbacks have led to various modification of Newton's method.

In order to guarantee convergence for any initial estimate, line search techniques can be used at each iteration to determine a suitable step in the Newton direction (see [Dennis and Schnabel, 1983] for a good discussion of currently used line search techniques). If the Hessian is not positive definite, it can be approximated by a matrix which is positive definite. A popular strategy [Gill et al., 1981] is to add to the Hessian a sufficiently large positive multiple of the identity matrix. In order to avoid calculations involving the Hessian matrices, Quasi-Newton method [Dennis and Morè, 1977] have been developed. Essentially, these methods use gradient information to build an approximation to the inverse Hessian at each point.

However, avoidance of the Hessian has resulted in certain limitations to the minimization process. The methods using actual Hessians are more reliable and usually require significantly fewer iterations and function evaluations. Also, availability of the Hessian allows an easy check to whether a possible solution satisfies the necessary and sufficient conditions for a local minimum.

Recently, the concept of factorable functions [McCormick, 1983] has led to a modification of Newton's method [Sisser, 1982] which overcomes some of the difficulty in calculating the inverse Hessian matrix. McCormick observed that most of the objective functions that arise in nonlinear programming are complicated compositions of transformed sums and products of functions of a single variable. He then formalized a class of functions, called factorable functions, based on his observation. He showed that if a factorable function is at least twice differentiable, its Hessian can be expressed in a special dyadic (outer product) form. This special form provides an easy way to manipulate the actual Hessian. Also, the evaluation of the gradient vectors and Hessian matrices require little additional calculation or storage once the function itself has been evaluated.

In Jackson and McCormick [1986], the authors state that, since the discovery of factorable functions, the theory of factorable programming has been further developed and refined in a variety of ways. In particular, they mention that routines from FACSUMT [Mylander et al., 1971; Pugh, 1972; Ghaemi and McCormick, 1979] have been separated out into the stand alone package FACTIN. The FACTIN package can be used with any nonlinear programming system to automatically provide the values of the function, gradient, and Hessian at particular a point. The basic requirement for FACTIN is that the user write the problem in factorable form.

Jackson and McCormick state further that FACTIN is just an input processor. It contains no code that exploits the special structure of factorable function, i.e. the dyadic form of Hessian.

A current version of the factorable programming system is called FACTPROG. It will require that a user be able to present the objective function in a natural mathematical language. Then, it will translate the input into factorable form and process the input to produce the dyadic form of the Hessian.

Sisser [1982] has taken the advantage of the dyadic form of the Hessian to develop a modified Newton's method for minimizing factorable functions. An iteration of Sisser's method proceeds as follows. The Hessian matrix, which is naturally given as a diagonal matrix plus the sum of dyad pairs, is converted by a transformation into a diagonal matrix plus an equivalent sum of symmetric dyads. If the diagonal matrix is not positive definite, the negative and zero elements are replaced by a value of one. A new symmetric dyad is added for each diagonal element that is changed. Then, the string of symmetric dyads is arranged in non-increasing order according to their nonzero eigenvalues.

The inverse Hessian is obtained by using the SMW updated formula [Sherman and Morrison, 1949; Woodbury, 1950] to add the symmetric dyads, once at a time, to the diagonal matrix. This update formula provides a test which can be used before a dyad is added to determine whether the resulting matrix will still be positive definite. When adding a dyad causes the loss of positive definiteness, a lower bound on the smallest eigenvalue of the Hessian will be computed in order to construct a positive definite approximation to the Hessian.

Computational results with Sisser's method show that the factorable function approach is competitive with the other minimization algorithms, for example, the Quasi-Newton methods. The number of iterations and function evaluations required by Sisser's method is much less than the number required by the well known BFGS method. However, the computational effort required at each iteration of Sisser's method is considerably larger. This is due to the transformations, the ordering, and the indefinite Hessian strategy. Although Sisser's method requires expensive iterations, it does provide reliability in that it solved hard problems which other methods could not solve. These favorable results indicate the advantages, and capabilities, of the factorable function approach to unconstrained minimization problem. Thus, the factorable function approach should be further investigated.

This thesis presents a new modification of Newton's method for minimizing factorable functions. The new modification exploits the fact that most of the dyads appear naturally in pairs. This results in an improvement over Sisser's method in that transformations required at each iteration are avoided. In addition, the modification does not require an ordering of dyads, nor does it require the expensive calculation of a perturbation parameter when the positive definite property of the Hessian is destroyed.

1.3 Mathematical background

In this thesis we consider the unconstrained minimization problem given by

$$min\{f(x) | x \in \mathbf{R}^n\}, \tag{1.1}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is at least twice continuously differentiable. In particular, we wish to determine a strong local minimizer for $f(x)$.

Definition 1.3.1 *The point $\check{x}$ is a strong local minimizer for (1.1) if there exists an $\epsilon > 0$ such that $f(\check{x}) < f(x)$ for all $x \in \{x \in \mathbf{R}^n \mid \quad \| x - \check{x} \| < \epsilon, x \neq \check{x}\}$.*

The first-order necessary condition for $\check{x}$ to be a strong local minimizer, or simply, a minimizer, for (1.1) is given in Lemma (1.3.1), while the second-order sufficient condition is given in Lemma (1.3.2). First, we define the gradient of f at x by

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right)^T,$$

where the superscript "T" denotes matrix and vector transposition, and the Hessian matrix of f at x by

$$\mathcal{H}_f(x) = \nabla^2 f(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right],$$

where $1 \leq i, j \leq n$.

Lemma 1.3.1 *If the point $\breve{x} \in \mathbf{R}^n$ is a strong local minimizer for (1.1), then $\nabla f(\breve{x}) = 0$.*

Lemma 1.3.2 *If $\breve{x} \in \mathbf{R}^n$ is such that $\nabla f(\breve{x}) = 0$ and if $\mathcal{H}_f(\breve{x})$ is positive definite, then $\breve{x}$ is a strong local minimizer for (1.1).*

The algorithms for the solution of (1.1) considered herein attempt to find a point $\breve{x}$ such that $\nabla f(\breve{x}) = 0$ and $\mathcal{H}_f(\breve{x})$ is positive definite. They are all descent methods in that they produce a sequence of iterates $x_{k+1} = x_k + \rho_k d_k$, where x_0 is some initial estimate of the minimizer $\breve{x}$, d_k is a descent direction as defined below, and ρ_k is a step size, such that $f(x_{k+1}) < f(x_k)$.

Definition 1.3.2 *A vector d is said to be a descent direction for $f(x)$ at the point x if $\nabla f(x)^T d < 0$.*

The existence of a step size $\rho_k > 0$ is guaranteed by the following lemma.

Lemma 1.3.3 *If d is a descent direction for $f(x)$ at x, then there exists a $\breve{\rho} > 0$ such that*

$$f(x + \rho d) < f(x)$$

for $0 < \rho < \breve{\rho}$.

The descent methods to be considered have the form of the following model algorithm.

Model algorithm

Let x_0 be given as an initial estimate of the minimizer $\breve{x}$ and let $\tilde{H}_0$ be a positive definite matrix. Set $k = 0$.

1. **Test for convergence:** If $\|\nabla f(x_k)\| = 0$, stop with $\bar{x} = x_k$. Otherwise, goto 2.

2. **Determine a search direction:** Set $d_k = -\tilde{H}_k^{-1}\nabla f(x_k)$.

3. **Determine a optimal step size:** Choose ρ_k such that $f(x_k + \rho_k d_k) = min\{f(x_k + \rho d_k)| \, \rho > 0\}$.

4. **Update:** Set $x_{k+1} \leftarrow x_k + \rho_k d_k$, determine a positive definite matrix $\tilde{H}_{k+1}$, and $k \leftarrow k + 1$. Goto Step 1

Some remarks are in order. In practice, the convergence criteria is replaced with $\|\nabla f(x_k)\| \leq \epsilon$ where, for example, $\epsilon = 10^{-6}$. The direction d_k is a descent direction since $\tilde{H}_k$, and thus $\tilde{H}_k^{-1}$, is positive definite. There are several difficulties associated with determining the step size ρ_k. For instance, the solution of the minimization problem, $min\{f(x_k + \rho d_k)| \, \rho > 0\}$, may not exist. Even if the solution does exist, there is no guarantee that the best ρ can be obtained. A reasonable solution to the step size problem can be found in Dennis and Schnabel [1983]. Their method is summarized in the following Step 3'.

3' **Determine a suitable step size:** Set $j = 1$, $\rho_j = 1$, and $\alpha = 10^{-4}$.

(a) If $f(x_k + \rho_j d_k) \leq f(x_k) + \alpha \rho_j \nabla f(x_k)^T d_k$, then accept ρ_j and set $\rho_k = \rho_j$.

(b) If $j > 1$ then goto Step (c) else set $j = j+1$, $\rho_j = [-\nabla f(x_k)^T d_k]/[2(f(x_k + d_k) - f(x_k) - \nabla f(x_k)^T d_k)]$, goto Step (a).

(c) Set $j = j + 1$, $\rho_j = [-b + \sqrt{b^2 - 3a\nabla f(x_k)^T d_k}]/[3a]$ where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{(\rho_{j-1} - \rho_{j-2})} \times \begin{bmatrix} \frac{1}{(\rho_{j-1})^2} & -\frac{1}{(\rho_{j-2})^2} \\ -\frac{\rho_{j-2}}{(\rho_{j-1})^2} & \frac{\rho_{j-1}}{(\rho_{j-2})^2} \end{bmatrix} \times \begin{bmatrix} f(x_k + \rho_{j-1} d_k) - f(x_k) - \rho_{j-1}\nabla f(x_k)^T d_k \\ f(x_k + \rho_{j-2} d_k) - f(x_k) - \rho_{j-2}\nabla f(x_k)^T d_k \end{bmatrix},$$

Goto Step (a).

We note that we replace Step 3 with Step 3' in our implementation of the algorithm.

Finally, for ease of presentation, we denote the function, gradient, and Hessian of $f(x)$, evaluated at some iterate x_k, by

$$\begin{aligned} f_k &= f(x_k), \\ g_k &= \nabla f(x_k), \quad \textit{and} \\ \mathcal{H}_k &= \mathcal{H}_f(x_k), \end{aligned}$$

respectively.

Chapter 2

Newton's method

The concept behind Newton's method is to approximate the function being minimized by a quadratic function and to minimize the quadratic exactly. If the gradient and the Hessian of the objective function are available, a quadratic model of the function can be obtained by taking the first three terms of the Taylor's series expansion about the current iterate, say x_k. That is,

$$f(x_k + d) \simeq f_k + g_k{}^T d + \frac{1}{2} d^T \mathcal{H}_k d. \tag{2.1}$$

It is helpful to formulate the quadratic model in terms of d rather than the predicted minimizer itself. The minimum of the right hand side of Equation (2.1) will be achieved if d_k is a minimizer of the quadratic function

$$F(d) = g_k{}^T d + \frac{1}{2} d^T \mathcal{H}_k d. \tag{2.2}$$

A stationary point, d_k, satisfies the first-order necessary condition

$$\nabla F(d_k) = 0, \tag{2.3}$$

which are equivalent to the linear system

$$\mathcal{H}_k d_k = -g_k. \tag{2.4}$$

Thus, in the model algorithm $\tilde{H}_k = \mathcal{H}_k, \rho_k = 1$, and

$$x_{k+1} = x_k + d_k, \quad k = 0, 1, \ldots. \tag{2.5}$$

When Newton's method works, the rate of convergence of the sequence of points, $\{x_k\}$ to $\check{x}$, is at least superlinear. Furthermore, Newton's method, when started close enough to a point whose gradient vanishes and whose Hessian satisfies a Lipschitz condition, is guaranteed to converge with a rate of convergence that is at least quadratic.

Theorem 2.0.1 *Suppose that the sequence $\{x_k\}$ generated by Equation (2.5) converges to $\check{x} \in \mathbf{R}^n$, where $\nabla f(\check{x}) = 0$ and $\mathcal{H}_f(\check{x})$ exists. Then, Newton's method converge at superlinear rate. If there exists ϵ, $\sigma > 0$ such that $\| \mathcal{H}_f(\check{x}) - \mathcal{H}_f(y) \| \leq \sigma \| \check{x} - y \|$ for all $y \in \{x \in \mathbf{R}^n \mid \ \| \check{x} - y \| < \epsilon\}$, then the rate of convergence is quadratic.*

Proof: see [Dennis and Schnabel, 1983] or [Ortega and Rheinboldt, 1970]. **QED**

In practice, Newton's method may fail because either $\mathcal{H}_k$ is not positive definite or because $\rho_k = 1$ is not an acceptable step size. To overcome the first difficulty, one popular strategy is to approximate $\mathcal{H}_k$ by some positive definite matrix $\tilde{H}_k$. The search direction is then given by the solution of

$$\tilde{H}_k d_k = -g_k. \tag{2.6}$$

The use of the step 3' in the model algorithm to choose ρ_k overcomes the second problem. Note that step 3' first attempts $\rho_k = 1$, and modifies ρ_k only if it is necessary to do so.

However, even if the above problems do not arise, Newton's method requires the construction of $\mathcal{H}_k$ and, subsequently, the solution of Equation (2.6). These computationally expensive operations are the main reasons that Quasi-Newton methods have become so popular. In the remainder of thesis we show how the theory of factorable functions can be used to derive an implementation of Newton's method that is competitive with the Quasi-Newton methods.

