Toward efficient maximum likelihood algorithms

Tae-Sung Shin
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Computer Sciences Commons, and the Statistics and Probability Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/11809

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Toward efficient maximum likelihood algorithms

by

Tae-Sung Shin

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Statistics
Major Professor: Mervyn G. Marasinghe

Iowa State University
Ames, Iowa
1998

Copyright © Tae-Sung Shin, 1998. All rights reserved.
Graduate College
Iowa State University

This is to certify that the Ph.D. dissertation of

Tae-Sung Shin

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ................................................. x

1 INTRODUCTION ..................................................... 1

2 OVERVIEW of MAXIMUM LIKELIHOOD ALGORITHMS ........ 4
   2.1 Introduction .................................................. 4
   2.2 Approximations to the Likelihood Function ............ 8
      2.2.1 Quadratic Approximation: Newton-type Algorithms 8
      2.2.2 Complete-data Likelihood Approximations: EM-type Algorithms 11
      2.2.3 Other approximations .................................. 14
   2.3 The Extensions of Numerical Linear System Solvers to ML Algorithms ........................................ 15
      2.3.1 Numerical Linear System Solvers ....................... 15
      2.3.2 Use of Linear System Solvers in ML Algorithms .... 17
   2.4 Generalized Conditional Maximization Algorithms .... 22

3 LINEARIZATION OF LIKELIHOOD EQUATIONS ................. 25
   3.1 Introduction .................................................. 25
   3.2 Theory of Univariate Linearization .................... 26
   3.3 Selection Procedure for Univariate Linearization .. 29
   3.4 Univariate Linearizations Examples .................... 30
      3.4.1 Univariate t-model with Unknown Degrees of Freedom 30
      3.4.2 Two Component Finite Normal Mixture Model ......... 35
<table>
<thead>
<tr>
<th>3.4.3 Neural Network Prediction Model</th>
<th>39</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4 DESIGN OF A GCM ALGORITHM</strong></td>
<td>44</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>44</td>
</tr>
<tr>
<td>4.2 Review of Conventional Algorithms</td>
<td>45</td>
</tr>
<tr>
<td>4.3 Partition and Reparameterization</td>
<td>46</td>
</tr>
<tr>
<td>4.4 Selection of an Approximation Function</td>
<td>47</td>
</tr>
<tr>
<td>4.5 Decision for R-step</td>
<td>47</td>
</tr>
<tr>
<td>4.6 Numerical Examples</td>
<td>48</td>
</tr>
<tr>
<td>4.6.1 Random Effects Model with overrelaxation</td>
<td>49</td>
</tr>
<tr>
<td>4.6.2 Variance Components Model (Two components)</td>
<td>54</td>
</tr>
<tr>
<td>4.6.3 Contingency Table with Incomplete Cells</td>
<td>59</td>
</tr>
<tr>
<td>4.6.4 Univariate T-model with Unknown Degrees of Freedom</td>
<td>62</td>
</tr>
<tr>
<td>4.6.5 Normal Finite Mixture Model with Two components</td>
<td>64</td>
</tr>
<tr>
<td><strong>5 LINEAR VARIANCE COMPONENT MODEL</strong></td>
<td>70</td>
</tr>
<tr>
<td>5.1 Conventional Methods for Variance Component Model</td>
<td>70</td>
</tr>
<tr>
<td>5.2 New GCM Algorithms for Variance Component Estimation</td>
<td>78</td>
</tr>
<tr>
<td>5.2.1 Computation of First and Second Derivatives of Loglikelihood</td>
<td>83</td>
</tr>
<tr>
<td>5.3 Empirical Comparison of ML algorithms</td>
<td>84</td>
</tr>
<tr>
<td>5.4 New GCM Algorithms with True Missing Data in y</td>
<td>91</td>
</tr>
<tr>
<td>5.4.1 Simple Example for Missing Data in y</td>
<td>96</td>
</tr>
<tr>
<td>5.5 Discussion</td>
<td>99</td>
</tr>
<tr>
<td><strong>6 FRAMEWORK FOR MORE APPLICATIONS OF THE GCM ALGORITHMS FOR FUTURE STUDY</strong></td>
<td>100</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>100</td>
</tr>
<tr>
<td>6.2 New GCM Algorithms with Missing Data in X for Variance Component Model</td>
<td>100</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 4.1 Comparisons of the number of iterations among GCM-1, GCM-2, GCM-3, and ECME .................................................. 58
Table 4.2 A 2 x 2 x 2 contingency table with partially classified observations from Little and Rubin(1987, p. 187) ............................. 59
Table 4.3 Comparisons of the number of iterations and CPU time per iteration among ECM, ERCM, and ERCM with overrelaxation (ERCMO) ................................................................. 61
Table 4.4 Comparisons of the number of iterations and total CPU Time among GCM-1, GCM-2, and ECME ............................ 65
Table 4.5 Comparisons of the number of iterations among GCM and EM ...................................................... 68
Table 5.1 Total CPU times (number of iterations) of the four algorithms converging to \( \hat{\sigma}_0 = 0.0386969, \hat{\sigma}_1 = 0.0088309 \) in the first example (total CPU time = 0 (0) implies divergence of the algorithm) 88
Table 5.2 Total CPU times (number of iterations) of the four algorithms converging to \( \hat{\sigma}_0 = 2.9440619, \hat{\sigma}_1 = 0 \) in the second example (total CPU time (number of iteration) = 0 (0) implies divergence of the algorithm) ................................. 89
Table 5.3  Total CPU times (number of iterations) of the four algorithms converging to $\hat{\sigma}_0 = 137.3160997$, $\hat{\sigma}_1 = 121.0584859$ in the third example (total CPU time (number of iteration) = 0 (0) implies divergence of the algorithm) ........................................ 90

Table 5.4  The results of ML estimation for example of missing data in $y$ with Battese et al’s(1988) model .................................................. 97
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Original Loglikelihood, Its First Derivative, and Its Second Derivative Functions as a Function of Degrees of Freedom $v$</td>
</tr>
<tr>
<td>3.2</td>
<td>Original Loglikelihood, Its Linearized Score, and Its Linearized Hessian Functions as a Function of Degrees of Freedom $v$ with $p=2.6$</td>
</tr>
<tr>
<td>3.3</td>
<td>Original Loglikelihood, Its Linearized Score, and Its Linearized Hessian Functions as a Function of Degrees of Freedom $v$ with $p=3.0$</td>
</tr>
<tr>
<td>3.4</td>
<td>The Surfaces of Loglikelihood, First, and Second Derivative as a Function of $p$ for Various True $p = 0.2, 0.4, 0.7, \text{ and } 0.9$</td>
</tr>
<tr>
<td>3.5</td>
<td>The Surfaces of Loglikelihood, First, Second Derivative (Thick Lines) and Modified Hessian Function (Thin Lines) as a Function of $\tau$ for Various True $p = 0.2, 0.4, 0.7, \text{ and } 0.9$</td>
</tr>
<tr>
<td>3.6</td>
<td>The Surfaces of Loglikelihood, Linearized Score, and Linearized Hessian as a Function of $\alpha = p/(1 - p)$ for Various True $p = 0.2, 0.4, 0.7, \text{ and } 0.9$</td>
</tr>
<tr>
<td>3.7</td>
<td>The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $w_{h1}$ for Various Seeds</td>
</tr>
<tr>
<td>3.8</td>
<td>The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $\pi_{h1} = \exp(w_{h1})$ for Various Seeds</td>
</tr>
</tbody>
</table>
Figure 3.9 The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $\pi_{h1} = \exp(w_{h1})$ for Various Seeds ... 43

Figure 4.1 Simulation Study 1 $\log(P X - EM)$ and $\log(P X - EM - OR)$ Indicate the Log-scaled Number of Iterations of Pure PX-EM and PX-EM with Overrelaxation, Respectively. the Solid Line Denotes the Line $\log(P X - EM) = \log(P X - EM - OR)$ ... 55

Figure 4.2 The Surfaces of Loglikelihood, First, Second Derivative (Thick Lines) and Negative Hessian Function (Thin Lines) in Terms of $\tau$ for Various True $p = 0.2, 0.4, 0.7, \text{and} 0.9$ ... 66

Figure 4.3 Comparisons of Log-scaled Number of Iterations between EM and GCM with 100 Random Data Sets ... 69

Figure 5.1 Original Loglikelihood, Its First Derivative, and its Second Derivative Functions in Terms of $\sigma^2$ ... 80

Figure 5.2 Original Loglikelihood, Its Linearized Score Function, and Its Hessian Functions with Linearization Factor $\sigma^2_i/q_i$ in Terms of $\sigma^2_i$ 81

Figure 5.3 Scatter Plot Matrix of Satellite Data, var 0 = v, var 1 = 'Segment', var 2 = y, var 3 = $x_1$, and var 4= $x_2$ ... 98
ACKNOWLEDGEMENTS

I would like to thank to my parents, friends, and major professor for giving me this opportunity.