Chapter 3

Factorable functions

3.1 Factorable functions

The modifications of Newton's method to be discussed in Chapters four and five of this thesis are based on the concept of factorable functions introduced by McCormick [1968]. McCormick makes the observation that most functions of several variables which occur in nonlinear optimization are complicated compositions of transformed sums and products of functions of a single variable. (McCormick points out that exceptions to the class are, for example, functions that are given implicitly rather then explicitly.) That is, the natural way of looking at a multivariable function is the way we look at the function when we would evaluate it for a particular set of values. Consider the following example.

Example 3.1.1 *Consider at Sisser's function given by:*

$$f(x_1, x_2) = 3x_1^4 - 2x_1^2x_2^2 + 3x_2^4$$

Suppose (x_1, x_2) is given. We summarize the way used to evaluate this function at this particular point in Table 3.1

The procedure for putting the function in the form of a factorable function parallels that required to evaluate the function, as observed in Table 3.1. Based on

Evaluation Step	Portion of Original Function	Note on Evaluation
—	x_1	Given $f_1 = x_1$
—	x_2	Given $f_2 = x_2$
1	${x_1}^4$	Calculate $f_3 = f_1^4$
2	$3{x_1}^4$	Calculate $f_4 = 3f_3$
3	${x_1}^2$	Calculate $f_5 = f_1^2$
4	${x_2}^2$	Calculate $f_6 = f_2^2$
5	${x_1}^2{x_2}^2$	Calculate $f_7 = f_5 \cdot f_6$
6	$2{x_1}^2{x_2}^2$	Calculate $f_8 = 2f_7$
7	$3{x_1}^4 - 2x_1{x_2}^2$	Calculate $f_9 = f_4 - f_8$
8	${x_2}^4$	Calculate $f_{10} = f_2^4$
9	$3{x_2}^4$	Calculate $f_{11} = 3f_{10}$
10	$3{x_1}^4 - 2{x_1}^2{x_2}^2 + 3{x_2}^4$	Calculate $f_{12} = f_9 + f_{11}$

Table 3.1: Evaluation of Sisser's function

this idea, we can formalize the definition of the factorable function in the following way.

Definition 3.1.1 *Consider the function $f = f(x_1, \ldots, x_n)$. Let's define the following rules.*

1 f_i is defined to be the i th Euclidean coordinate for $i = 1, \ldots, n$. For instance, we have $f_i = x_i$.

2 f_i is formed using one of the following compositions where $\sigma(i), \breve{\sigma}(i) < i$ and T_i is a function of a single variable for $i = n+1, \ldots, N$.

(a) $f_i = f_{\sigma(i)} + f_{\breve{\sigma}(i)}$

(b) $f_i = f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}$

(c) $f_i = T_i\left[f_{\sigma(i)}\right]$

A function f is said to be factorable if it can be formed according to rule 1 and 2. Thus, $f = f_N$ is defined as a factorable function provided f is factorable. The resulting sequence of functions $\{f_1, \ldots, f_N\}$ is called a factored sequence for function f and we refer to $f_1, \ldots, f_N$ as the factored sequence functions (FSF).

Rule	f_i	∇f_i
1	x_i	e_i
$2a$	$f_{\sigma(i)} + f_{\breve{\sigma}(i)}$	$\nabla f_{\sigma(i)} + \nabla f_{\breve{\sigma}(i)}$
$2b$	$f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}$	$f_{\sigma(i)} \cdot \nabla f_{\breve{\sigma}(i)} + f_{\breve{\sigma}(i)} \cdot \nabla f_{\sigma(i)}$
$2c$	$T_i[f_{\sigma(i)}]$	$\dot{T}_i[f_{\sigma(i)}] \cdot \nabla f_{\sigma(i)}$

Table 3.2: Monadic gradients of the factored sequence function

Rule	f_i	$\nabla^2 f_i$
1	x_i	$0_{n\times n}$
$2a$	$f_{\sigma(i)} + f_{\breve{\sigma}(i)}$	$\nabla^2 f_{\sigma(i)} + \nabla^2 f_{\breve{\sigma}(i)}$
$2b$	$f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}$	$f_{\sigma(i)} \cdot \nabla^2 f_{\breve{\sigma}(i)} + f_{\breve{\sigma}(i)} \cdot \nabla^2 f_{\sigma(i)} + \nabla f_{\sigma(i)} \cdot \nabla f^T_{\breve{\sigma}(i)} + \nabla f_{\breve{\sigma}(i)} \cdot \nabla f^T_{\sigma(i)}$
$2c$	$T_i[f_{\sigma(i)}]$	$\dot{T}_i[f_{\sigma(i)}] \cdot \nabla^2 f_{\sigma(i)} + \nabla f_{\sigma(i)} \cdot \ddot{T}_i[f_{\sigma(i)}] \cdot \nabla f^T_{\sigma(i)}$

Table 3.3: Dyadic Hessians of the factored sequence function

We now show how to compute the gradient and Hessian of a factored sequence function.

Using elementary calculus, we easily derive the gradients and Hessians corresponding to FSF of Definition (3.1.1). They are given in Tables 3.2 and 3.3, respectively. In the tables, $\dot{T}[f]$, $\ddot{T}[f]$, and e_i are used to denote $\frac{\partial T}{\partial f}$, $\frac{\partial^2 T}{\partial^2 f}$, and the ith unit vector in $\mathbf{R}^n$, respectively. We say that the gradient is in monadic form and that the Hessian is in dyadic form.

The next section outlines key properties of the monadic and dyadic forms of the gradient and Hessian, respectively.

3.2 Properties of factorable functions

In Tables 3.2 and 3.3, we observe that the computation of $\nabla[f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}]$ involves $f_{\sigma(i)}$ and $f_{\breve{\sigma}(i)}$ which would already been computed in order to evaluate their product. Also, the computation of $\nabla^2[f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}]$ involves $\nabla f_{\sigma(i)}$ and $\nabla f_{\breve{\sigma}(i)}$ which have been computed in order to evaluate $\nabla[f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}]$. Furthermore, $\nabla^2 T_i[f_{\sigma(i)}]$ uses $\dot{T}_i[f_{\sigma(i)}]$

and $\nabla f_{\sigma(i)}$ which are previously needed in order to evaluate $\nabla T_i[f_{\sigma(i)}]$.

Thus, factorable functions have two very special properties that can be exploited to produce fast and accurate algorithms. Those special properties are:

1 Once the gradients and Hessians are written in monadic and dyadic forms, respectively, they may be computed exactly, automatically, and efficiently.

2 The Hessians occur naturally as the sum of dyad pairs.

The first property has eased the task of providing derivatives of a nonlinear programming problem to a computer code that is used to solve it. The second property changes the way we look at the matrix operations required by computer code, which in many cases, results in less computational effort. In order to motivate what follows, the structure theorems of monadic gradients and dyadic Hessians are given below. The structure theorem of monadic gradients is stated without proof. The structure theorem of dyadic Hessians, however, plays an important role in this work and, hence, the proof is given.

Theorem 3.2.1 (The structure theorem of monadic gradient)
Let $f : \mathbf{R}^n \to \mathbf{R}$ be a factorable function and $\{f_1, \ldots, f_N\}$ be the factored sequence for f. We have $f = f_N$ and $N > n$. If f_i is continuously differentiable in $\mathbf{R}^n$ for all $i = 1, \ldots, N$, then ∇f_i can be expressed as the sum of n dimensional vectors, i.e. monads, for all $i = 1, \ldots, N$, that is

$$\nabla f_i = d_i + \sum_{j=1}^{m_i} c_{ij} u_{ij} \tag{3.1}$$

where u_{ij} is a gradient of a factored sequence function, d_i and c_{ij} are composed of a product of factored sequence functions and first derivative of the single variable transformation T_i used in factored sequence, and m_i is an integer. Also, d_i is a constant and u_{ij} and c_{ij} are functions of $x \in \mathbf{R}^n$.

Theorem 3.2.2 (The structure theorem of dyadic Hessians)
Let $f : \mathbf{R}^n \to \mathbf{R}$ be a factorable function and let $\{f_1, \ldots, f_N\}$ be the factored sequence

for f. We have $f = f_N$ and $N > n$. If f_i is twice continuously differentiable in $\mathbf{R}^n$ for all $i = 1, \ldots, N$, then $\nabla^2 f_i$ can be expressed as the sums of the outer product of vectors, i.e. dyads, for all $i = 1, \ldots, N$, that is

$$\nabla^2 f_i = D_i + \sum_{j=1}^{m_i} [c_{ij}(u_{ij} v_{ij}^T + v_{ij} u_{ij}^T)] \tag{3.2}$$

where

$$D_i = diag\{d_{ij} | j = 1, \ldots, n\}.$$

We note u_{ij} and v_{ij} are gradients of a factored sequence function, D_i and c_{ij} are composed of a product of factored sequence functions and first and second derivatives of the single variable transformation T_i used in factored sequence, and m_i is an integer. Also, all D_i, u_{ij}, v_{ij}, c_{ij}, and d_{ij} are functions of $x \in \mathbf{R}^n$.

Proof: The proof is by induction on i. For $i < n + 1$, we have that $f_i = x_i$ which implies that $\nabla f_i = e_i$ and, hence, that $\nabla^2 f_i = 0_{n \times n}$.

For $i = n + 1$, we have $\sigma(i)$, $\breve{\sigma}(i) < n + 1$. We consider three cases. In case one, $\nabla^2 f_i = \nabla^2 f_{\sigma(i)} + \nabla^2 f_{\breve{\sigma}(i)}$. Thus, $\nabla^2 f_{n+1} = 0_{n \times n}$. In case two, $f_{n+1} = f_{\sigma(i)} \cdot f_{\breve{\sigma}(i)}$. Thus,

$$\begin{aligned} \nabla^2 f_{n+1} &= f_{\sigma(i)} \cdot \nabla^2 f_{\breve{\sigma}(i)} + f_{\breve{\sigma}(i)} \cdot \nabla^2 f_{\sigma(i)} + \nabla f_{\sigma(i)} \cdot \nabla f_{\breve{\sigma}(i)}^T + \nabla f_{\breve{\sigma}(i)} \cdot \nabla f_{\sigma(i)}^T \\ &= e_{\sigma(i)} \cdot e_{\breve{\sigma}(i)}^T + e_{\breve{\sigma}(i)} \cdot e_{\sigma(i)}^T. \end{aligned}$$

In this case, we get $D_i = 0_{n \times n}$ and two dyads. In case three, we have $f_{n+1} = T_i[f_{\sigma(i)}]$. Thus,

$$\begin{aligned} \nabla^2 f_{n+1} &= \dot{T}_i[f_{\sigma(i)}] \cdot \nabla^2 f_{\sigma(i)} + \nabla f_{\sigma(i)} \cdot \ddot{T}_i[f_{\sigma(i)}] \cdot \nabla f_{\sigma(i)}^T \\ &= e_{\sigma(i)} \cdot \ddot{T}_i[f_{\sigma(i)}] \cdot e_{\sigma(i)}^T. \end{aligned}$$

In this case, we have a nonzero matrix D_i, where $d_{i\sigma(i)} = \ddot{T}_i[f_{\sigma(i)}]$ and $d_{ij} = 0, j \neq \sigma(i)$.

Now, for $n+1 < i < N$, we suppose that Equation (3.2) holds. We need to show Equation (3.2) is also valid for $i+1 \leq N$. For simplicity, we use σ and $\breve{\sigma}$ to denote $\sigma(i+1)$ and $\breve{\sigma}(i+1)$. Hence, we have σ, $\breve{\sigma} < i$. From the induction hypothesis, we have

$$\nabla^2 f_\sigma = D_\sigma + \sum_{j=1}^{m_\sigma}[c_{\sigma j}(u_{\sigma j}v_{\sigma j}^T + v_{\sigma j}u_{\sigma j}^T)]$$

$$\nabla^2 f_{\breve{\sigma}} = D_{\breve{\sigma}} + \sum_{j=1}^{m_{\breve{\sigma}}}[c_{\breve{\sigma} j}(u_{\breve{\sigma} j}v_{\breve{\sigma} j}^T + v_{\breve{\sigma} j}u_{\breve{\sigma} j}^T)]$$

As before, we consider three case. In case one, we have $f_{i+1} = f_\sigma + f_{\breve{\sigma}}$. Thus,

$$\begin{aligned} \nabla^2 f_{i+1} &= \nabla^2 f_\sigma + \nabla^2 f_{\breve{\sigma}} \\ &= D_{i+1} + \sum_{j=1}^{m_{i+1}}[c_{i+1,j}(u_{i+1,j}v_{i+1,j}^T + v_{i+1,j}u_{i+1,j}^T)], \end{aligned}$$

where

$$\begin{aligned} D_{i+1} &= D_\sigma + D_{\breve{\sigma}} = diag\{d_{\sigma j} + d_{\breve{\sigma} j} | j = 1, \ldots, n\}, \\ u_{i+1,j} &= u_{\sigma j} \quad \text{for } j = 1, \ldots, m_\sigma, \\ v_{i+1,j} &= v_{\sigma j} \quad \text{for } j = 1, \ldots, m_\sigma, \\ c_{i+1,j} &= c_{\sigma j} \quad \text{for } j = 1, \ldots, m_\sigma, \\ u_{i+1,m_\sigma+j} &= u_{\breve{\sigma} j} \quad \text{for } j = 1, \ldots, m_{\breve{\sigma}}, \\ v_{i+1,m_\sigma+j} &= v_{\breve{\sigma} j} \quad \text{for } j = 1, \ldots, m_{\breve{\sigma}}, \\ c_{i+1,m_\sigma+j} &= c_{\breve{\sigma} j} \quad \text{for } j = 1, \ldots, m_{\breve{\sigma}}, \end{aligned}$$

and $m_{i+1} = m_\sigma + m_{\breve{\sigma}}$. In case two, we have $f_{i+1} = f_\sigma f_{\breve{\sigma}}$.

$$\begin{aligned} \nabla^2 f_{i+1} &= f_\sigma \nabla^2 f_{\breve{\sigma}} + f_{\breve{\sigma}} \nabla^2 f_\sigma + \nabla f_\sigma \nabla f_{\breve{\sigma}}^T + \nabla f_{\breve{\sigma}} \nabla f_\sigma^T \\ &= D_{i+1} + \sum_{j=1}^{m_{i+1}}[c_{i+1,j}(u_{i+1,j}v_{i+1,j}^T + v_{i+1,j}u_{i+1,j}^T)], \end{aligned}$$

where

$$D_{i+1} = f_\sigma D_{\breve{\sigma}} + f_{\breve{\sigma}} D_\sigma = diag\{f_\sigma d_{\breve{\sigma}j} + f_{\breve{\sigma}} d_{\sigma j} | j = 1, \ldots, n\},$$

$$u_{i+1,j} = u_{\sigma j}, v_{i+1,j} = v_{\sigma j} \text{ and } c_{i+1,j} = f_{\breve{\sigma}} c_{\sigma j} \text{ for } j = 1, \ldots, m_\sigma,$$

$$u_{i+1,m_\sigma+j} = u_{\breve{\sigma}j}, v_{i+1,m_\sigma+j} = v_{\breve{\sigma}j} \text{ and } c_{i+1,m_\sigma+j} = f_\sigma c_{\breve{\sigma}j} \text{ for } j = 1, \ldots, m_{\breve{\sigma}},$$

$$u_{i+1,j} = \nabla f_\sigma, v_{i+1,j} = \nabla f_{\breve{\sigma}}, \text{ and } c_{i+1,j} = 1 \text{ for } j = m_\sigma + m_{\breve{\sigma}} + 1,$$

and $m_{i+1} = m_\sigma + m_{\breve{\sigma}} + 1$. For the last case, we have $f_{i+1} = T_{i+1}[f_\sigma]$. Thus,

$$\begin{aligned} \nabla^2 f_{i+1} &= \dot{T}_{i+1}[f_\sigma]\nabla^2 f_\sigma + \nabla f_\sigma \ddot{T}_{i+1}[f_\sigma]\nabla f_\sigma^T \\ &= D_{i+1} + \sum_{j=1}^{m_{i+1}} [c_{i+1,j}(u_{i+1,j}v_{i+1,j}^T + v_{i+1,j}u_{i+1,j}^T)], \end{aligned}$$

where

$$D_{i+1} = \dot{T}_{i+1}[f_\sigma]D_\sigma = diag\{\dot{T}_{i+1}[f_\sigma]d_{\sigma j} | j = 1, \ldots, n\},$$

$$u_{i+1,j} = u_{\sigma j}, v_{i+1,j} = v_{\sigma j} \text{ and } c_{i+1,j} = \dot{T}_{i+1}[f_\sigma]c_{\sigma j} \text{ for } j = 1, \ldots, m_\sigma,$$

$$u_{i+1,j} = \nabla f_\sigma, v_{i+1,j} = \nabla f_\sigma \text{ and } c_{i+1,j} = \frac{\ddot{T}_{i+1}[f_\sigma]}{2} \text{ for } j = m_\sigma + 1,$$

and $m_{i+1} = m_\sigma + 1$. This completes the proof. **QED**

For ease of presentation, we suppress the index of the last element in the factored sequence, i.e. the given function f. We write the monadic gradient and the dyadic Hessian of the function f as follows:

$$\nabla f = d + \sum_{j=1}^{m} c_j u_j \tag{3.3}$$

$$\mathcal{H}_f = \nabla^2 f = D + \sum_{j=1}^{m} [c_j(u_j v_j^T + v_j u_j^T)] \tag{3.4}$$

Once the function value $f = f(x)$ has been computed, the extra amount of work required to compute the monadic gradient and dyadic Hessian involves only the effort required to compute the T_i's, $\dot{T}_i$'s, and $\ddot{T}_i$'s. Also, as long as the gradient and Hessian are kept in monadic and dyadic form, the extra storage required is minimal.

	Form	Computational requirement
f	$e^{a^T x}$	n multiplications and addition, 1 exponential
∇f	$ae^{a^T x}$	None extra in monadic form, n multiplications otherwise
$\nabla^2 f$	$ae^{a^T x}a^T$	None extra in dyadic form, $\approx \frac{n^2}{2}$ multiplications otherwise

Table 3.4: Example from McCormick [1983]

Consider the example in McCormick [1983] which is given in Table 3.4. The amount of computation required to evaluate the monadic gradient and dyadic Hessian of the function $f = e^{a^T x}$ is not much more than that required to compute the value of the function itself. Moreover, the Hessian of such function is automatically in dyadic form.

However, all the advantages of these special forms are lost when we obtain the gradient and the Hessian in their usual forms. Fortunately, it is not always necessary to do so. For example, Step 3' of the model algorithm may require the calculation of the directional derivative $\nabla f(x_0 + \rho s_0)^T s_0$ for several values of ρ. To do this, we typically compute $\nabla f(x_0 + \rho s_0)$ and then postmultiply it by s_0 for each different value of ρ. If f is a factorable function, it may be more economical to keep the gradient in its monadic form. We then have

$$\nabla f(x_0 + \rho s_0)^T s_0 = d^T s_0 + \sum_{j=1}^{m} c_j(x_0 + \rho s_0) u_j(x_0 + \rho s_0)^T s_0.$$

The quantity $d^T s_0$ need only be computed once and further use can be made of it later. Also, it is very important to keep the Hessian in its dyadic form. For example, using the SMW update formula [Sherman and Morrison, 1949; Woodbury, 1950] recursively, we can obtain $\mathcal{H}^{-1}$ without actually calculating $\mathcal{H}$.

We conclude this chapter with an example showing how to transform Sisser's function (Example 3.1.1) into its factorable form, and how to construct the corresponding dyadic Hessian. The result is given in Table 3.5.

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	f_1^4	$(4f_1^3)\nabla f_1$	$(12f_1^2)\nabla f_1 \nabla f_1^T$
4	$3f_3$	$(3)\nabla f_3$	$(36f_1^2)\nabla f_1 \nabla f_1^T$
5	f_1^2	$(2f_1)\nabla f_1$	$(2)\nabla f_1 \nabla f_1^T$
6	f_2^2	$(2f_2)\nabla f_2$	$(2)\nabla f_2 \nabla f_2^T$
7	$(-2)f_5$	$(-2)\nabla f_5$	$(-4)\nabla f_1 \nabla f_1^T$
8	$f_6 \cdot f_7$	$f_6 \nabla f_7 + f_7 \nabla f_6$	$(-4f_6)\nabla f_1 \nabla f_1^T + (2f_7)\nabla f_2 \nabla f_2^T + [\nabla f_7 \nabla f_6^T + \nabla f_6 \nabla f_7^T]$
9	f_2^4	$(4f_2^3)\nabla f_2$	$(12f_2^2)\nabla f_2 \nabla f_2^T$
10	$3f_9$	$(3)\nabla f_9$	$(36f_2^2)\nabla f_2 \nabla f_2^T$
11	$f_4 + f_8$	$\nabla f_4 + \nabla f_8$	$(36f_2^2)\nabla f_2 \nabla f_2^T + (-4f_6)\nabla f_1 \nabla f_1^T + (2f_7)\nabla f_2 \nabla f_2^T + [\nabla f_7 \nabla f_6^T + \nabla f_6 \nabla f_7^T]$
12	$f_{11} + f_{10}$	$\nabla f_{11} + \nabla f_{10}$	$(36f_1^2 - 4f_6)\nabla f_1 \nabla f_1^T + (36f_2^2 + 2f_7)\nabla f_2 \nabla f_2^T + [\nabla f_7 \nabla f_6^T + \nabla f_6 \nabla f_7^T]$

Table 3.5: FSF's, monads, and dyads of Sisser's function

Chapter 4

Sisser's method

4.1 Sisser's method

A modified Newton's method for minimizing unconstrained factorable functions which exploits the dyadic form of the Hessian was developed by Sisser around 1982. We recall that the modified Newton iteration is given by

$$x_{k+1} = x_k - \rho_k \tilde{H}_k^{-1} g_k,$$

where $\tilde{H}_k = \mathcal{H}_k$ if $\mathcal{H}_k$ is positive definite, otherwise

$$\tilde{H}_k = \mathcal{H}_k + \mu_k I \ , \tag{4.1}$$

for some $\mu_k > 0$.

The dyadic form of the Hessian is used to determine when $\mathcal{H}_k$ may be indefinite. If a possibility of indefiniteness is detected, the dyadic form is used to determine a suitable μ_k. By using the identity,

$$c(uv^T + vu^T) = (\frac{c}{2})[(u+v)(u+v)^T - (u-v)(u-v)^T], \tag{4.2}$$

Sisser first transforms the dyadic form in Equation (3.4) into the form

$$\mathcal{H} = D + \sum_{j=1}^{2m} s_j a_j a_j^T. \tag{4.3}$$

Also, Sisser orders the symmetric dyads in Equation (4.3) according to their nonzero eigenvalues, that is, to satisfy

$$s_j a_j^T a_j \geq s_{j+1} a_{j+1}^T a_{j+1}. \tag{4.4}$$

For the diagonal matrix D, Sisser replaces each $d_j \leq 0$ with a one. For each element d_j of D that is changed, a new dyad, $(d_j - 1)e_j e_j^T$ where e_j is the jth unit column vector, is added to the collection of dyads in (4.3). This give us the dyadic form

$$\mathcal{H} = \bar{D} + \sum_{j=1}^{r} s_j a_j a_j^T, \tag{4.5}$$

where all the diagonal elements of $\bar{D}$ are positive and $r \geq 2m$. Sisser suggests inverting the Hessian by means of the SMW inverse update formula,

$$(A + saa^T)^{-1} = A^{-1} - A^{-1}a(s^{-1} + a^T A^{-1} a)^{-1} a^T A^{-1}, \tag{4.6}$$

where A is a nonsingular $(n \times n)$ matrix, a is n dimensional vector, and s is a nonzero scalar. To invert the Hessian, A will be equal to $\bar{D}$ initially and the r dyads in (4.5) will be added, once at a time, by using (4.6) recursively. The formula (4.6) has the advantage of providing a convenient test which can be used before a dyad is added to determine whether the resulting matrix will still be positive definite. This test is given by following lemma.

Lemma 4.1.1 *If A is a $(n \times n)$ symmetric, positive definite matrix, then $A + saa^T$ will be positive definite if and only if*

$$1 + sa^T A^{-1} a > 0,$$

where a is a n-vector and s is a scalar.

Proof: See [Sisser, 1982]. **QED**

Since adding a dyad $s_j a_j a_j^T$ with a positive scalar to a positive definite matrix will result in a positive definite matrix, this test will be only applied when adding those dyads with $s_j < 0$. Since the dyads have been arranged so that those with positive scalar come first, we first build up a positive definite matrix and then add those dyads with negative scalars. If positive definiteness has been retained after all of the dyads have been added, we know that the original $\mathcal{H}$ is positive definite. Also, we have obtained the matrix $\mathcal{H}^{-1}$.

However, if we have found that we can no longer add dyads without losing the positive definite property, a lower bound μ on the smallest eigenvalue of $\mathcal{H}$ can be computed and used in Equation (4.1). The dyadic form of the Hessian and the following lemma enable us to determine such a μ.

Lemma 4.1.2 *Let $\bar{A} = A + saa^T$ where A is a $(n \times n)$ symmetric matrix, a be a n-vector, and s be a nonzero scalar. If $\lambda(A)_{min}$ and $\lambda(\bar{A})_{min}$ are the smallest eigenvalues of A and $\bar{A}$ respectively, then*

$$\lambda(\bar{A})_{min} \geq \lambda(A)_{min} + sa^T a.$$

Proof: See [Sisser, 1982] or [Wilkinson, 1965]. **QED**

Now, we will rewrite Equation (4.5) in the form

$$\mathcal{H} = \bar{D} + \sum_{j=1}^{p} s_j a_j a_j^T + \sum_{j=p+1}^{q} s_j a_j a_j^T + \sum_{j=q+1}^{r} s_j a_j a_j^T \tag{4.7}$$

where $s_j > 0$ for $j = 1, \ldots, p$ and $s_j < 0$ for $j = p+1, \ldots, r$. Suppose that the first q dyads have been added to $\bar{D}$, one at a time, and that the positive definite test (refer to Lemma 4.1.1) has been passed each time. Upon adding the $(q+1)$st dyad, the positive definite test is failed. It then follows from Lemma (4.1.2) that $\lambda(\mathcal{H})_{min}$

is bounded from below, i.e.

$$\lambda(\mathcal{H})_{min} \geq \sum_{j=q+1}^{r} s_j a_j^T a_j.$$

The suitable value for μ is

$$\mu = \sum_{j=q+1}^{r} \mid s_j \mid a_j^T a_j, \qquad (4.8)$$

and the modified Hessian $\tilde{H}$ will be in the form

$$\tilde{H} = \bar{D} + \sum_{j=1}^{p} s_j a_j a_j^T + \sum_{j=p+1}^{q} s_j a_j a_j^T + \sum_{j=1}^{n} \mu e_j e_j^T + \sum_{j=q+1}^{r} s_j a_j a_j^T. \qquad (4.9)$$

We note that $\sum_{j=1}^{n} \mu e_j e_j^T$ is equivalent to μI. Since μ is positive and large enough to ensure $\tilde{H}$ will be positive definite, there is no need to test for positive definiteness before adding the dyads such as $\mu e_j e_j^T$ for $j = 1, \ldots, n$ and $s_j a_j a_j^T$ for $j = q+1, \ldots, r$.