My parents, Ki-Hwan Shin and Kwi-Ja Kang, have always given me a motivation and prayed for me with their trust and love.

My friends in Korea and in Ames have understood, advised, and encouraged me to finish this dissertation. Especially, comments and encouragement from Dr. Chung Yeol Shin. Dr. Sung Soo Kim, and Mr. Jae Kwang Kim were gratefully helpful in preparing this dissertation. And Ms. Jeong Si Kim in Korea has given me great motivation with her love since I started this dissertation.

I also appreciate all corrections and comments from my committee who consists of Dr. Hal Stern, Dr. John Stufken, Dr. Soumendra Lahiri, and Dr. Sung Song.

Finally, thanks to all friends in Computing Section, Mrs. Kathy Shelley, Mrs. Darlene Wicks, Mr. Charles Peterson, Mrs. Ho H Liu, Mrs. Shiaau-Er Huarng, and Ms. Deanne Reber for their warm friendship.

I can’t list all my friends and what they did to me in this page. But I know that I have been so lucky to have all of them.
Toward efficient maximum likelihood algorithms

Tae-Sung Shin

Major Professor: Mervyn G. Marasinghe
Iowa State University

Motivated by recent extensive studies of maximum likelihood (ML) algorithms, especially EM-type schemes, the author proposes a class of generalized conditional maximization (GCM) algorithms that pursues dimension reduction as well as stability of algorithm simultaneously. This model-dependent approach for developing ML algorithms is to apply an appropriate, but possibly different approximation to each selected subset of parameters that ensure fast and stable convergence to a candidate for a local maximum. In the first part of this dissertation, the author illustrates the application of this algorithm to several examples - random effects model, variance components, normal finite mixture, t-distribution model, contingency table, and compares the performance of each to conventional EM-type algorithms using numerical studies. For the rest of this dissertation, new models emphasizing variance components model, which might be helpful in a data analysis, are studied and new GCM algorithms for those ML estimations are developed.
1 INTRODUCTION

The development of efficient algorithms for Maximum likelihood (ML) estimation is considered in this dissertation. While the statistical aspect of this topic may appear to be insignificant since the subjects related to ML algorithms we are concerned with (e.g., convergence rate, computation time, and monotone convergence) may seem to be only of interest to numerical analysts, the impact of this study might be quite significant in the areas of statistics that introduce new methodology which requires the development of new and more efficient algorithms for parameter estimation.

For example, an industrial company which has giga-capacity of database related to its products might want to analyze various complex modeling procedures using data from the database to improve the quality of its products. When a statistician in the company attempts to use ML procedures which is one of the best estimation methods, she will soon discover that it takes double-digit hours to obtain ML estimators. In the worst case, she might not be able to do this without simplifying the model used in the analysis. This is a situation where the availability of algorithms for reducing the computation time for MLE may lead to better quality of data analysis.

In reviewing the literature on ML algorithms, we have found that various techniques that are available in the numerical analysis literature on optimization have been adopted for use in statistical research without adequate knowledge or experience regarding the actual performance of such methods in the types of problems to which such techniques are applied. For example, the Newton-GS algorithm incorporates Gauss-Seidel type iterations for the solution of equations, into a standard optimization technique (see
Ortega and Rheinboldt(1970)). However, no information is available how this approach will work in ML estimation problems. In Chapter 2, we will outline how this method and many other numerical approaches form the basis for many ML algorithms introduced during the past decade.

In this dissertation we will present a unified approach to ML algorithms that enables the user to investigate, and incorporate if desired, all available options for increasing the efficiency of a specified ML problem. This new approach, called the 'Generalized Conditional Maximization (GCM) Algorithms', allows the application of conventional algorithms to reduced dimensions, with the objective of increasing speed of convergence while maintaining the desirable property of monotone convergence.

A conventional algorithm that is of primary importance and one that plays a leading role in many GCM algorithms, is Newton's method. However, Newton-type algorithms are well-known for their failure of monotone convergence. Since one of our main concerns is speed of convergence, we investigate a technique to improve this aspect of Newton-type algorithms. The ability of Newton-type algorithms to converge depends on the quadratic nature of the likelihood surface near the optimum. Our approach is an extension of an idea introduced by Callanan and Harville(1991) where the an attempt is made to 'linearize' the first derivative of the loglikelihood function.

In Chapter 4, we present the main features involved in the design of a GCM algorithm for a specific model by following the development of the algorithm for an example. The model we use for this purpose is the variance components model, and the reader is taken through the various stages of development of a GCM algorithm. In later sections of that chapter, GCM algorithms are developed for several other model situations and compared numerically with conventional algorithms available for them.

The dissertation is organized as follows:

- Chapter 2
The conventional ML algorithms and the use of iterative linear system solvers for ML algorithms are reviewed.

Generalized conditional maximization algorithms, in which it is possible to apply the methods overviewed above, is defined.

• Chapter 3

Univariate linearization theorem, that guarantees monotone convergence of Newton’s algorithm, is derived and applied to several MLE problems.

• Chapter 4

Considerations involved in the development of efficient algorithms using the GCM approach are discussed using an example.

New GCM algorithms are developed and are compared with conventional algorithms in five different statistical models.

• Chapter 5

Computational method using GCM algorithm for ML estimation of variance components in the general mixed model is developed.

Computational method using GCM algorithm for ML estimation of variance components in general mixed model with missing data in dependent variable is developed.

• Chapter 6

Computational method using GCM algorithm for ML estimation of variance components in general mixed model with missing data in independent variables is derived.

Computational method using GCM algorithm for ML estimation in neural network prediction model with variance component structure is derived.
2 OVERVIEW OF MAXIMUM LIKELIHOOD ALGORITHMS

2.1 Introduction

A parametric statistical model is specified with an assumed probability density \( f(y|\theta) \) where \( y \) is an \( n \times 1 \) observed data vector, and \( \theta \) is a \( p \times 1 \) unknown parameter vector in the parameter space \( \Omega \). Based on the observed data and the probability density, the classical analysis of this model involves applying statistical inference on the parameter. The density \( f(y|\theta) \) is also called the likelihood function and denoted by \( L(\theta) \) when it is considered as a function of \( \theta \) with observed data fixed. Maximum likelihood estimation (MLE) of \( \theta \) is an estimation method which finds a value of \( \theta \) that maximizes the likelihood \( L(\theta) \) or its log value \( l(\theta) = \log(L(\theta)) \). MLE has been one of the most popular tools in large sample inference due to its sound asymptotic properties, e.g., consistency, asymptotic normality, and asymptotic efficiency. Except for special probability models, however, a formula for MLE does not exist in closed form, and it should be found by an iterative method.

Maximum likelihood (ML) algorithms are iterative procedures to find a local maximizer of loglikelihood \( l(\theta) \). Each of these ML algorithms can be summarized by an iterative function \( M(\theta) \) which updates new iterate \( \theta^{(t+1)} \) from the current iterate \( \theta^{(t)} \), i.e., \( \theta^{(t+1)} = M(\theta^{(t)}) \). The performance of a ML algorithm depends on how well its iteration function is established. In the rest of this section, we shall stipulate conditions for an efficient ML algorithm in terms of the iteration function \( M(\theta) \). In the following
discussion, $M(\theta)$ and $l(\theta)$ are assumed to have continuous second derivatives.

The most important goal of an ML algorithm is to require that the sequence generated from $M(\cdot)$ converge to a local maximum or at least a stationary point if the sequence converges. That is, for any $\theta^*$ such that $\theta^* = M(\theta^*)$ the following should hold:

$$\frac{\partial l(\theta)}{\partial \theta} |_{\theta = \theta^*} = 0.$$  

Of course, this statement is based on the assumptions that any local maximum does not exist on the boundary of $\Omega$ and that there is at least one maximum of $l(\theta)$ on $\Omega$. If the above condition holds, a stationary point is a good candidate for local maximum.

The next issue is to improve the possibility that the found stationary point is a local maximum by requiring $M(\theta)$ to have the property that $l(\theta^{(t)}) \leq l(\theta^{(t+1)})$ where $\{\theta^{(t)}\}$ is the sequence produced by $M(\cdot)$. This will ensure that the sequence will not converge to a saddle point. We define this property as monotone convergence.

Third consideration would concern convergence. Although the monotone convergence property above preserves convergence of $\{l(\theta^{(t)})\}$ by assuming boundedness of $l(\theta)$, it does not imply convergence of $\{\theta^{(t)}\}$. If it is difficult to find a $\theta^{(0)}$ that leads to convergence of $\{\theta^{(t)}\}$, then the algorithm will be less preferable. For further discussion, we denote $\| \cdot \|$ as spectral matrix or vector norm, that is, $\|A\|$ is square root of the largest eigenvalue of $A^T A$. The following lemma and its proof by Ortega and Rheinboldt (1970) is useful for assessing the convergability of $\{\theta^{(t)}\}$.

**Lemma 1 (Contraction Mapping Theorem)** Suppose a continuous mapping $M(\theta)$ on $\Omega$ is differentiable and has as its derivative $DM(\theta)$. If there is a closed set $\Omega_0 \subset \Omega$ such that

$$\| DM(\theta) \| \leq \lambda < 1, \text{ for all } \theta \in \Omega_0,$$

and if $\theta \in \Omega_0$ implies $M(\theta) \in \Omega_0$, then every sequence from $M(\cdot)$ with any $\theta^{(0)} \in \Omega_0$ converges to unique $\theta^* \in \Omega_0$ such that $M(\theta^*) = \theta^*$. 


Proof By Mean Value Theorem and assumption (2.1), Lipschitz condition holds as follows:

\[ ||M(\theta^{(1)}) - M(\theta^{(2)})|| = ||DM(\theta^{(0)})(\theta^{(1)} - \theta^{(2)})|| \leq \lambda||\theta^{(1)} - \theta^{(2)}||, \]  

(2.2)

where \( \theta_0 = a\theta_1 + (1 - a)\theta_2 \) for some \( a \in [0, 1] \). This condition leads to

\[ ||\theta^{(t+1)} - \theta^{(t)}|| \leq \lambda^t||\theta^{(1)} - \theta^{(0)}||, \]  

(2.3)

and hence

\[ ||\theta^{(t+p)} - \theta^{(t)}|| \leq \frac{1}{1 - \lambda}||\theta^{(t+1)} - \theta^{(t)}||, \]  

(2.4)

\[ \leq \frac{\lambda^t}{1 - \lambda}||\theta^{(1)} - \theta^{(0)}||, \]  

(2.5)

where each of \( \theta^{(t)} \in \Omega_0, t = 0, 1, \cdots \) by the assumption of closedness of \( \Omega_0 \) for \( M(\cdot) \).

Thus, by last inequality above, \( \{\theta^{(t)}\} \) is a Cauchy sequence which implies convergence to some \( \theta^* \) and \( \theta^* \in \Omega_0 \) due to closedness of \( \Omega_0 \). This proves the existence of \( \theta^* \).

For proof of uniqueness of \( \theta^* \in \Omega_0 \), suppose that \( \theta^+ \in \Omega_0 \) is another fixed point of \( M(\theta) \). By the Lipschitz condition (2.2) above,

\[ ||\theta^* - \theta^+|| = ||M(\theta^*) - M(\theta^+)|| \leq \lambda||\theta^* - \theta^+|| \]  

(2.6)

Thus \( \lambda \geq 1 \) which leads to a contradiction and the proof is complete.

An \( M(\cdot) \) that has the property (2.2) with \( 0 < \lambda < 1 \) is said to be contractive. In fact, by having a contractive iteration function, it will be easier to analyze the speed of the algorithm as well as guarantee its convergence. \( M(\cdot) \) which has larger \( \Omega_0 \) in which \( M(\cdot) \) is contractive is more favorable and should be first considered.

Computational burden for finding maxima of \( l(\theta) \) in case of convergence is an important consideration for efficiency of ML algorithms. The amount of computing can be
divided into CPU time for computing $M(\theta)$ and the number of iterations needed for the desired convergence.

Assume that the stopping criterion for an ML algorithm is $||\theta^{(t+1)} - \theta^*|| < \epsilon$, and that $\{\theta^{(t)}\}$ converges to some fixed point $\theta^*$. Then as $t$ increases to a large value and hence $\theta^{(t)}$ become closer to $\theta^*$, Mean Value Theorem leads to

$$\theta^{(t+1)} - \theta^* \approx DM(\theta^*)(\theta^{(t)} - \theta^*).$$

Thus we can easily see that the required number of iterations to satisfy the criterion is proportional to $\log(\lambda^*) = \log(||DM(\theta^*)||)$. $\lambda^*$ and $DM(\theta^*)$ are called the global (asymptotic) rate of convergence, and the rate of convergence, respectively. Thus we can assess to the amount of computation needed with CPU time for computing $M(\theta)$ and the size of $||DM(\theta^*)||$.

As we have discussed in this section, an efficient ML algorithm is defined relatively rather than absolutely as an iterative method with $M(\theta)$ satisfying the following conditions:

\begin{itemize}
  \item [a] any $\theta^*$ such that $\theta^* = M(\theta^*)$ should imply $\frac{\partial M(\theta^*)}{\partial \theta}|_{\theta = \theta^*} = 0$,
  \item [b] there is a large closed subset $\Omega_0$ of $\Omega$ such that $||DM(\theta)|| < 1$, for all $\theta \in \Omega_0$,
    where $DM(\cdot)$ is the first derivative of $M(\cdot)$,
  \item [c] $l(\theta) \leq l(M(\theta))$ for all $\theta \in \Omega$.
  \item [d] small computation time for computing $M(\theta)$ and
  \item [e] small $\lambda^*$ where $\lambda^* = ||DM(\theta^*)||$, and $\theta^*$ is a fixed point of $M(\theta)$.
\end{itemize}
2.2 Approximations to the Likelihood Function

A practical way to construct $M(\theta)$ is to find a function $A(\theta|\theta')$ which approximates $l(\theta)$ in the neighborhood of $\theta'$, the current iterate. Of course, $A(\theta|\theta')$ must be easy to maximize, and lead $M(\theta') \equiv \arg \max_\theta A(\theta|\theta')$ to possess the properties [a]-[e] defined in Section 2.1.

2.2.1 Quadratic Approximation: Newton-type Algorithms

One popular approximation to the nonlinear function $l(\theta)$ given current iterate $\theta'$, is the quadratic approximation,

$$A_q(\theta|\theta') = l(\theta') + (\theta - \theta')^T Dl(\theta') + \frac{1}{2}(\theta - \theta')^T D^2l(\theta')(\theta - \theta'),$$

whose maximizer with respect to the first argument is

$$M_q(\theta') = \theta' - (D^2l(\theta'))^{-1} Dl(\theta'),$$

where $Dl(\cdot)$, $D^2l(\cdot)$ are first and second derivatives of $l(\cdot)$, respectively. It is more commonly known as Newton's algorithm.

The iteration function $M_q(\theta)$ yields $DM_q(\theta^*) = 0$ where $\theta^*$ is a fixed point of $M(\theta)$. This property leads to super-linear convergence which implies that

$$||\theta^{(t+1)} - \theta^*|| \leq \alpha||\theta^{(t)} - \theta^*||^2$$

for some $\alpha > 0$. (2.10)

if $\theta^{(t)}$ is in the neighborhood of $\theta^*$.

The following lemma is modified from the global convergence theorem for Newton's method by Ortega and Rheinboldt(1970).

**Lemma 2** Let $Dl(\theta)$ be continuous, differentiable, and concave on $\Omega$. And suppose that $D^2l(\theta)$ is negative definite for all $\theta \in \Omega$. Then every sequence generated from $M_q(\theta)$ with any $\theta^{(0)} \in \Omega$ converges to the unique fixed point of $M_q(\theta)$. 
In many cases, however, $D^2l(\theta)$ is not negative definite for every $\theta \in \Omega$, and when $D^2l(\theta')$ is not negative definite, the following may happen:

$$A(M(\theta')|\theta') = l(\theta') - [Dl(\theta')]^T D^2l(\theta') [Dl(\theta')]$$

$$\leq A(\theta'|\theta') = l(\theta')$$

so that convergence to a local maximum is not ensured. To rectify this, the quadratic approximation is modified by finding an appropriate negative definite Hessian matrix $H(\theta)$ to replace $D^2l(\theta')$ in (2.8), i.e.,

$$A_{qm}(\theta|\theta') = l(\theta') + (\theta - \theta')^T Dl(\theta') + \frac{1}{2}(\theta - \theta')^T H(\theta')(\theta - \theta').$$

This approach has produced many different modified algorithms known as Newton-type methods. (See Ortega and Rheinboldt(1970), and Kennedy and Gentle(1980) for details of these methods).