Sisser realized that a better value for μ could be obtained by splitting the $(q+1)$st and all subsequent dyads, and then absorbing parts of these dyads without losing positive definiteness. To determine such value, we take

$$\mu = \sum_{j=q+1}^{r} (1 - p_j) \mid s_j \mid a_j^T a_j \qquad (4.10)$$

where p_j represents the proportion of the jth dyad that will be absorbed for $j = q+1, \ldots, r$. The value of p_j is determined by

$$p_j = \left[\frac{(\delta - 1)}{s_j a_j^T A_{j-1}^{-1} a_j} \right] \qquad (4.11)$$

where A_{j-1}^{-1} represents the inverse of $\bar{D}$ plus the first $(j-1)$ dyads, and δ is chosen experimentally (i.e. Sisser chooses $\delta = 0.0005$). The modified Hessian $\tilde{H}$ in this case will be in the form

$$\tilde{H} = \bar{D} + \sum_{j=1}^{p} s_j a_j a_j^T + \sum_{j=p+1}^{q} s_j a_j a_j^T + \sum_{j=q+1}^{r} p_j s_j a_j a_j^T + \sum_{j=1}^{n} \mu e_j e_j^T + \sum_{j=q+1}^{r} (1 - p_j) s_j a_j a_j^T \qquad (4.12)$$

where μ is obtained by Equation (4.10) and (4.11). Similarly, there is no need to test the positive definiteness before adding the dyads such as $p_j s_j a_j a_j^T$ for $j = q+1, \ldots, r$, $\mu e_j e_j^T$ for $j = 1, \ldots, n$, and $(1 - p_j) s_j a_j a_j^T$ for $j = q + 1, \ldots, r$. This is due to the fact that

$$1 + p_j s_j a_j^T A_{j-1}^{-1} a_j = \delta > 0$$

and hence the $p_j s_j a_j a_j^T$ for $j = q + 1, \ldots, r$ satisfies the positive definite test. The scalar μ is large enough to ensure $\tilde{H}$ will be positive definite after adding the rest of dyads.

The reason for splitting and absorbing parts of the remaining dyads is that the value of μ can be reduced. As a result, Sisser's method will be less likely to produce steepest descent steps.

4.2 Discussion of Sisser's method

Sisser's method was compared to the BFGS Quasi-Newton method [1970], Gill and Murray's method [1977], and Sorensen's method [1979] on a series of unconstrained test problems. These problems are popular with authors of modified Newton methods since, starting from the standard point, they force the algorithm to proceed through regions in which Hessian is indefinite. We note that some of test problems and computational results given by Sisser are also given in Chapter six and Appendix A.

The amount of computation per iteration in Sisser's method is considerably greater than in the BFGS method. This is because of the transformations required to obtain the dyads form (4.3), the preordering of dyads to satisfy (4.4), the expensive computation of the lower bound μ on the smallest eigenvalue of $\mathcal{H}$ when further updates destroy positive definiteness, and the computation of $\mathcal{H}^{-1}$. On the other hand, Sisser's method seems to require fewer iterations than the BFGS method. Also, Sisser's method appears to be competitive with both Sorenson's and Gill and Murray's methods.

However, Sisser's method had one clear advantage over BFGS method. It solved all the recorded test problems. Thus, if reliability is a more important criterion than execution time, Sisser's method merits more consideration. In Chapter five, we present a new modified Newton's method that was motivated by our study of Sisser's method.

Chapter 5

A new modified Newton's method

5.1 Introduction

In Chapter four, we have seen an indication of the advantages of a modification of Newton's method which was based on a representation of the objective function in factorable form. The advantages were essentially due to the resulting dyadic form of the Hessian matrix.

Chapter four also indicated some disadvantages of Sisser's modified Newton method. They are (1) the transformation of the given unsymmetric dyads into symmetric dyads, (2) the subsequent ordering of the symmetric dyads according to their nonzero eigenvalues, and (3) in the case of indefiniteness, the calculation of a perturbation parameter.

These disadvantages result in a large computational cost for each Sisser iteration. However, Sisser's method is still competitive with other unconstrained minimization techniques, for example, the BFGS method [Broyden,1970; Fletcher, 1970], Gill and Marry's method [1974], and Sorensen's method [1977]. Thus, a new method based on factorable function, which significantly decreases the cost per iteration, would be a valuable contribution to the field.

Sisser states that the main reasons for the transformation of unsymmetric dyads into symmetric dyads are (1) to allow the use of the symmetric rank one update formula in calculating the inverse Hessian, (2) to provide a mechanism to order the dyads, (3) to test the positive definiteness of the Hessian, and (4) to calculate the perturbation parameter.

In this chapter, we show how Sisser's method can be modified so that the transformation of the dyads is unnecessary. In addition, we give arguments to indicate that it is both unnecessary to order the dyads and to calculate the perturbation parameter. This results in a new modification of Newton's for the solution of the unconstrained minimization problem.

The modification is based on the observation that most of the dyads actually appear naturally in pairs, and that the pairs are symmetric. By making use of a symmetric rank two update formula [cf. Best and Caron, 1985] we can still calculate the inverse Hessian. Furthermore, using the formulas given in Caron and Gould [1986], we can still test the positive definiteness of the Hessian matrix. In addition, for dyad pairs or dyads which cause indefiniteness, these formulas automatically provide a perturbation parameter for adjusting the dyad pairs or dyads so that positive definiteness is maintained.

5.2 The dyadic form of the Hessian

Let $f(x)$ be a factorable function that is twice continuously differentiable. From Theorem (3.2.2), we have that the Hessian can be written in the dyadic form

$$\mathcal{H} = D + \sum_{j=1}^{m} \left[c_j (u_j v_j^T + v_j u_j^T) \right] \tag{5.1}$$

where D is a diagonal matrix, $u_1, u_2, \ldots, u_m$ are n-vectors; and where $c_1, c_2, \ldots, c_n$ are scalars. It is important to note that D, and u_j, v_j and c_j, $j = 1, \ldots, m$, are functions of $x \in \mathbf{R}^n$.

In general, there are some indices j such that u_j and v_j are linearly dependent. In this case, the dyad pair $c_j(u_j v_j^T + v_j u_j^T)$ can be replaced with a single symmetric dyad. We note that it is possible to later combine some of the single dyads into dyad pairs. Thus, the Hessian can be written in the dyadic form

$$\mathcal{H} = D + \sum_{j=1}^{p}[\alpha_j(u_jv_j^T + v_ju_j^T)] + \sum_{j=p+1}^{p+q}[\beta_j(u_ju_j^T + v_jv_j^T)] + \sum_{j=p+q+1}^{p+q+r}[\gamma_ja_ja_j^T] \quad (5.2)$$

where u_j and v_j are now linearly independent for all $j = 1, \ldots, p+q$. We remark that it is, in some sense, more natural to use the dyadic form given in (5.2) rather than the form given in (5.1). This is because terms of each of those dyad types, i.e., $\alpha_j(u_jv_j^T + v_ju_j^T)$, $\beta_j(u_ju_j^T + v_jv_j^T)$ and $\gamma_ja_ja_j^T$ arise naturally when determining the dyadic Hessian (refer to the examples in Appendix A).

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_s\}$ be such that $d_i \leq 0$ if $i \in \mathcal{I}$ and $d_i > 0$ if $i \notin \mathcal{I}$. Let $\bar{D}$ be obtained from D by replacing d_i with $\bar{d}_i$, $\bar{d}_i > 0$, for each $i \in \mathcal{I}$ (note: Sisser's method set $\bar{d}_i = 1$ for all $i \in \mathcal{I}$). Thus, we can now write

$$\mathcal{H} = \bar{D} + \sum_{j=1}^{p}[\alpha_j(u_jv_j^T + v_ju_j^T)] + \sum_{j=p+1}^{p+q}[\beta_j(u_ju_j^T + v_jv_j^T)] + \sum_{j=p+q+1}^{p+q+r}[\gamma_ja_ja_j^T] + \sum_{j=p+q+r+1}^{p+q+r+s}[\delta_jb_jb_j^T] \quad (5.3)$$

where $b_j = e_{i_j}$is the i_j th elementary column vector, $\delta_j = (d_{i_j} - \bar{d}_{i_j})$, and where $\bar{D}$ is positive definite. For simplicity, we rewrite (5.3) in recursive form, i.e.

$$H_j = H_{j-1} + \tau_jB_j\ , \quad j = 1, \ldots, p+q+r+s \quad (5.4)$$

where

$$B_j = \begin{cases} (u_jv_j^T + v_ju_j^T) & j = 1, \ldots, p \\ (u_ju_j^T + v_jv_j^T) & j = p+1, \ldots, p+q \\ a_ja_j^T & j = p+q+1, \ldots, p+q+r \\ b_jb_j^T & j = p+q+r+1, \ldots, p+q+r+s, \end{cases}$$

$$\tau_j = \begin{cases} \alpha_j & j = 1, \ldots, p \\ \beta_j & j = p+1, \ldots, p+q \\ \gamma_j & j = p+q+1, \ldots, p+q+r \\ \delta_j & j = p+q+r+1, \ldots, p+q+r+s, \end{cases}$$

and $H_0 = \bar{D}$. Setting $M = p+q+r+s$, the dyadic Hessian is given by $\mathcal{H} = H_M$. It is clear from Equation (5.4) that $\tau_j B_j$ is either a symmetric rank two or a symmetric rank one matrix for all $j = 1, \ldots, M$. We note that this recursive form of the dyadic Hessian will be used throughout this chapter.

The inverse of the Hessian can be obtained using the dyadic form given by Equation (5.4) as follows :

$$H_j^{-1} = (H_{j-1} + \tau_j B_j)^{-1}, \quad j = 1, \ldots, M \tag{5.5}$$

where $H_0^{-1} = \bar{D}^{-1}$ and $\mathcal{H}^{-1} = H_M^{-1}$. For each $j = 1, \ldots, M$, we can calculate H_j^{-1} using the symmetric rank two update formula given in next section.

5.3 Symmetric rank two update

Best and Caron [1985] give a form of the Sherman-Morrison-Woodbury update formula which can be used to obtain the inverse of the dyadic Hessian. It is presented in the following lemma, which is given without proof.

Lemma 5.3.1 *Let u and v be linearly independent n-vectors, c and λ be scalars, and H be a singular $(n \times n)$ matrix. Then $\hat{H} = H + c(uu^T + \lambda vv^T)$ is nonsingular if and only if*

$$\hat{\psi} = 1 + [u^T H^{-1} u + \lambda v^T H^{-1} v]c + \lambda[u^T H^{-1} u v^T H^{-1} v - (v^T H^{-1} u)^2]c^2 \neq 0.$$

If $\hat{\psi} \neq 0$, then

$$\hat{H}^{-1} = \hat{\psi}^{-1}[T_1 + T_2 c + T_3 c^2],$$

$$\begin{aligned}
\textit{where}\quad T_1 &= H^{-1}, \\
T_2 &= [u^T H^{-1} u + \lambda v^T H^{-1} v] H^{-1} - [H^{-1} u u^T H^{-1} + \lambda H^{-1} v v^T H^{-1}], \\
\textit{and}\quad T_3 &= \lambda [u^T H^{-1} u v^T H^{-1} v - (v^T H^{-1} u)^2] H^{-1} - \\
&\quad \lambda [v^T H^{-1} v H^{-1} u u^T H^{-1} + u^T H^{-1} u H^{-1} v v^T H^{-1}] + \\
&\quad \lambda [(v^T H^{-1} u)(H^{-1} v u^T H^{-1} + H^{-1} u v^T H^{-1})].
\end{aligned}$$

Lemma (5.3.1) can be used directly in Equation (5.5) provided that $\tau_j B_j$ is of the type $\beta_j(u_j u_j^T + v_j v_j^T)$, $\gamma_j a_j a_j^T$, or $\delta_j b_j b_j^T$. In order to handle updates of the type $\alpha_j(u_j v_j^T + v_j u_j^T)$, we require the following lemma, which is easily derived from Lemma (5.3.1).

Lemma 5.3.2 *Let u and v be linearly independent n-vector, c be a scalar, and H be a nonsingular $(n \times n)$ matrix. Then $\check{H} = H + c(uv^T + vu^T)$ is nonsingular if and only if*

$$\check{\psi} = 1 + 2[v^T H^{-1} u]c - [(u^T H^{-1} u)(v^T H^{-1} v) - (v^T H^{-1} u)^2]c^2 \neq 0.$$

If $\check{\psi} \neq 0$, then

$$\check{H}^{-1} = \check{\psi}^{-1}[T_1 + T_2 c + T_3 c^2],$$

$$\begin{aligned}
\textit{where}\quad T_1 &= H^{-1}, \\
T_2 &= 2[v^T H^{-1} u] H^{-1} - [H^{-1} v u^T H^{-1} + H^{-1} u v^T H^{-1}], \\
\textit{and}\quad T_3 &= [u^T H^{-1} u][H^{-1} v v^T H^{-1}] + [v^T H^{-1} v][H^{-1} u u^T H^{-1}] - \\
&\quad [(u^T H^{-1} u)(v^T H^{-1} v) - (v^T H^{-1} u)^2] H^{-1} - \\
&\quad [v^T H^{-1} u][H^{-1} u v^T H^{-1} + H^{-1} v u^T H^{-1}].
\end{aligned}$$

Proof: Using the identity given in Equation (4.2), the result follows from previous lemma. **QED**

The algorithm used to obtain the inverse of dyadic Hessian is given as follows.

Algorithm 5.3.1

```
1 Set, $H_0^{-1} \leftarrow \bar{D}^{-1}$, $j \leftarrow 1$
2 If $j > M$ then stop with $\mathcal{H}^{-1} = H_M^{-1}$
3 Set $H_j^{-1} \leftarrow (H_{j-1} + \tau_j B_j)^{-1}$ [ Apply with Lemma (5.3.1) or (5.3.2) ]
4 Set $j \leftarrow j + 1$ and goto 2
```

We recall that in order for the search direction, $d_k = -\tilde{H}_k^{-1} g_k$, obtained in step 2 of the model algorithm (refer to Section 1.3) to be a descent direction, we want $\tilde{H}_k$ to be positive definite. If $\mathcal{H}$ is positive definite, then we can set $\tilde{H}_k = H_M = \mathcal{H}$. However, if $\mathcal{H}$ is not positive definite, we would like to modify Algorithm (5.3.1) so that H_M is some positive definite approximation to $\mathcal{H}$. Then, we can set $\tilde{H}_k = H_M$. This is discussed in Section 5.6.

In order to do this we use the results given in the next section to determine whether or not H_j is positive definite for all j. Clearly, if this is the case, then $\mathcal{H}$ is positive definite. However, we note that while this condition is sufficient, it is certainly not necessary. That is, $\mathcal{H}$ may be positive definite even though some of the H_j are not.

5.4 Positive semidefinite interval

The problem of finding a positive semidefinite interval for a parametric matrix has been well studied by Caron and Gould [1986]. They show that if we are given two symmetric $n \times n$ matrices $\bar{H}$ and $\bar{B}$ such that $\bar{H}$ is positive semidefinite and $\bar{B}$ is of rank one or two, we can determine two real number $\check{\tau} \leq 0$ and $\hat{\tau} \geq 0$ such that $\bar{H} + \tau\bar{B}$ is positive semidefinite if and only if $\tau \in [\check{\tau}, \hat{\tau}]$..

This provides us an easy way to test the positive definiteness of the matrix $H_j = H_{j-1} + \tau_j B_j$ before we add $\tau_j B_j$ to H_{j-1} (refer to Equation 5.4). Also, this interval automatically provides a perturbation parameter for adjusting the matrix $\tau_j B_j$ such that positive definiteness is maintained (see Section 5.6).

The following lemmas give formulas for the positive semidefinite interval $[\check{\tau}, \hat{\tau}]$.

Lemma 5.4.1 *Let u and v be linearly independent n-vectors, τ be a scalar, and H be a symmetric, positive definite $(n \times n)$ matrix. Then, the positive semidefinite interval for the matrix $\hat{H} = H + \tau(uu^T + vv^T)$ is given by $[\check{\tau}, \hat{\tau})$ where*

$$\begin{aligned} \hat{\tau} &= +\infty \quad \textit{and} \\ \check{\tau} &= 2\left[-u^T H^{-1} u - v^T H^{-1} v - \sqrt{(u^T H^{-1} u - v^T H^{-1} v)^2 + 4(v^T H^{-1} u)^2}\right]^{-1}. \end{aligned}$$

Also, we have

1 $\hat{H}$ is positive semidefinite if and only if $\tau = \check{\tau}$.

2 $\hat{H}$ is positive definite if and only if $\tau > \check{\tau}$.

Proof: See Table one of Caron and Gould [1986]. **QED**

We note that Lemma (5.4.1) allows the n-vector $v = 0$.

Lemma 5.4.2 *Let u and v be linearly independent n-vectors, τ be a scalar, and H be a symmetric, positive definite $(n \times n)$ matrix. Then, the positive semidefinite interval for the matrix $\check{H} = H + \tau(uv^T + vu^T)$ is given by $[\check{\tau}, \hat{\tau}]$ where*

$$\begin{aligned} \hat{\tau} &= \left[-u^T H^{-1} v + \sqrt{(u^T H^{-1} u)(v^T H^{-1} v)}\right]^{-1} \quad \textit{and} \\ \check{\tau} &= \left[-u^T H^{-1} v - \sqrt{(u^T H^{-1} u)(v^T H^{-1} v)}\right]^{-1}. \end{aligned}$$

Also, we have

1 $\check{H}$ is positive semidefinite if and only if $\tau = \check{\tau}$ or $\tau = \hat{\tau}$.

2 $\check{H}$ is positive definite if and only if $\tau \in (\check{\tau}, \hat{\tau})$.

Proof: Apply the identity (4.2), the result follows from the equation given in Table one of Caron and Gould [1986]. **QED**

As we have stated before, we want H_j^{-1} to be positive definite for all j. A naive strategy for handling an update which destroys the positive definite property is to adjust the update according to some parameter determined either by Lemma (5.4.1) or (5.4.2). This naive strategy is given below.

Algorithm 5.4.1

```
1 Set $H_0^{-1} \leftarrow \tilde{D}^{-1}; j \leftarrow 1$
2 If $j > M$ then stop with $H_M^{-1}$
3 Determine $[\breve{\tau}_j, \hat{\tau}_j]$ for $(H_{j-1} + \tau B_j)$
4 If $\tau_j \notin (\breve{\tau}_j, \hat{\tau}_j)$ then goto 5
  Set $H_j^{-1} \leftarrow (H_{j-1} + \tau_j B_j)^{-1}$
  Set $j \leftarrow j + 1$
  Goto 2
5 Choose $\bar{\tau}_j \in (\breve{\tau}_j, \hat{\tau}_j)$
  Set $H_j^{-1} \leftarrow (H_{j-1} + \bar{\tau}_j B_j)^{-1}$
  Set $j \leftarrow j + 1$
  Goto 2
```

We note that if B_j is of type $(u_j u_j^T + v_j v_j^T)$, $a_j a_j^T$, or $b_j b_j^T$, the positive definiteness test will only be needed when the corresponding $\tau_j < 0$. It should also be noted that $\breve{\tau}_j$ and $\hat{\tau}_j$ are easily calculated using only matrix-vector products. This is because H_{j-1}^{-1}, not H_{j-1}, is available in Algorithm (5.4.1).

One disadvantage of this strategy is that it does not guarantee $H_M^{-1} = \mathcal{H}^{-1}$ even if the Hessian is positive definite. To resolve this problem, we need a strategy for ordering the dyad pairs and the single dyads.

5.5 Ordering

If the Hessian is positive definite, then whether or not H_j is positive definite for each j depends on the order in which the dyads are added in Equation (5.4). Consider the following example.

Example 5.5.1 *Let*

$$D = \begin{pmatrix} 10 & 0 \\ 0 & 3 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 0 \\ 0 & -4 \end{pmatrix}, B_2 = \begin{pmatrix} -6 & 0 \\ 0 & 0 \end{pmatrix}, \text{ and } B_3 = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

Clearly, in order that each $H_j, j = 1,2,3,$ be positive definite, the dyads must be added in any order except $(B_1, B_2, B_3), (B_1, B_3, B_2),$ and (B_2, B_1, B_3).

Sisser overcomes the problem of ordering by first performing a "Static" ordering and then a "dynamic" ordering. The static ordering is done to ensure that inequalities (4.4) are satisfied, and is completed before the updates begin. The dynamic ordering is performed during the inverse update algorithm, i.e. Sisser's version of Algorithm (5.4.1). The dynamic ordering essentially avoids nonpositive definite updates until no positive definite updates remain.

We point out that Sisser's static ordering is a heuristic. Consider the following example.

Example 5.5.2 *Let*

$$D = \begin{pmatrix} 1.00005 & 0 \\ 0 & 2 \end{pmatrix}, B_1 = \begin{pmatrix} -1 & 0.2 \\ 0.2 & -0.04 \end{pmatrix}, \text{ and } B_2 = \begin{pmatrix} 0 & 0 \\ 0 & -1.05 \end{pmatrix}.$$

The nonzero eigenvalues $\lambda(B_1)$ and $\lambda(B_2)$ of B_1 and B_2 are -1.04 and -1.05 respectively. According to static ordering used by Sisser, we first add B_1 to D and then B_2. Clearly, the matrix $(D + B_1)$ is not positive definite. However, we have that $(D + B_2)$ is positive definite.

As we will see in the next section, we can also provides a static ordering scheme. However, another example can be constructed to show that , as with Sisser's static ordering, it is simply a heuristic. That is, it may or may not work. Thus, it is not used in our method. This gives us a further saving in computational cost at each iteration. We note that Sisser's method can also drop the static ordering scheme to achieve the same savings.

The essential ordering is the dynamic ordering. It is equivalent to Sisser's dynamic ordering and is described in section 5.5.2 .

5.5.1 Static ordering

Equivalent to Sisser's static ordering, i.e. inequalities (4.4), we can order the symmetric dyad pairs and symmetric dyads according their smallest eigenvalues. We require the following lemma.

Lemma 5.5.1 *Let u and v be linearly independent n-vectors, and c be a nonzero scalar. For any n-vectors x and y, the nonzero eigenvalues of the matrix $c(xv^T + yu^T)$ are*

$$\lambda = \left(\frac{c}{2}\right)\left[v^Tx + u^Ty \pm \sqrt{(v^Tx - u^Ty)^2 + 4u^Txv^Ty}\right].$$

Proof: Let T be a nonsingular matrix, $x, y, u,$ and v be n-vectors, and c be a nonzero scalar. Suppose that u and v are linearly independent. To determine the nonzero eigenvalues of $c(xv^T + yu^T)$, we set $det[\lambda I - c(xv^T + yu^T)] = 0$. Since

$$det(T^{-1}AT) = det(T^{-1})det(A)det(T) = det(A),$$

we have

$$det[\lambda I - c(xv^T + yu^T)] = det[\lambda I - cT^{-1}xv^TT - cT^{-1}yu^TT].$$

Now, we choose T to satisfy the equations

$$v^TT = e_1^T \text{ and } u^TT = e_2^T$$

where e_i are the unit coordinate vectors. Let $T^{-1}x = a$ and $T^{-1}y = b$, where $a = (a_1, \ldots, a_n)^T$ and $b = (b_1, \ldots, b_n)^T$ are n-vectors. Then, the determinant becomes

$$\begin{aligned} det[\lambda I - cae_1^T - cbe_2^T] &= det[\lambda e_1 - ca, \lambda e_2 - cb, \lambda e_3, \ldots, \lambda e_n] \\ &= \lambda^{n-2}[(\lambda - ca_1)(\lambda - cb_2) - c^2a_2b_1]. \end{aligned}$$

Since

$$\begin{aligned} a_1 &= e_1^T a = e_1^T T^{-1} x = v^T x, \\ a_2 &= e_2^T a = e_2^T T^{-1} x = u^T x, \\ b_1 &= e_1^T b = e_1^T T^{-1} y = v^T y, \text{ and} \\ b_2 &= e_2^T b = e_2^T T^{-1} y = u^T x, \end{aligned}$$

we have that the nonzero eigenvalues are roots of $(\lambda - cv^T x)(\lambda - cu^T y) - c^2 u^T x v^T y = 0$. Thus, the nonzero eigenvalues are given by

$$\lambda = \left(\frac{c}{2}\right) \left[v^T x + u^T y \pm \sqrt{(v^T x - u^T y)^2 + 4u^T x v^T y}\right].$$

This completes the proof. **QED**

The above lemma enables us to determine the nonzero eigenvalues for the dyad pairs and pairs defined in Equation (5.4). It is clear from Lemma (5.5.1), the formula for the nonzero eigenvalues gives the largest and smallest eigenvalues of the matrix $c(xv^T + yu^T)$. Let $\lambda(\bar{B})_{max}$ and $\lambda(\bar{B})_{min}$ represent the largest and smallest eigenvalues of a matrix $\bar{B}$, respectively. Suppose that c is greater than zero. Then, we have

$$\lambda\left(c(xv^T + yu^T)\right)_{max} = \left(\frac{c}{2}\right) \left[v^T x + u^T y + \sqrt{(v^T x - u^T y)^2 + 4u^T x v^T y}\right] \qquad (5.6)$$

and

$$\lambda\left(c(xv^T + yu^T)\right)_{min} = \left(\frac{c}{2}\right) \left[v^T x + u^T y - \sqrt{(v^T x - u^T y)^2 + 4u^T x v^T y}\right]. \qquad (5.7)$$

This gives us the following results.