As Böhning and Lindsay(1988) point out, however, even if an appropriate negative definite Hessian matrix is found and the quadratic approximation to $l(\theta)$ is well-defined, monotonic convergence as defined by condition $[c]$ may not be achievable. To remedy this, a modified version of $M_q$,

$$M_{qm}(\theta') = \theta' - \alpha(H(\theta'))^{-1} Dl(\theta')$$

is commonly used. To guarantee monotonic convergence, $\alpha$ is obtained in each iteration from a linear search in order that $l(M_{qm}(\theta')) \geq l(\theta')$ holds. However, evaluating $l(\theta)$ in each iteration to obtain a suitable $\alpha$ might lead to considerable increase in computation in each iteration especially when the computational cost of $l(\theta)$ is high.

On the other hand, Böhning and Lindsay(1988) propose a monotone quadratic approximation method that avoids linear search, called the lower bound (LB) algorithm which is based on the following lemma:
Lemma 3 Suppose that there is a negative definite scalar Hessian matrix $H$ such that $H \leq D^2 l(\theta)$ for all $\theta \in \Omega$. Then the lower bound algorithm with its iteration function

$$M_{lb}(\theta) = \theta - H^{-1} Dl(\theta)$$

satisfies $[a]$, $[b]$ with $\Omega_0 = \Omega$, $[c]$ of the conditions stipulated for an efficient ML algorithm, and its asymptotic rate of convergence is $||I - H^{-1} Dl(\theta^*)|| < 1$.

By the Lemma above, it is not necessary for LB algorithm to perform linear search in each iteration. However, if $l(\theta)$ is not well-approximated by a quadratic function, i.e., $Dl(\theta)$ is not approximately linear, and $D^2 l(\theta')$ can not be bounded below; hence it is not possible to obtain monotonic sequence by this strategy. Furthermore, if $H$ is much different from $D^2 l(\theta^*)$, the algorithm becomes intolerably slow.

The following algorithm and theorem are extended from the lower bound algorithm. Each $(t + 1)$th iteration of Extended Lower-Bound QA (ELBQA) Algorithm is defined by

$$\theta^{(t+1)} = \theta^{(t)} - B^{-1}(\theta^{(t)}) Dl(\theta^{(t)}),$$

where $B(\theta^{(t)})$ is a symmetric, negative definite matrix such that $B(\theta^{(t)}) \leq D^2 l(\theta)$ for all $\theta \in \{\theta'; D^2 l(\theta')$ is negative definite} belonging to the multivariate interval $(\theta^{(t)}, \theta^{(t+1)})$.

The multivariate interval is defined as the following:

$$\{\theta^{(t)}, \theta^{(t+1)}\} \equiv \{\theta; \theta = \gamma \theta^{(t)} + (1 - \gamma) \theta^{(t+1)} \mbox{ for some } \gamma, 0 < \gamma < 1\}.$$

Theorem 1 Let $\{\theta^{(t)}\}$ denote the sequence produced by ELBQA algorithm in (2.15). Then the sequence $\{\theta^{(t)}\}$ is a monotonely increasing sequence in the sense that $l(\theta^{(t)}) \leq l(\theta^{(t+1)})$, guaranteed to converge to $\hat{\theta}$ such that $Dl(\hat{\theta}) = 0$ if $l(\cdot)$ is bounded above, and has its convergence rate

$$||I - B^{-1}(\hat{\theta}) D^2 l(\hat{\theta})|| < 1$$

Proof of the above theorem follows in the same lines as that of theorem of lower-bound algorithm in Böhning and Lindsay(1988).
Corollary 1 Let $\theta$ be one dimensional and $\hat{\theta}$ a local maximum. If $D^2 l(\theta)$ is negative and strictly decreasing function on segment $\Theta^+ \equiv (\hat{\theta}, \infty)$, or if $D^2 l(\theta)$ is negative and strictly increasing function on segment $\Theta^+ \equiv (-\infty, \hat{\theta})$, then Newton-Raphson algorithm with $\theta^{(0)} \in \Theta^+$ is an ELBQA algorithm.

Since the quadratic approximation may run into problems when the loglikelihood surface is markedly non-quadratic, one intuitive solution is the linearization of $Dl(\theta)$ which induces a well-defined quadratic approximation of $l(\theta)$. If a positive definite block diagonal matrix $K(\theta)$ exists such that $Dl_L(\theta) = K(\theta)Dl(\theta)$ has negative definite slope which is almost constant for all $\theta$, i.e.,

$$Dl_L(\theta) = K(\theta)Dl(\theta) \simeq A\theta + b$$  \hspace{1cm} (2.17)

for some negative definite matrix $A$ and a vector $b$, the conditions [a]-[e] for an efficient ML algorithms are easily verified to hold. Formally, assuming existence of such a $K(\theta)$, linearized quadratic approximations is

$$A_{Lq}(\theta|\theta') = l(\theta') + (\theta - \theta')^T Dl_L(\theta') + \frac{1}{2}(\theta - \theta')^T D^2 l_L(\theta')(\theta - \theta'),$$  \hspace{1cm} (2.18)

which leads to

$$M_{Lq}(\theta') = \theta' - (D^2 l_L(\theta'))^{-1} Dl_L(\theta'),$$  \hspace{1cm} (2.19)

where $D^2 l_L(\cdot)$ is the first derivative of $Dl_L(\cdot)$ above. This linearization may be viewed as a way to reparameterize $\theta$ to a parameterization $\phi = \phi(\theta)$ if we set $K(\theta) = \frac{\partial \phi}{\partial \theta}$, where $\phi(\theta)$ is one-to-one function. The linearization factor $K(\theta)$ is usually found by inspection of the algebraic form of $Dl(\cdot)$, reparameterization, or by using graphical methods in low dimensions. A detailed discussion of this approach is presented in Chapter 2.

2.2.2 Complete-data Likelihood Approximations: EM-type Algorithms

Another popular ML algorithm in statistics is the Expectation-Maximization(EM) algorithm in which, assuming that missing data is random, $l(\theta)$ is approximated by the
conditional expectation of complete data (missing data + observed data) loglikelihood given observed data and the current iterate $\theta'$.

Define $x \in X$ to be complete data, some part of which, $y$ is observed, $f_c(x|\theta)$ and $f_o(y|\theta)$ to be probability density functions of complete data and observed data, respectively. This is viewed as a missing data problem in which we do not observe complete data $x$ directly but indirectly through $X(y)$, the set of all possible $x$'s with observed data $y$. The goal is to find the maximum point of observed-data loglikelihood,

$$l(\theta) = \log f_o(y|\theta) \equiv \log \int_{X(y)} f_c(x|\theta) dx$$

(2.20)

In order to construct an iteration function, Dempster et al(1977) introduce an approximation function

$$A_{EM}(\theta|\theta') = \int_{X(y)} (\log f_c(x|\theta)) \frac{f_c(x|\theta')}{f_o(y|\theta')} dx = E(\log f_c(x|\theta)|y, \theta').$$

(2.21)

and prove the following lemma:

Lemma 4 Any $M(\theta')$ such that $A_{EM}(M(\theta')|\theta') \geq A_{EM}(\theta'|\theta')$ implies that $l(M(\theta')) \geq l(\theta')$ for all $\theta' \in \Omega$.

The lemma above can be proved by showing that

$$R_{EM}(\theta|\theta') \equiv A_{EM}(\theta|\theta') - l(\theta) \leq R_{EM}(\theta'|\theta'),$$

(2.22)

that is,

$$D^{10}R_{EM}(\theta|\theta) = D^{01}R_{EM}(\theta|\theta) = 0,$$

(2.23)

and both of $D^{20}R_{EM}(\theta|\theta)$ and $D^{02}R_{EM}(\theta|\theta)$ are negative semi-definite, where $D^{ij}f(a, b)$ denotes $i$th and $j$th derivative of $f$ for first and second argument, respectively. By the monotonic property above, the convergence of a sequence $\{l(\theta^{(i)})\}$ from the EM algorithm is guaranteed under the assumption of boundedness of $l(\theta)$. 
If there exist \( K(\theta), T_1(x), \) and \( t_2(x) \) such that
\[
K(\theta) \frac{\partial}{\partial \theta} \log f_c(x|\theta) = T_1(x)\theta - t_2(x), \tag{2.24}
\]
where \( K(\theta) \) is a positive definite matrix, \( T_1(x) \) is a nonsingular matrix, and \( t_2(x) \) is a vector, the iteration function for the EM algorithm
\[
M_{EM}(\theta') = \theta' - [D^{20} A_{EML}(\theta'|\theta')]^{-1} D^{10} A_{EML}(\theta'|\theta')
= [E(T_1(x)|y, \theta')]^{-1} E(t_2(x)|y, \theta')
\]
has a unique closed form for maximizing (2.21) and is the iteration function of the EM algorithm, where
\[
D^{10} A_{EML}(\theta|\theta') = K(\theta) D^{10} A_{EM}(\theta|\theta')
\]
and \( D^{20} A_{EML}(\theta|\theta') \) is the derivative of \( D^{10} A_{EML}(\theta|\theta') \) with respect to the first argument.

In this case, we can verify that a \( \theta^* \) such that
\[
D^{10} A_{EML}(\theta^*|\theta^*) = 0 \text{ implies } D^{10} A_{EM}(\theta^*|\theta^*) = Dl(\theta^*) = 0,
\]
i.e., that \( \theta^* \) is a stationary point of \( l(\theta) \).

Assuming negative definite \( D^2 l(\theta^*) \) for such a \( \theta^* \), the asymptotic rate of convergence of the algorithm is
\[
||D M_{EM}(\theta^*)|| = ||I - [D^{20} A_{EM}(\theta^*|\theta^*)]^{-1} D^{2} l(\theta^*)||
\leq ||[D^{20} A_{EM}(\theta^*|\theta^*)]^{-1}|| ||D^{20} R_{EM}(\theta^*|\theta^*)||
\leq ||[D^{20} R_{EM}(\theta^*|\theta^*) + D^2 l(\theta^*)]^{-1}|| ||D^{20} R_{EM}(\theta^*|\theta^*)||
< 1 \tag{2.25}
\]

The form of given in (2.24) is common when the complete data density belongs to the exponential family. When the closed form does not exist, instead of maximizing \( A_{EM}(\theta|\theta') \) in each iteration, merely increasing from \( A_{EM}(\theta'|\theta') \), that is, finding \( \theta^{(t)} \) such
that $A_{EM}(\theta^{(t)}|\theta^{(t-1)}) \geq A_{EM}(\theta^{(t-1)}|\theta^{(t-1)})$ might be more efficient. This defines a general class of algorithms known as generalized EM (GEM) algorithms. One example of a GEM algorithm is the EM gradient algorithm which uses another quadratic approximation to increase $A_{EM}(\theta|\theta')$ in each iteration. GEM algorithms, however, can produce a sequence whose limit is not a stationary point of $l(\theta)$ unless the complete data density is from the exponential family (Wu, 1983).

EM (and, in general, GEM) algorithm can be used for estimation of parameters in various statistical models by taking the latent variable as unobserved (missing) data. And since complete-data loglikelihood is relatively simple, EM algorithm usually has a partial model reduction, that is, full computation of inverse matrix $[E(T_1(x)|y, \theta)]^{-1}$ is not necessary because $T_1(x)$ is usually block diagonal. Although these schemes satisfy the condition [a]-[d] for ML algorithms, they might be very slow and practically useless when the the proportion of missing data is relatively large compared to observed data since the sequence from EM is converging linearly with rate of convergence matrix given in (2.25).

2.2.3 Other approximations

- **Quadratic Approximation with Block Diagonal Hessian:** Assuming that parameter vector $\theta$ consists of $p$ block vectors $\theta_1, \cdots, \theta_p$, this approximation function is given by

$$
A_{q+b}(\theta|\theta') = l(\theta') + (\theta - \theta')^T D l(\theta') + \frac{1}{2} (\theta - \theta')^T \text{blockdiag}(h_1(\theta'_1), \cdots, h_p(\theta'_p)) (\theta - \theta'),
$$

(2.26)

where each of $h_i(\cdot), i = 1, \cdots, p$ is appropriate Hessian corresponding to $\theta_i$, and $\text{blockdiag}(\cdots)$ denotes block diagonal matrix with element $\cdots$. If $h_i(\theta_i) = \frac{\partial l(\theta)}{\partial \theta_i}$ is used, then this approximation leads to the Jacobi-Newton algorithm, a nonlinear Jacobi method which is introduced in Section 2.3.
• Quadratic Approximation with Hessian, $\frac{1}{\alpha} I$ : The approximation function

$$A_{qs}(\theta|\theta') = l(\theta') + (\theta - \theta')^T DL(\theta') + \frac{1}{2\alpha}(\theta - \theta')^T(\theta - \theta') \quad (2.27)$$

leads to the steepest ascent algorithm (Kennedy and Gentle, 1980), overrelaxation, and univariate Aitken acceleration algorithm according to methods used for estimating $\alpha$. Overrelaxation, and the univariate Aitken acceleration algorithm will be defined in Section 2.3.

### 2.3 The Extensions of Numerical Linear System Solvers to ML Algorithms

In this section, we review iterative linear system solvers and extend them in order to develop new, efficient ML algorithms.

#### 2.3.1 Numerical Linear System Solvers

When we have a linear system $A\theta = b$ where $A$ is a $p \times p$ nonsingular matrix with elements $a_{ij}$ and $b$ is $p \times 1$ vector with elements $b_i$, we can solve the system by several iterative schemes. One such is the Jacobi method whose iteration function is

$$M_{JA}(\theta') = A_D^{-1}[b - (A_L + A_U)\theta'] = (m_1^J, \ldots, m_p^J)^T,$$

$$m_j^J = a_{jj}^{-1}(b_j - \sum_{k \neq j} a_{jk}\theta'_k), j = 1, \ldots, p.$$  

where $A_D$, $A_L$ and $A_U$ are $p \times p$ matrices which consist of diagonal, strictly lower triangular , and strictly upper triangular elements of $A$, respectively. Since

$$DM_{JA}(\theta) = -A_D^{-1}(A_L + A_U)$$

is constant over $\theta$, Jacobi converges linearly if $A$ and $2A_D - A$ are positive definite matrix.
Gauss-Seidel method improves the speed of Jacobi by using iterates which are updated in previous steps in each new iteration. Its iteration function is

\[ M_{GS}(\theta') = (A_D + A_L)^{-1}(b - A_U\theta') = (m_1^G, \ldots, m_p^G)^T, \]

\[ m_j^G = a_{jj}^{-1}(b_j - \sum_{k>j}^p a_{jk}\theta_k' - \sum_{k<j}^p a_{jk}m_k^G), j = 1, \ldots, p. \]

With constant rate of convergence \( DM_{GS}(\theta) = -(A_D + A_L)^{-1}A_U \), convergence of this method is guaranteed if \( A \) is a positive definite.

One popular acceleration method of linear-converging sequences is by using an overrelaxation parameter. Assuming \( M(\cdot) \) is a linear-converging iteration function, the improved iteration function with overrelaxation is

\[ M_\omega(\theta') = (1 - \omega)\theta' + \omega M(\theta'), \tag{2.28} \]

where \( 1 < \omega < 2 \). Successive overrelaxation (SOR) method is a overrelaxation scheme applied to the Gauss-Seidel method. That is, the iteration function of SOR is

\[ M_{SOR(\omega)}(\theta') = (1 - \omega)\theta' + \omega M_{GS}(\theta'). \tag{2.29} \]

We can easily verify that if \( \lambda_{GS} \equiv ||DM_{GS}(\theta)|| < 1 \) and \( 1 < \omega \leq 1/(1 - \lambda_{GS}) \), then

\[ ||DM_{SOR(\omega)}(\theta)|| = ||(1 - \omega)I + \omega DM_{GS}(\theta)|| \]
\[ = (1 - \omega) + \omega||DM_{GS}(\theta)|| \]
\[ = |1 - \omega + \omega\lambda_{GS}| < \lambda_{GS}. \tag{2.30} \]

If \( A \) is positive definite and if \( \omega \in (1, 2) \), then the SOR is convergent regardless of \( \lambda_{GS} \). For example, overrelaxation with \( \omega = 1/(1 - \lambda_{GS}) \), called the univariate Aitken acceleration is a special case. Although this method has zero asymptotic rate of convergence by equation (2.30), when estimated \( \lambda_{GS} \) is bigger than 0.5, difference between true \( \lambda_{GS} \) and its numerical estimate may cause \( \omega > 2/(1 - \lambda_{GS}) \), \( ||DM_{SOR(\omega)}(\theta)|| > 1 \),
and hence divergence in the algorithm. But when the estimated $\lambda_{GS}$ is smaller than 0.5, Aitken acceleration of Gauss-Seidel is always convergent since $\omega = 1/(1 - \lambda_{GS}) \in (1, 2)$.