1 Let $x = u$ and $y = v$. Then,

$$\begin{aligned} \lambda\left(c(uv^T + vu^T)\right)_{max} &= c\left[u^T v + \| u \| \| v \|\right] \\ &\textit{and} \\ \lambda\left(c(uv^T + vu^T)\right)_{min} &= c\left[u^T v - \| u \| \| v \|\right] \end{aligned} \tag{5.8}$$

2 Let $x = v$ and $y = u$. Then,

$$\begin{aligned} \lambda\left(c(uu^T + vv^T)\right)_{max} &= \left(\frac{c}{2}\right)\left[u^T u + v^T v + \sqrt{(u^T u - v^T v)^2 + 4u^T v}\right] \\ &\text{and} \\ \lambda\left(cuu^T + vv^T)\right)_{min} &= \left(\frac{c}{2}\right)\left[u^T u + v^T v - \sqrt{(u^T u - v^T v)^2 + 4u^T v}\right] \end{aligned} \tag{5.9}$$

3 Let $x = 0$ and $y = u$. Then,

$$\begin{aligned} \lambda\left(cuu^T\right)_{max} &= c\left[u^T u\right] \\ &\textit{and} \\ \lambda\left(cuu^T\right)_{min} &= 0 \end{aligned} \tag{5.10}$$

From Wilkinson [1965] or Shuzhi [1987], we have the following lemma.

Lemma 5.5.2 *Let $H, \bar{H}$, and $\bar{B}$ be $(n \times n)$ symmetric matrices. We denote the sets of eigenvalues of $H, \bar{H}$, and $\bar{B}$ by $\{\lambda(H)_i\}, \{\lambda(\bar{H})_i\}$, and $\{\lambda(\bar{B})_i\}$, respectively and where all three sets are arranged in non-increasing order. If $H = \bar{H} + \bar{B}$, then*

$$\lambda(\bar{B})_n \leq \lambda(H)_i - \lambda(\bar{H})_i \leq \lambda(\bar{B})_1$$

for all $i = 1, \ldots, n$.

Lemma (5.5.2) implies that when $\bar{B}$ is added to $\bar{H}$, all of its eigenvalues are changed by an amount which lies between the smallest and the greatest eigenvalues of $\bar{B}$. Consider the smallest eigenvalue, we have

$$\lambda(H)_{min} \geq \lambda(\bar{H})_{min} + \lambda(\bar{B})_{min} \cdot \qquad (5.11)$$

Assume that $\bar{H}$ is positive definite, i.e. $\lambda(\bar{H})_{min} > 0$. It is obviously that the matrix H will not lose the positive definiteness when $\bar{B}$ is not indefinite, i.e. $\lambda(\bar{B})_{min} \geq 0$. Now, we suppose $\bar{B}$ is indefinite. Then, the possibility for losing positive definiteness of the matrix $H = \bar{H} + \bar{B}$ will increase when $\lambda(\bar{B})_{min}$ becomes smaller and smaller. This implies the order of the dyad pairs and dyads should be arranged by means of the following relation (c.f. Equation 5.4).

$$\lambda(\tau_j B_j)_{min} \geq \lambda(\tau_{j+1} B_{j+1})_{min}\ , \quad j = 1, \ldots, M \qquad (5.12)$$

From Equation (5.8 - 5.10), the smallest eigenvalues of the dyad pairs and dyads can be easily obtained. To illustrate this idea, we give the following example.

Example 5.5.3 *Let*

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 1.2 \\ 1.2 & 0 \end{pmatrix}, \ and\ B_2 = \begin{pmatrix} 0 & -0.8 \\ -0.8 & 0 \end{pmatrix}.$$

Without any static ordering, we have that $(D + B_1)$ is indefinite and $(D + B_1 + B_2)$ is positive definite. The smallest eigenvalues $\lambda(B_1)_{min}$ and $\lambda(B_2)_{min}$ of B_1 and B_2 are -1.2 and -0.8 respectively. According to static ordering given by inequality (5.12), we first add B_2 to D and then B_1. Clearly, the matrix $(D + B_2)$ is positive definite and so is $(D + B_2 + B_1)$.

Now, the algorithm (5.4.1) can be modified as follows.

Algorithm 5.5.1

1 Obtain the order $\lambda(\tau_j B_j)_{min} \geq \lambda(\tau_{j+1} B_{j+1})_{min}$ for $j = 1, \ldots, M$

```
2 Set $H_0^{-1} \leftarrow \bar{D}^{-1}; j \leftarrow 1$
3 If $j > M$ then stop with $H_M^{-1}$
4 Determine $[\check{\tau}_j, \hat{\tau}_j]$ for $(H_{j-1} + \tau B_j)$
5 If $\tau_j \notin (\check{\tau}_j, \hat{\tau}_j)$ then goto 6
  Set $H_j^{-1} \leftarrow (H_{j-1} + \tau_j B_j)^{-1}$
  Set $j \leftarrow j + 1$
  Goto 2
6 Choose $\bar{\tau}_j \in (\check{\tau}_j, \hat{\tau}_j)$
  Set $H_j^{-1} \leftarrow (H_{j-1} + \bar{\tau}_j B_j)^{-1}$
  Set $j \leftarrow j + 1$
  Goto 2
```

5.5.2 Dynamic ordering

A new scheme can be used to obtain an implicit, i.e. dynamic,ordering. If, for some j, it was determined, using Lemmas (5.4.1) and (5.4.2), that H_j^{-1} was not positive definite, this update would be delayed and another update performed. A necessary condition for a nonpositive definite $\mathcal{H}$ is that, for some j, all remaining updates result in an indefinite H_j. However, this is not a sufficient condition. Consider the following example.

Example 5.5.4 *Let*

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 1.3 \\ 1.3 & 0 \end{pmatrix}, \text{ and } B_2 = \begin{pmatrix} 0 & -1.1 \\ -1.1 & 0 \end{pmatrix}.$$

Obviously, we have $(D + B_1)$ and $(D + B_2)$ are indefinite but $(D + B_1 + B_2)$ is positive definite.

In order to handle all the remaining updates which cause indefiniteness, those updates would be adjusted according to some parameter determined by Lemmas (5.4.1) and (5.4.2).

The Examples (5.5.1 - 5.5.4) show that the dynamic ordering scheme can obtain the same order as the static scheme. This implies that only the dynamic scheme need be used. However, the dynamic scheme may be required independent of whether or not the static scheme is used. Therefore, only the dynamic ordering scheme is used in our new modified Newton's method. The inverse update algorithm is now given by

Algorithm 5.5.2

```
1 Main() : Dynamic ordering

        set goahead ← false ; moretogo ← true.
        set pick(j) ← false, $\forall_{j=1,\ldots,M}$.
        set $H^{-1}$ ← $\bar{D}^{-1}$.
        while ($M > 0$)
            set gupdate ← false
            for j ← 1, ..., M
                if ^pick (j) then
                    goto sub
                end if
                if update then
                    set $M$ ← $M - 1$ ; gupdate ← true.
                    set pick(j) ← true ; goahead ← false
                end if
            end for
            if ^gupdate then
                set goahead ← true
            end if
            if $M = 1$ then
                set moretogo ← false
            end if
end while
Stop with $H_M^{-1}$
```

2-sub() : Positive definiteness test and Symmetric rank two update

```
set $\bar{\tau}_j \leftarrow \tau_j$.
Determine $[\check{\tau}_j, \hat{\tau}_j]$.
if $\tau_j \notin [\check{\tau}_j, \hat{\tau}_j]$ then
    if ˆgoahead then
        Return to main
    else
        Choose $\bar{\tau}_j \in (\check{\tau}_j, \hat{\tau}_j)$
    end if
end if
if $(\tau_j = \check{\tau}_j$ or $\tau_j = \hat{\tau}_j)$ and ˆmoretogo then
    Choose $\bar{\tau}_j \in (\check{\tau}_j, \hat{\tau}_j)$
end if
set $H^{-1} \leftarrow (H + \bar{\tau}_j B_j)^{-1}$; update $\leftarrow$ true
Return to main.
```

Note: the symbol ˆ is defined to be the logical NOT operation.

5.6 A strategy for indefinite matrix

Algorithm (5.4.1), (5.5.1), and (5.5.2), all require the choice of a perturbation parameter, $\bar{\tau} \in [\check{\tau}, \hat{\tau}]$, when we can no longer add dyad pairs or dyads without losing positive definiteness. Computational results show that the choice of such a parameter has little effect in solving the minimization problem. This is because those update strategies do provide a positive definite matrix and hence a descent direction can be obtained. However, it does affect the updates for remaining, if any, dyad pairs and dyads. In order to minimize the need of adjustment, it is suggested that we choose a parameter which will give, in some sense, the "most" positive definite matrix after the update. As a result, the remaining dyad pairs and dyads may not need to be modified.

Based on this argument, the perturbation parameter is chosen to be the midpoint of the positive definite interval. For the dyad pairs and dyads in which have the upper bound in the interval of infinity, the parameter is then chosen to be the absolute value of the lower bound of that interval. We note that unlike Sisser's method, our modification will not bias search direction towards the steepest descent direction.

We recall from Section 5.2 that if the given diagonal matrix $\bar{D}$ is indefinite, then all of its nonpositive diagonal elements must be replaced by positive value, say $\bar{d}$. Again, computational results show that some care must be taken in choosing $\bar{d}$. In particular, $\bar{d}$ should depend on the matrix $\bar{D}$. In our implementation, we choose

$$max\{ \mid d_j \mid \mid j = 1, \ldots, n\} + 1.$$

5.7 Conclusion

This chapter presented, via Algorithm (5.5.2), a new modification of Newton's method for minimizing factorable functions. The method exploits the fact that most of the dyads appear naturally in pairs. This avoids the transformation required by Sisser's method in each iteration to convert the given dyad pairs to a string of symmetric dyads. The explicit ordering has been shown to be unnecessary when implicit ordering is used. To test the positive definiteness, we use the technique of positive semidefinite interval. Since this technique automatically provides a perturbation parameter for handling updates which destroy the positive definite property, it overcomes the fact that Sisser's method requires the expensive calculation of the perturbation parameter. Avoidance of splitting dyads which cause indefiniteness, our method results in significant computational savings.

Based on the above arguments, the cost per iteration of this new modification should be significantly less than the cost required for Sisser's iteration. It should be noted that Sisser's method could also be modified so that it avoids the static ordering and the calculation of the perturbation parameter. However, it cannot avoid the transformation.

Chapter 6

Computational results

Our new modification of Newton's method was implemented using the computer language PL/I on the IBM 4381 P13 processor (VM/CMS) at University of Windsor. The computational results are compared to the results given in Sisser [1982] for a series of two dimensional test problems. The FSF (Factorable Sequence Functions), Monads, and Dyads of each test problems are given in Appendix A. The computational results are listed in Table 6.1 along with Sisser's results. We should note that the given comparison between our new modification and Sisser's modification does not provide the complete picture. In particular, a comparison of execution time is not possible, as Sisser used a different computer to generate his test results.

However, some pertinent remarks can be made. We note that, as with Sisser's method, our method solved all the test problems, while BFGS failed on Test Problem 2. We required about the same number of iterations to solve the problems as did Sisser. However, our iterations would be significantly less costly. Finally, our method outperformed Sisser's method on the last test problem.

Test problem	Complexity of New modification (a/b)	Complexity of Sisser's modification (a/b)	New modification $(c : d)$	Sisser $(c : d)$	BFGS $(c : d)$
Sisser's function, $n = 2$	(2/3)	(2/1)	$(14 : 15)$	$(15 : 21)$	$(22 : 30)$
Rosenbrock's cliff function, $n = 2$	(2/0)	(2/0)	$(27 : 28)$	$(27 : 28)$	$(* : *)$
Rosenbrock's function, $n = 2$	(1/0)	(1/0)	$(24 : 33)$	$(21 : 28)$	$(32 : 52)$
Hyperbola-circle function, $n = 2$	(4/2)	(4/2)	$(7 : 11)$	$(6 : 9)$	$(9 : 14)$
Beale's function, $n = 2$	(9/1)	(9/1)	$(8 : 10)$	$(7 : 9)$	$(13 : 21)$
Gottfriend's function, $n = 2$	(6/6)	(6/3)	$(14 : 21)$	$(9 : 12)$	$(18 : 26)$
Powell's badly scaled function, $n = 2$	(4/6)	(4/48)	$(36 : 45)$	$(74 : 139)$	$(- : -)$

where

$-$ = no record

$*$ = line search difficulty

a = number of original dyads

b = number of times modification was used

c = number of iterations

d = number of function evaluation

Table 6.1: Table of Comparison

Chapter 7

Concluding remarks

7.1 Introduction

This chapter serves as both a summary of the contributions of this thesis, and as a guide to future research projects.

7.2 Contribution of Thesis

The main contributions of this thesis are listed below.

1 This thesis provides a new modification of Newton's method for the minimization of factorable functions. The new method takes advantage of the fact that the Hessian can be expressed as a diagonal matrix plus a sum of dyads, many of which appear in pairs. The method has the following advantage over the modification given by Sisser.