Since the convergence matrix of Gauss-Seidel, $DM_{GS}(\theta) = (A_D + A_L)^{-1}A_U$ is non-symmetric, i.e. a non-normal matrix, the largest eigenvalue of $DM_{GS}(\theta)$ may be a complex number, further accelerations of Gauss-Seidel and SOR which are based on an estimate of the largest eigenvalue of convergence matrix may not be possible. A simple approach for making the convergence matrix of Gauss-Seidel and SOR symmetric, is by the use of symmetric iteration, that is, by running one more iteration of Gauss-Seidel(or SOR) in the reverse order after each iteration is run in a given order. For more details of iterative linear system solvers, refer to Young(1982).

2.3.2 Use of Linear System Solvers in ML Algorithms

There are two ways of applying Gauss-Seidel (GS) to maximization problems. One is to use one or several iteration of GS in solving $D^{10}A(\theta|\theta') = 0$ in each iteration. Since $D^{10}A(\theta|\theta') = 0$ can be easily a linear problem, i.e., $D^{10}A(\theta|\theta') = C\theta + c = 0$ for some square matrix $C$ and vector $c$, this extension of GS might be effective, when $\theta$ has large dimension. We shall call this scheme approximated Gauss-Seidel (AGS).

The other approach is to divide $\theta$ into subsets $\theta_1, \cdots, \theta_p$ and likelihood equation $Dl(\theta) = 0$ into corresponding partitioned equations $Dl_i(\theta_i) = 0, \cdots, Dl_p(\theta_p) = 0$, respectively and solve every approximated likelihood equation

$$D^{10}A_i(\theta_i|\theta') = 0, i = 1, \cdots, p$$

once or several times in each iteration. This method is known as nonlinear Gauss-Seidel (NGS).

If the iterate in previous iteration is used in the next iteration instead of continuing to use the last iterate from the previous step, the scheme is called nonlinear Jacobi (NJA). When the off-diagonals of $D^2l(\theta)$ are relatively small, nonlinear Jacobi saves on compu-
tations necessary to be done in each step of NGS, while it has speed similar to that of NGS.

2.3.2.1 Newton-type Algorithms

Since methods of quadratic approximation described in Subsection 2.2.1 are designed to solve a linear system of the form

$$H(\theta')\theta = H(\theta')\theta' - Dl(\theta')$$

(2.31)

in each iteration, we can easily apply several iterations of Jacobi or Gauss-Seidel to each iteration of these methods. Running one iteration of Jacobi or Gauss-Seidel in the quadratic approximation method, for example, provides a smaller step $\theta^{(t)} - \theta^{(t-1)}$ than when direct quadratic approximation method is used, and then leads to a well-fitted quadratic approximation in the neighborhood of $\theta^{(t-1)}$. However, convergence of quadratic approximated Jacobi or Gauss-Seidel depends on the convergence of the original quadratic approximation algorithm. (Ortega and Rheinboldt, 1970)

Partition $\theta$ and $Dl(\theta)$ into $\theta_1, \ldots, \theta_p$ and $Dl_1(\theta_1|\theta_{-1}), \ldots, Dl_p(\theta_p|\theta_{-p})$, respectively, where $\theta_{-j} = (\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_p)^T$. Let $D^2l_{jj}(\theta_j|\theta_{-j}), j = 1, \ldots, p$ denote the $j$th diagonal element of $D^2l(\theta)$. Then the $t$th iteration of nonlinear Gauss-Seidel using quadratic approximation (NGSQ) has $p$ steps, the $j$th step of which computes

$$M_j(\theta(t)) = \theta_j^{(t+1)} = \theta_j^{(t)} [-D^2l_{jj}(\theta_j^{(t)}|\theta_{-j}^{(t)})]^{-1} Dl_j(\theta_j^{(t)}|\theta_{-j}^{(t)})$$

for $j = 1, \ldots, p$, (2.32)

where $\theta_{-j}^{(t)} = (\theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)})^T$. Hence, its iteration function is

$$M_{NGSQ}(\theta) = (M_1(\theta), \ldots, M_p(\theta))^T.$$

The usefulness of this algorithm can be seen by the following two theorems:

Theorem 2 Suppose that for a local maximum $\hat{\theta}$, each of $D^2l_{jj}(\theta_j|\theta_{-j}) < 0$, $j = 1, \ldots, p$ is strictly increasing on $(-\infty, \hat{\theta}_j]$ and strictly decreasing on $[\hat{\theta}_j, \infty)$ with respect to the first argument. Then $\{l((\theta^{(t)}))$ from the NGSQ converges monotonely to $l(\hat{\theta})$. 
Theorem 3 Under the same assumption as in Theorem 2 above, Partition $D^2 l(\hat{\theta})$ into $D^2 l(\hat{\theta}) = L + D + U$, where $D$, $L$, and $U$ are diagonal, strictly lower triangular, and strictly upper triangular matrices of $D^2 l(\hat{\theta})$. Then the NGSQ has its rate of convergence

$$DM_{NGSQ} = -D^{-1}(L + U) - (D + L)^{-1}U[D^{-1}D^2 l(\hat{\theta})].$$

(2.33)

If off-diagonals of $D^2 l(\theta)$ are relatively small, we can use NJA using quadratic approximation (NJAQ). In fact, this NJAQ is the same as usual quadratic approximation where $D^2 l(\theta)$ is assumed to be a diagonal matrix.

2.3.2.2 EM-type Algorithms

Approximated Gauss-Seidel version of EM algorithm is the Expectation-Conditional Maximization (ECM) algorithm of Meng and Rubin (1993). Assume $\theta$ is partitioned into several vectors, i.e., $\theta = (\theta_1, \cdots, \theta_p)^T$. If there exist $K(\cdot), T_1(\cdot, \cdot)$, and $t_2(\cdot, \cdot)$ for a given complete-data loglikelihood $f_c(x|\theta)$ such that

$$K(\theta)\frac{\partial}{\partial \theta} \log f_c(x|\theta) = T_1(x, \theta)\theta - t_2(x, \theta),$$

(2.34)

where the $i$th block rows of the $p \times p$ block matrix, $T_1(x, \theta)$ and the $i$th block of the $p \times 1$ block vector $t_2(x, \theta)$ are not dependent on $\theta_i$, $i = 1, \cdots, p$. Of course, $\theta$ need to be reparameterized to obtain equation (2.34) or more complex partitioning using space-filling condition (Meng and Rubin, 1993) might be needed. Definitions related to the space-filling is presented in Appendix B.

Each iteration of a ECM algorithm performs an iteration of Gauss-Seidel to solve the linear system,

$$E(T_1(x, \theta)|y, \theta')\theta = E(t_2(x, \theta)|y, \theta'),$$

(2.35)

where $\theta'$ is the current iterate of the parameter vector, and $y$ is the observed data. In general, ECM has slower convergence rate than EM algorithm, but retains the same
desirable convergence properties as EM (See Meng(1994), Meng and Rubin(1993) for details).

Meng(1994) also shows that the rate of convergence of ECM given a fixed point \( \theta^* \) is

\[
DM_{ECM}(\theta^*) = DM_{EM}(\theta^*) + DM_{CM}(\theta^*)(I - DM_{EM}(\theta^*)),
\]

(2.36)

where \( DM_{EM}(\theta^*) = [D^{20}A_{EM}(\theta^*|\theta^*)]^{-1}D^{20}R_{EM}(\theta^*|\theta^*) \), with \( A_{EM} \) and \( R_{EM} \) defined in Section 2.2.2, is the rate of convergence of EM algorithm assuming we maximize \( A_{EM}(\cdot|\theta(t)) \) in each iteration for a given model, and \( DM_{CM}(\theta^*) = -(D_{EM} + L_{EM})^{-1}U_{EM} \) is the rate of convergence of iterative conditional modes algorithm assuming we have complete data and \( D_{EM}, L_{EM}, \) and \( U_{EM} \) are diagonal, strictly lower triangular, and strictly upper triangular matrices containing the corresponding elements of \( D^{20}A_{EM}(\theta^*|\theta^*) \), respectively.

A modified version of ECM algorithm can be obtained by performing more than one iteration of Gauss-Seidel to solve (2.35) after each E-step. We shall denote this as the Expectation-Repeated Conditional Maximizations(ERCM) algorithm. ERCM inherits all convergence properties of ECM described in Meng and Rubin(1993) and in addition, improves convergence rate of ECM algorithm, due to this simple modification.

Theorem 4 Under the assumptions of Theorem 1 in Meng(1994), the rate of convergence of ERCM is given by

\[
DM_{ERCM}(\theta^*) = DM_{EM}(\theta^*) + (DM_{CM})^r[I - DM_{EM}(\theta^*)],
\]

(2.37)

where \( r \) is the number of repetitions.

The proof of the Theorem 4 follows in the same lines as the convergence result in the case of ECM (Meng 1994). By equation (2.37), the speed of ERCM (\( r = 2 \) without loss of generality) is

\[
I - DM_{ERCM} = (I - (DM_{CM})^2)(I - DM_{EM})
\]

\[
= (I + DM_{CM})(I - DM_{CM})(I - DM_{EM})
\]
\[ (I - DM_{CM})(I - DM_{ECM}), \]

where \( DM_{ECM} \) is the rate of convergence of ECM. ERCM is typically faster than ECM because of the fact that positive definiteness of \(-A_{EM}(\theta^*|\theta^*)\) implies \( 0 \leq ||DM_{CM}|| \leq 1 \). ERCM is useful especially when E-step has a relatively large computational burden compared to the CM-steps in ECM. For practical purposes, repeating CM-steps once, that is setting \( r = 2 \), is sufficient in many cases.

### 2.3.2.3 Overrelaxation Methods in ML Algorithms

Overrelaxation methods for nonlinear systems are a little different from those for linear systems. Since any sequence from nonlinear system solvers do not converge linearly at the beginning, we have to make use of the overrelaxation parameter \( \omega \) only when the sequence is beginning to converge linearly. Overrelaxation method for monotone approximation algorithms is justified by the following theorem.

**Theorem 5** Assume that \( A(\theta|\theta') \) is a concave function over the first argument given any \( \theta' \in \Omega \) and that overrelaxation for the algorithm is started for a sufficient large \( t' \) so that

\[
R(t) = M[M(\theta^{(t)})] - M(\theta^{(t)}) - \tau(M(\theta^{(t)}) - \theta^{(t)})
\]

is negligible for each \( t \geq t' \), where \( M(\cdot) \) is the iteration function of the algorithm. Suppose that \( A(M(\theta)|\theta) \geq A(\theta|\theta) \) and that this implies \( l(M(\theta)) \geq l(\theta) \) for all \( \theta \in \Omega \), that is, the algorithm has the monotone convergence property. If \( (\omega - 1)/\tau < 1 \) for a given \( \omega \) such that \( 1 < \omega < 2 \), then the overrelaxation method given by

\[
M_\omega(\theta^{(t)}) = (1 - \omega)\theta^{(t)} + \omega M(\theta^{(t)}), \text{ for } t \geq t'
\]

also has monotone convergence, and its rate of convergence is \( (1 - \omega)I + \omega DM(\theta^*) \).

**Proof** By assumption above,

\[
M_\omega(\theta^{(t)}) = (\omega - 1)(M(\theta^{(t)}) - (\theta^{(t)}) + M(\theta^{(t)})
\]
\[ \frac{(\omega - 1)}{\tau} (M(M(\theta^{(t)})) - M(\theta^{(t)})) = \frac{(\omega - 1)}{\tau} + M(\theta^{(t)}) \]

Then since \((\omega - 1)/\tau < 1\), concavity of \(A(\cdot|\theta^{(t+1)})\) completes the first part of proofs. The second part is trivial.

Thus, an overrelaxation method based on a monotonely convergent approximation method inherits the monotone convergence property. \(A_q(\theta|\theta')\) is always concave, and the concavity of \(A_{EM}(\theta|\theta')\) can be easily verified when complete-data density is from regular exponential family (Dempster et al, 1977). Also, \(\omega\) can be obtained by setting \(\omega \approx 1 + \tau\), where \(\tau\) is the numerical convergence rate of \(l(\theta^{(t)})\). Because computing numerical convergence rate of \(l(\theta^{(t)})\) is computationally cost free, overrelaxation for GEM algorithms increases the efficiency for many models. In addition, one can easily derive supplementary algorithms for computing an approximated variance-covariance matrix for a fixed point \(\hat{\theta}\). Note that, in practice, a suitable cutoff value for a negligible \(R(t)\), which is the criterion used for checking linear convergence, is about 0.005 ~ 0.0005.

2.4 Generalized Conditional Maximization Algorithms

As we review approximations for maximization of loglikelihood, we find that there is no perfect scheme which works for all probabilistic models. Furthermore, even in considering a single model, one approximation might be suitable for a part of parameter space while a different scheme may work for the rest. Accordingly, we introduce an approach towards efficient ML computations through an algorithm we term generalized conditional maximization (GCM).

For the purpose of defining the GCM algorithm, divide \(\theta\) into \(B\) block vectors \(\theta_1, \theta_2, \ldots, \theta_B\), and define \(G_c(\theta), c = 1, \ldots, C\) as, possibly overlapping, subsets of these block vectors, i.e., \(G_c(\theta) = \{\theta_{c_1}, \theta_{c_2}, \ldots, \theta_{c_B}; c_i \in \{1, 2, \ldots, B\}\}\), where \(B_c\) is the number
of block vectors in each subset, \( G_c(\theta), c = 1, \cdots, C \), and \( \theta = \cap_{c=1}^C G_c(\theta) \).

**Definition:** GCM algorithm \( \theta^{(c)} \) denotes the iterate after \( c-1 \)th cycle of each iteration. A \( t \)th iteration of GCM consists of \( C \) cycles, each \( c \)th cycle of which has three steps:

*Approximation step (A-step):* compute \( A^c(\theta|\theta^{(c-1)}) \) which is an approximation function of \( l(\theta) \) in the neighborhood of \( \theta^{(c-1)} \) with complement set of \( G_c(\theta) = \theta \cap [G_c(\theta)]^c \) fixed at the previous iterate.

*Conditional Maximization steps (CM-steps):* find \( \theta^{(c-1+b/Bc)} \) such that

\[
A^c(\theta^{(c-1+b/Bc)}|\theta^{(c-1)}) \geq A^c(\theta|\theta^{(c-1)}),
\]

for all \( \theta \in \{ \theta \in \Omega : \theta \star \theta_{cb} = \theta^{(c-1+(b-1)/Bc)} \star \theta_{cb}^{(c-1+(b-1)/Bc)} \} \),

where \( \star \) is the set-difference operator.

*Repetition step (R-step):* Repeat CM-step or both of A-step and CM-step \( r_c \) times.

The \( t \)th iteration is completed by setting \( \theta^{(t+1)} = \theta^{(C+1+Bc/Bc)} \). Optionally, an overrelaxation of this iterate can be performed:

*Overrelaxation (optional):* Perform overrelaxation procedure at the end of the iteration as described in Section 2.3, i.e., new \( \theta^{(t+1)} = (1 - \omega)\theta^{(t)} + \omega \theta^{(t-1)} \). Dividing \( \theta \) into \( \theta_1, \theta_2, \cdots, \theta_B \) as defined above is a simple partition of the parameter space. Partition by space-filling conditions (Meng and Rubin, 1993) would be more general and can be applied to GCM without difficulty. (see Appendix B for more details)

In most cases, however, a simple partition with necessary reparameterization is sufficient. The R-step may only be necessary to balance differences in speed among cycles. Whether we repeat CM-steps or both of A-step and CM-steps in an R-step depends on the computational complexity of \( A^c(\theta|\theta^{(c-1)}) \). If computational burdens of A-step and CM-step are similar, repeating both of A-step and CM-step leads to faster convergence. However, when computing cost of A-step is relatively large, repeating only the CM-steps is a good idea for increasing efficiency. Note that methods of quadratic approximation
and EM algorithm become special cases of the GCM algorithm with one cycle and only one CM step where $G_1(\theta)$ consists of only one block vector.