(a) The transformations (4.2) for converting the given dyads into symmetric dyads is avoided. This reduces the computational cost of every iteration.

(b) The static ordering of dyads (4.4) is eliminated. This further reduces the computational cost of every iteration.

(c) The calculation of a perturbation parameter (4.10,4.11) is avoided. This reduces the computational costs of iterations at which nonpositive definite updates occur.

2 This thesis provides a proof for the structure theorem of dyadic Hessians.

3 This thesis provides a rank two static ordering scheme.

4 This thesis provides a new method for obtaining a positive definite approximation to an nonpositive definite Hessian. The new approximation appears to be good in that all the test problems were solved.

7.3 Future research directions

This section provides a guide to research projects related to the work of this thesis. There are, more or less, two main directions. The first is the investigation of questions raised, but not answered, by this thesis. The second is the investigation of new problems related to the work of this thesis. The following is a list of open questions.

1 What is the best strategy for choosing τ in the positive definite interval?

2 What is the best strategy for choosing the number $\bar{d}$ which replaces the nonpositive entries in D?

3 How can we determine whether or not the actual Hessian is positive definite when all remaining dyads cause a loss of positive definiteness?

New problems related to this thesis are discussed below.

1 Jackson and McCormick showed that the factorable function approach allows the use of higher than second order derivatives. They have a so-called polyadic structure. How can we use the special polyadic structure to modify higher order methods, e.g., Halley's method, for the solution of the minimization problem?

2 Can the factorable function approach be used to obtain efficient implementation of interior point linear programming algorithms?

3 Can the factorable function approach be used to obtain new results in sensitivity analysis?

7.4 Summary

This thesis further highlights the advantages of the factorable functions approach to nonlinear programming. While this thesis makes using Newton's method more feasible, it also raises questions which must yet be answered.

Appendix A

FSF, Monad, and Dyad

A.1 Sisser's function, $n = 2$

$f(x_1, x_2) = 3x_1^4 - 2x_1^2x_2^2 + 3x_2^4$
Starting point: $(1, 0.1)$.

A.2 Rosenbrock's cliff function, $n = 2$

$f(x_1, x_2) = ((x_1 - 2)/100)^2 - (x_1 - x_2) + \exp[20(x_1 - x_2)]$
Starting point: $(0, -1)$.

A.3 Rosenbrock's function, $n = 2$

$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$
Starting point: $(-1.2, 1)$.

j	Portion-of Original function
1	x_1
2	x_2
3	x_1^4
4	$3x_1^4$
5	x_1^2
6	x_2^2
7	$-2x_1^2$
8	$-2x_1^2x_2^2$
9	x_2^4
10	$3x_2^4$
11	$3x_1^4+(-2x_1^2x_2^2)$
12	$f(x_1,x_2)$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times2}$
2	x_2	e_2	$0_{2\times2}$
3	f_1^4	$(4f_1^3)\nabla f_1$	$(12f_1^2)\nabla f_1\nabla f_1^T$
4	$3f_3$	$(3)\nabla f_3$	$(36f_1^2)\nabla f_1\nabla f_1^T$
5	f_1^2	$(2f_1)\nabla f_1$	$(2)\nabla f_1\nabla f_1^T$
6	f_2^2	$(2f_2)\nabla f_2$	$(2)\nabla f_2\nabla f_2^T$
7	$(-2)f_5$	$(-2)\nabla f_5$	$(-4)\nabla f_1\nabla f_1^T$
8	$f_6\cdot f_7$	$f_6\nabla f_7+f_7\nabla f_6$	$(-4f_6)\nabla f_1\nabla f_1^T + (2f_7)\nabla f_2\nabla f_2^T + (\nabla f_7\nabla f_6^T+\nabla f_6\nabla f_7^T)$
9	f_2^4	$(4f_2^3)\nabla f_2$	$(12f_2^2)\nabla f_2\nabla f_2^T$
10	$3f_9$	$(3)\nabla f_9$	$(36f_2^2)\nabla f_2\nabla f_2^T$
11	f_4+f_8	$\nabla f_4+\nabla f_8$	$(36f_2^2)\nabla f_2\nabla f_2^T + (-4f_6)\nabla f_1\nabla f_1^T + (2f_7)\nabla f_2\nabla f_2^T+(\nabla f_7\nabla f_6^T+\nabla f_6\nabla f_7^T)$
12	$f_{11}+f_{10}$	$\nabla f_{11}+\nabla f_{10}$	$(36f_1^2 - 4f_6)\nabla f_1\nabla f_1^T + (36f_2^2 + 2f_7)\nabla f_2\nabla f_2^T+(\nabla f_7\nabla f_6^T+\nabla f_6\nabla f_7^T)$

Table A.1: Sisser's function, $n=2$

j	Portion of Original function
1	x_1
2	x_2
3	$x_1 - 3$
4	$(\frac{x_1-3}{100})$
5	$(\frac{x_1-3}{100})^2$
6	$-x_2$
7	$(x_1 - x_2)$
8	$-(x_1 - x_2)$
9	$(\frac{x_1-3}{100})^2 - (x_1 - x_2)$
10	$20(x_1 - x_2)$
11	$\exp[20(x_1 - x_2)]$
12	$f(x_1, x_2)$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	$f_1 - 3$	∇f_1	$0_{2\times 2}$
4	$(\frac{1}{100})f_3$	$(\frac{1}{100})\nabla f_3$	$0_{2\times 2}$
5	f_4^2	$(2f_4)\nabla f_4$	$(2)\nabla f_4 \nabla f_4^T$
6	$(-1)f_2$	$(-1)\nabla f_2$	$0_{2\times 2}$
7	$f_1 + f_6$	$\nabla f_1 + \nabla f_6$	$0_{2\times 2}$
8	$-f_7$	$(-1)\nabla f_7$	$0_{2\times 2}$
9	$f_5 + f_8$	$\nabla f_5 + \nabla f_8$	$(2)\nabla f_4 \nabla f_4^T$
10	$20 f_7$	$20\nabla f_7$	$0_{2\times 2}$
11	$\exp(f_{10})$	$(\exp(f_{10}))\nabla f_{10}$	$\exp(f_{10})\nabla f_{10} \nabla f_{10}^T$
12	$f_9 + f_{11}$	$\nabla f_9 + \nabla f_{11}$	$(2)\nabla f_4 \nabla f_4^T + (\exp(f_{10}))\nabla f_{10} \nabla f_{10}^T$

Table A.2: Rosenbrock's cliff function, $n = 2$

j	Portion of Original function
1	x_1
2	x_2
3	$1 - x_1$
4	$(1 - x_1)^2$
5	x_1^2
6	$-x_1^2$
7	$x_2 - x_1^2$
8	$(x_2 - x_1^2)^2$
9	$100(x_2 - x_1^2)^2$
10	$f(x_1, x_2)$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	$1 - f_1$	$(-1)\nabla f_1$	$0_{2\times 2}$
4	f_3^2	$(2f_3)\nabla f_3$	$(2)\nabla f_3 \nabla f_3^T$
5	f_1^2	$(2f_1)\nabla f_1$	$(2)\nabla f_1 \nabla f_1^T$
6	$-f_5$	$(-1)\nabla f_5$	$(-2)\nabla f_1 \nabla f_1^T$
7	$f_2 + f_6$	$\nabla f_2 + \nabla f_6$	$(-2)\nabla f_1 \nabla f_1^T$
8	f_7^2	$(2f_7)\nabla f_7$	$(-4f_7)\nabla f_1 \nabla f_1^T + (2)\nabla f_7 \nabla f_7^T$
9	$100f_8$	$(100)\nabla f_8$	$(-400f_7)\nabla f_1 \nabla f_1^T + (200)\nabla f_7 \nabla f_7^T$
10	$f_4 + f_9$	$\nabla f_4 + \nabla f_9$	$(-400f_7)\nabla f_1 \nabla f_1^T + (200)\nabla f_7 \nabla f_7^T + (2)\nabla f_3 \nabla f_3^T$

Table A.3: Rosenbrock's function, $n = 2$

A.4 Hyperbola-circle function, $n = 2$

$f(x_1, x_2) = (x_1 x_2 - 1)^2 + (x_1^2 + x_2^2 - 4)^2$

Starting point: $(0, 1)$.

A.5 Beale's function, $n = 2$

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

Starting point: $(1, 1)$.

j	Portion of Original function
1	x_1
2	x_2
3	x_1x_2
4	x_1x_2-1
5	$(x_1x_2-1)^2$
6	x_1^2
7	x_2^2
8	$x_1^2+x_2^2$
9	$x_1^2+x_2^2-4$
10	$(x_1^2+x_2^2-4)^2$
11	$f(x_1,x_2)$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times2}$
2	x_2	e_2	$0_{2\times2}$
3	$f_1\cdot f_2$	$f_1\nabla f_2+f_2\nabla f_1$	$\nabla f_1\nabla f_2^T+\nabla f_2\nabla f_1^T$
4	f_3-1	∇f_3	$\nabla^2 f_3$
5	f_4^2	$(2f_4)\nabla f_4$	$(2f_4)(\nabla f_1\nabla f_2^T + \nabla f_2\nabla f_1^T) + (2)\nabla f_4\nabla f_4^T$
6	f_1^2	$(2f_1)\nabla f_1$	$(2)\nabla f_1\nabla f_1^T$
7	f_2^2	$(2f_2)\nabla f_2$	$(2)\nabla f_2\nabla f_2^T$
8	f_6+f_7	$\nabla f_6+\nabla f_7$	$(2)(\nabla f_1\nabla f_1^T+\nabla f_2\nabla f_2^T)$
9	f_8-4	∇f_8	$(2)(\nabla f_1\nabla f_1^T+\nabla f_2\nabla f_2^T)$
10	f_9^2	$(2f_9)\nabla f_9$	$(4f_9)(\nabla f_1\nabla f_1^T + \nabla f_2\nabla f_2^T) + (2)\nabla f_9\nabla f_9^T$
11	f_5+f_{10}	$\nabla f_5+\nabla f_{10}$	$(4f_9)(\nabla f_1\nabla f_1^T + \nabla f_2\nabla f_2^T) + (2f_4)(\nabla f_1\nabla f_2^T + \nabla f_2\nabla f_1^T) + (2)(\nabla f_4\nabla f_4^T+\nabla f_5\nabla f_9^T)$

Table A.4: Hyperbola-circle function, $n=2$

j	Portion of Original function
1	x_1
2	x_2
3	$1 - x_2$
4	$x_1(1 - x_2)$
5	$1.5 - x_1(1 - x_2)$
6	$(1.5 - x_1(1 - x_2))^2$
7	x_2^2
8	$1 - x_2^2$
9	$x_1(1 - x_2^2)$
10	$(2.25 - x_1(1 - x_2^2))$
11	$(2.25 - x_1(1 - x_2^2))^2$
12	x_2^3
13	$(1 - x_2^3)$
14	$x_1(1 - x_2^3)$
15	$2.625 - x_1(1 - x_2^3)$
16	$(2.625 - x_1(1 - x_2^3))^2$
17	$(1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2$
18	$f(x_1; x_2)$

Table A.5a: Beale's function, $n = 2$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	$1 - f_2$	$(-1.0)\cdot\nabla f_2$	$0_{2\times 2}$
4	$f_1 \cdot f_3$	$f_1\nabla f_3 + f_3\nabla f_1$	$\nabla f_1\nabla f_3^T + \nabla f_3\nabla f_1^T$
5	$1.5 - f_4$	$(-1.0)\nabla f_4$	$(-1)(\nabla f_1\nabla f_3^T + \nabla f_3\nabla f_1^T)$
6	f_5^2	$(2f_5)\nabla f_5$	$(-2f_5)(\nabla f_1\nabla f_3^T + \nabla f_3\nabla f_1^T) + (2)\nabla f_5\nabla f_5^T$
7	f_2^2	$(2)f_2\nabla f_2$	$(2)\nabla f_2\nabla f_2^T$
8	$1 - f_7$	$(-1.0)\nabla f_7$	$(-2)\nabla f_2\nabla f_2^T$
9	$f_1 \cdot f_8$	$f_1\nabla f_8 + f_8\nabla f_1$	$(-2f_1)\nabla f_2\nabla f_2^T + \nabla f_1\nabla f_8^T + \nabla f_8\nabla f_1^T$
10	$2.25 - f_9$	$(-1.0)\nabla f_9$	$(2f_1)\nabla f_2\nabla f_2^T + (-1)(\nabla f_1\nabla f_8^T + \nabla f_8\nabla f_1^T)$
11	f_{10}^2	$(2f_{10})\nabla f_{10}$	$(2)\nabla f_{10}\nabla f_{10}^T + (4f_1f_{10})\nabla f_2\nabla f_2^T + (-2f_{10})(\nabla f_1\nabla f_8^T + \nabla f_8\nabla f_1^T)$
12	f_2^3	$(3f_2^2)\nabla f_2$	$(6f_2)\nabla f_2\nabla f_2^T$
13	$1 - f_{12}$	$(-1.0)\nabla f_{12}$	$(-6f_2)\nabla f_2\nabla f_2^T$
14	$f_1 \cdot f_{13}$	$f_1\nabla f_{13} + f_{13}\nabla f_1$	$(\nabla f_{13}\nabla f_1^T + \nabla f_1\nabla f_{13}^T) + (-6f_1f_2)\nabla f_2\nabla f_2^T$
15	$2.625 - f_{14}$	$(-1.0)\nabla f_{14}$	$(-1.0)(\nabla f_{13}\nabla f_1^T + \nabla f_1\nabla f_{13}^T) + (6f_1f_2)\nabla f_2\nabla f_2^T$
16	f_{15}^2	$(2f_{15})\nabla f_{15}$	$(-2f_1 5)(\nabla f_{13}\nabla f_1^T + \nabla f_1\nabla f_{13}^T) + (12f_1f_2f_{15})\nabla f_2\nabla f_2^T + (2)\nabla f_{15}\nabla f_{15}^T$
17	$f_6 + f_{11}$	$\nabla f_6 + \nabla f_{11}$	$(-2f_5)(\nabla f_1\nabla f_3^T + \nabla f_3\nabla f_1^T) + (2)\nabla f_5\nabla f_5^T + (2)\nabla f_{10}\nabla f_{10}^T + (4f_1f_{10})\nabla f_2\nabla f_2^T + (-2f_{10})(\nabla f_1\nabla f_8^T + \nabla f_8\nabla f_1^T)$
18	$f_{17} + f_{16}$	$\nabla f_{17} + \nabla f_{16}$	$(-2f_5)(\nabla f_1\nabla f_3^T + \nabla f_3\nabla f_1^T) + (2)(\nabla f_5\nabla f_5^T + \nabla f_{10}\nabla f_{10}^T) + (4f_1f_{10} + 12f_1f_2f_{15})\nabla f_2\nabla f_2^T + (-2f_{10})(\nabla f_1\nabla f_8^T + \nabla f_8\nabla f_1^T) + (-2f_{15})(\nabla f_{13}\nabla f_1^T + \nabla f_1\nabla f_{13}^T) + (2)\nabla f_{15}\nabla f_{15}^T$

Table A.5b: Beale's function, $n = 2$

j	Portion of Original function
1	x_1
2	x_2
3	$3x_2$
4	$x_1 + 3x_2$
5	$1 - x_1$
6	$(x_1 + 3x_2)(1 - x_1)$
7	$-0.1136(x_1 + 3x_2)(1 - x_1)$
8	$x_1 - 0.1136(x_1 + 3x_2)(1 - x_1)$
9	$[x_1 - 0.1136(x_1 + 3x_2)(1 - x_1)]^2$
10	$2x_1$
11	$-x_2$
12	$(2x_1 - x_2)$
13	$(1 - x_2)$
14	$(2x_1 - x_2)(1 - x_2)$
15	$7.5(2x_1 - x_2)(1 - x_2)$
16	$x_2 + 7.5(2x_1 - x_2)(1 - x_2)$
17	$[x_2 + 7.5(2x_1 - x_2)(1 - x_2)]^2$
18	$f(x_1, x_2)$

Table A.6a: Gottfriend's function, $n = 2$

A.6 Gottfriend's function, $n = 2$

$f(x_1, x_2) = [x_1 - 0.1136(x_1 + 3x_2)(1 - x_1)]^2 + [x_2 + 7.5(2x_1 - x_2)(1 - x_2)]^2$
Starting point: (0.5, 0.5).

A.7 Powell's badly scaled function, $n = 2$

$f(x_1, x_2) = (10^4 x_1 x_2 - 1)^2 + (\exp(-x_1) + \exp(-x_2) - 1.00001)^2$
Starting point: (0, 1).

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	$3f_2$	$(3)\nabla f_2$	$0_{2\times 2}$
4	$f_1 + f_3$	$\nabla f_1 + \nabla f_3$	$0_{2\times 2}$
5	$1 - f_1$	$-\nabla f_1$	$0_{2\times 2}$
6	$f_4 \cdot f_5$	$f_4 \nabla f_5 + f_5 \nabla f_4$	$(\nabla f_4 \nabla f_5^T + \nabla f_5 \nabla f_4^T)$
7	$-0.1136 f_6$	$(-0.1136)\nabla f_6$	$(-0.1136)(\nabla f_4 \nabla f_5^T + \nabla f_5 \nabla f_4^T)$
8	$f_1 + f_7$	$\nabla f_1 + \nabla f_7$	$(-0.1136)(\nabla f_4 \nabla f_5^T + \nabla f_5 \nabla f_4^T)$
9	f_8^2	$(2f_8)\nabla f_8$	$(-0.2272 f_8)(\nabla f_4 \nabla f_5^T + \nabla f_5 \nabla f_4^T) + (2)\nabla f_8 \nabla f_8^T$
10	$(2)f_1$	$(2)\nabla f_1$	$0_{2\times 2}$
11	$-f_2$	$-\nabla f_2$	$0_{2\times 2}$
12	$f_{10} + f_{11}$	$\nabla f_{10} + \nabla f_{11}$	$0_{2\times 2}$
13	$1 + f_{11}$	∇f_{11}	$0_{2\times 2}$
14	$f_{12} \cdot f_{13}$	$f_{13} \nabla f_{13} + f_{12} \nabla f_{13}$	$(\nabla f_{12} \nabla f_{13}^T + \nabla f_{13} \nabla f_{12}^T)$
15	$7.5 f_{14}$	$(7.5)\nabla f_{14}$	$(7.5)(\nabla f_{12} \nabla f_{13}^T + \nabla f_{13} \nabla f_{12}^T)$
16	$f_2 + f_{15}$	$\nabla f_2 + \nabla f_{15}$	$(7.5)(\nabla f_{12} \nabla f_{13}^T + \nabla f_{13} \nabla f_{12}^T)$
17	f_{16}^2	$(2f_{16})\nabla f_{16}$	$(15 f_{16})(\nabla f_{12} \nabla f_{13}^T + \nabla f_{13} \nabla f_{12}^T) + (2)\nabla f_{16} \nabla f_{16}^T$
18	$f_9 + f_{17}$	$\nabla f_9 + \nabla f_{17}$	$(-0.2272 f_8)(\nabla f_4 \nabla f_5^T + \nabla f_5 \nabla f_4^T) + (15 f_{16})(\nabla f_{12}$ $\nabla f_{13}^T + \nabla f_{13} \nabla f_{12}^T) + (2)(\nabla f_8 \nabla f_8^T + \nabla f_{16} \nabla f_{16}^T)$

Table A.6b: Gottfriend's function, $n = 2$

j	Portion of Original function
1	x_1
2	x_2
3	$x_1 x_2$
4	$10^4 x_1 x_2$
5	$10^4 x_1 x_2 - 1$
6	$(10^4 x_1 x_2 - 1)^2$
7	e^{-x_1}
8	e^{-x_2}
9	$e^{-x_1} + e^{-x_2}$
10	$e^{-x_1} + e^{-x_2} - 1.0001$
11	$(e^{-x_1} + e^{-x_2} - 1.0001)^2$
12	$f(x_1, x_2)$

j	f_j	∇f_j	$\nabla^2 f_j$
1	x_1	e_1	$0_{2\times 2}$
2	x_2	e_2	$0_{2\times 2}$
3	$f_1 \cdot f_2$	$f_1 \nabla f_2 + f_2 \nabla f_1$	$\nabla f_1 \nabla f_2^T + \nabla f_2 + g f 1^T$
4	$10^4 f_3$	$(10^4)\nabla f_3$	$10^4(\nabla f_1 \nabla f_2^T + \nabla f_2 \nabla f_1^T)$
5	$f_4 - 1$	∇f_4	$10^4(\nabla f_1 \nabla f_2^T + \nabla f_2 \nabla f_1^T)$
6	f_5^2	$(2f_5)\nabla f_5$	$(2 \times 10^4 f_5)(\nabla f_1 \nabla f_2^T + \nabla f_2 \nabla f_1^T) + (2)\nabla f_5 \nabla f_5^T$
7	e^{-f_1}	$(-e^{-f_1})\nabla f_1$	$e^{-f_1}\nabla f_1 \nabla f_1^T$
8	e^{-f_2}	$(-e^{-f_2})\nabla f_2$	$e^{-f_2}\nabla f_2 \nabla f_2^T$
9	$f_7 + f_8$	$\nabla f_7 + \nabla f_8$	$e^{-f_1}\nabla f_1 \nabla f_1^T + e^{-f_2}\nabla f_2 \nabla f_2^T$
10	$f_9 - 1.0001$	∇f_9	$e^{-f_1}\nabla f_1 \nabla f_1^T + e^{-f_2}\nabla f_2 \nabla f_2^T$
11	f_{10}^2	$(2f_{10})\nabla f_{10}$	$(2f_{10}e^{-f_1})\nabla f_1 \nabla f_1^T + (2f_{10}e^{-f_2})\nabla f_2 \nabla f_2^T + (2)\nabla f_{10} \nabla f_{10}^T$
12	$f_6 + f_{11}$	$\nabla f_6 + \nabla f_{11}$	$(2 \times 10^4 f_5)(\nabla f_1 \nabla f_2^T + \nabla f_2 \nabla f_1^T) + (2)\nabla f_5 \nabla f_5^T + (2f_{10}e^{-f_1})\nabla f_1 \nabla f_1^T + (2f_{10}e^{-f_2})\nabla f_2 \nabla f_2^T + (2)\nabla f_{10} \nabla f_{10}^T$

Table A.7: Powell's badly scaled function, $n = 2$

Bibliography

[1] M.J. Best and R.J. Caron [1985]. "A parameterized Hessian Quadratic Programming Problem", Annals of Operation Researches, pp. 373-394.

[2] C.G. Broyden [1970]. "The Convergence of a Class of Double Rank Minimization Algorithm, 2, The new Algorithm", Journal of the Institute of Mathematics and Its Application, Vol. 6, pp. 222-2311.

[3] R.J. Caron and N.I.M. Gould [1986]. "Finding a positive Semidefinite Interval for A parametric Matrix", Linear Algebra, Vol. 76, pp. 19-29.

[4] J. E. Dennis, JR. and J. J. Moré [1977]. "Quasi-Newton Methods, Motivation and Theory", SIAM Preview, Vol.19, No.1, pp.46-89.

[5] J. E. Dennis, JR. and R. B. Schnabel [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, New Jersey.

[6] A. V. Fiacco and G. P. McCormick [1968]. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York.

[7] R. Fletcher [1970]. "A New Approach to Variable - Metric Algorithms", Computer Journal, Vol. 13, pp. 317-322.

[8] L. Fratta, M. Gerla, and L. Kleinrock [1973]. "The Flow Deviation Method: An Approach to Store and Forward Communication Network Design", John Wiley & Sons, New York.

[9] A. Ghaemi and G. P. McCormick [1979]. "Factorable Symbolic SUMT: What is it? How is it used?", The George Washington University, Washington, D. C., Institute for Management Science and Engineering, Technical Paper Serial T-402.

[10] P.E. Gill and W. Murray. [1974]. "Newton-Type Methods for Unconstrained and Linearly Constrained Optimization", Mathematical Programming, Vol. 7, pp. 311-350.

[11] P. E. Gill, W. Murray, and M. H. Wright [1981]. *Practical Optimization*, Academic Press, New York.

[12] R. H. F. Jackson and G. P. McCormick [1986]. "The Polyadic Structure of Factorable Function Tensors with Application to Higher-Order Minimization Techniques", Journal of Optimization Theory and Applications, Vol.15, No.1, pp.63-94.

[13] R. H. F. Jackson and G. P. McCormick [1988]. "Second-Order Sensitivity Analysis in Factorable Programming", Mathematical Programming, Vol.41, pp.1- 27.

[14] G. P. McCormick [1983]. *Nonlinear programming: Theory, Algorithm, and Application*, John Wiley & Sons, New York.

[15] W. C. Mylander, R. Holmes, and G. P. McCormick [1971]. "A Guide to SUMT- VERSION 4: The Computer Program Implementing The Sequential Unconstrained Minimization Technique for Nonlinear Programming", RAC-P-63, Mclean, Virgina.

[16] J. M. Ortega and W. C. Rheinboldt [1970]. *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.

[17] R. E. Pugh [1972]. "A Language for Nonlinear Programming Problems", Mathematical Programming, Vol.2, pp.176-206.

[18] J. Sherman and W. J. Morrison [1949]. "Adjustment of An Inverse Matrix Corresponding to A Change in One Element of A Given Matrix", Annals of Mathematical Statistics, Vol.20, pp.124-127.

[19] D. S. Shuzhi [1987]. *Numerical Algebra*, Zhing Wah University Press, China, Chinese edition.

[20] F. S. Sisser [1982]. "A Modified Newton's Method for Minimizing Factorable Functions", Journal of Optimization Theory and Applications, Vol.38, No.4, pp. 461-482.

[21] D.C. Sorensen [1977]. "Updating the Symmetric Indefinite Factorization with Applications in a Modified Newton's Method". Argonne National Laboratory, Report No. ANL-77-49.

[22] M. Woodbury [1950]. "Inverting Modified Matrices", Princeton University, Princeton, New Jersey, Statistical Research Group, Memorandum, No.42.

Vita Auctoris

The author was born in Hong Kong on November 15, 1962.

He received his high school diploma from the Victoria Park Secondary School of Toronto in 1983.

He received his B.Cs. in Honours Computer Science from the University of Windsor in 1987.