Since the ECM algorithm also can be viewed as a GCM with one cycle consisting of $p$ CM steps and no R-step, we may regard ERCM algorithm as a GCM with one cycle of $p$ CM steps and an R-step which repeats the CM steps only. Also overrelaxation of a GEM algorithm can be considered as a special case of GCM using overrelaxation as the iterative scheme over cycles.

A method based on a single approximation with single full maximization might be enough for most of simple statistical models. But many complex and sophisticated models have been developed in the areas of robustness, missing data, correlated data, and more flexible and efficient techniques are needed for these models. We believe that GCM algorithm plays a role as a flexible approach for these problems.

Before going into further discussion in Chapter 3, the difference between GCM and ECME should be mentioned. Partition $\theta$ to $(\theta_1, \theta_2)$. For $\theta_1$, each iteration of ECME maximizes the conditional expectation of complete-data loglikelihood given observed data, while it directly maximizes observed loglikelihood $l(\theta)$ for $\theta_2$. The maximization for $\theta_2$, i.e., direct maximization of $l(\theta)$ in terms of $\theta_2$, is generally performed by several iterations of a quadratic approximation method unless a closed form solution exists for this maximization. However, although maximizing $l(\theta)$ in terms of $\theta_2$ by a quadratic approximation method in each iteration is ideal, it is not practical since too many unnecessary inner iterations are needed for the inner maximization of $l(\theta)$. Thus, either one or a few inner iterations are generally done instead of full maximization. This practical ECME as well as the ideal one can be viewed as a GCM algorithm consisting of two approximations, EM and quadratic with several R-steps for $\theta_2$. But more efficient GCM algorithms than the ECME can be developed so that it avoids the problem of inner maximizations in each iteration. We will discuss these in later chapters.
3 LINEARIZATION OF LIKELIHOOD EQUATIONS

3.1 Introduction

In Section 2.2, an overview of two approximation methods which are popular in statistics was presented. Linearity of the specified likelihood equation is advantageous in the complete-data approximation as well as in the quadratic approximation since linearity implies the existence of a closed form solution to the complete-data likelihood equations.

As discussed in Section 2.2.1, conventional linearization involves multiplying $Dl(\theta)$ by a positive definite matrix $K(\theta)$ so that $Dl_L(\theta) \equiv K(\theta)Dl(\theta)$ is nearly linear. In this case, instead of the iteration function of quadratic approximation function, the linearized iteration function

$$M_L(\theta') = \theta' - [D^2l_L(\theta)]^{-1}Dl_L(\theta)$$

is used giving a modified Newton's algorithm, where $D^2l_L(\theta)$ is the first derivative of $Dl_L(\theta)$.

There are few examples of this linearization in statistical literature, Callanan and Harville(1991) found some linearization factors $K(\theta)$ with different parameterizations to improve speed and convergence of Newton's algorithms for REML estimation of variance components. They derived their linearization factors from analogy of the balanced data case. But it seems that there are no guidelines for obtaining such linearization factors in general.
In this chapter, we establish a procedure for finding the linearization factors for fast and monotone convergence of Newton's algorithm in univariate parameter case. Once we construct the theorem and provide guidelines for the univariate case, we can apply them for multivariate problems using Jacobi or Gauss-Seidel methods. Thus univariate linearization helps the development of an efficient GCM algorithms that will be discussed in later chapters.

Section 3.2 provides a new definition of the linearization factor $K(\theta)$ and an existence theorem of such a $K(\theta)$ under mild conditions in the one dimensional parameter case. A procedure for determining the linearization factor is described in Section 3.3 and three examples of applications of univariate linearization are presented in Section 3.4.

### 3.2 Theory of Univariate Linearization

Transforming the nonlinear likelihood equations to be exactly a linear function is, although ideal, impossible to achieve in general. But in the univariate case, by Corollary 1 in Subsection 2.2.1, we learned that if the score function $Dl(\theta)$ is concave (convex) on a subspace in which $Dl(\theta)$ is negative (positive), the sequence of the iterates from Newton's algorithm is guaranteed to converge monotonely to a local maximum when the initial value belongs to that subspace.

Similarly, we can derive a multivariate version of the monotone global convergence theorem of Newton's algorithm. This theorem can be also used with EM gradient algorithm for preserving monotone convergence. Before proceeding further, we define the concept of linearization of likelihood equations as follows:

Linearization is defined by multiplying the score function $Dl(\theta)$ by a positive definite matrix $K'(\theta)$ in order to obtain a nearly linear and concave (convex) function $Dl_L(\theta) = K'(\theta)Dl(\theta)$ on the subspace in which $Dl_L(\theta)$ is negative (positive). We shall call $K(\theta)$ the linearization factor. Optimal linearization is a linearization for which $K'(\theta)$ leads to
the smallest $||D^3L(\theta)||$. A linearization guarantees fast speed of convergence as well as monotone convergence if an almost-optimal $K(\theta)$ can be found.

Corollary 1 in Chapter 1 actually suggests a strong reason for seeking a linearization in the univariate case. That is, if we can find a linearization factor satisfying the above definition under the assumption of the Corollary, the sequence \{\theta^{(i)}\} from Newton's algorithm constructed from $DL(\cdot)$, $D^2L(\cdot)$, and a suitably chosen initial value $\theta^{(0)}$ always converges monotonely to a unique local maximum. The following Lemma states this Corollary in a useful form:

**Lemma 1** Assume that $L(\theta) : R \rightarrow R$ has $\theta$ as the only stationary point which is the local maximum on $[a, b]$. If there is a $K(\theta) : [a, b] \rightarrow R^+ \equiv (0, \infty)$ such that $DL(\theta) = K(\theta)DL(\theta)$ is concave (convex) on $[\theta, b] ([a, \theta])$, then Newton's algorithm with initial value $\theta^{(0)} \in [\theta, b] \in [a, \theta]$ defined as $\theta^{(i+1)} = \theta^{(i)} - DL(\theta^{(i)})/D^2L(\theta^{(i)})$ has monotone convergence and its sequence of iterates converges to $\theta$.

Proof of this lemma is obvious by Corollary 1 in Subsection 2.2.1 since $DL(\theta)$ has the same stationary point $\theta$ and the same sign as $DL(\theta)$ on $[a, b]$. The existence of such $K(\theta)$ is proved by the univariate linearization theorem as follows:

**Theorem 1** Assume that there exists a positive, concave, and increasing function $t(\theta) : \Omega \rightarrow R^+$. And suppose that $l(\theta) : R^1 \rightarrow R^1$ has $\theta$, the only stationary point which is a local maximum on $[a, b]$, and that $l(\theta)$ has continuous first derivative $DL(\theta)$ and continuous second derivative $D^2L(\theta)$. Then for any $t(\theta)$ and any $l(\theta)$ satisfying these assumptions, there is a positive (negative) value $p^*$ such that $DL(\theta) = (t(\theta))^p DL(\theta)$ is concave (convex) on $[\theta, b] ([a, \theta])$ for all $p > p^*$ ($p < p^*$).

Proof Note that $DL(\theta) < 0$ on $[\theta, b]$ and that $DL(\theta)$ and $D^2L(\theta)$ are bounded on $[\theta, b]$. Set

$$p_1 = \max_{\theta \in [\theta, b]} \frac{|D^2L(\theta)Dt(\theta)|}{-DL(\theta)}, \quad (3.1)$$
\[ S_1(\theta) = (t(\theta))^{p_1} Dl(\theta), \text{ and} \]
\[ p_2 = \max_{\theta \in [\theta, b]} \frac{|D^2S_1(\theta)Dt(\theta)|}{-DS_1(\theta)} . \] (3.2)

Then it is sufficient to prove that

\[ S_2(\theta) = (t(\theta))^{p_2} S_1(\theta) = (t(\theta))^{p_1+p_2} Dl(\theta) \]

is concave on \([\theta, b]\). By (3.1), \(S_1(\theta)\) has first derivative

\[ DS_1(\theta) = (t(\theta))^{p_1} D^2l(\theta) + p_1(t(\theta))^{p_1-1} Dt(\theta) Dl(\theta) \]

which is negative on \((\theta, b]\). Since

\[ D^2S_2(\theta) = p_2(p_2 - 1)(t(\theta))^{p_2-2}(Dt(\theta))^2 S_1(\theta) + p_2(t(\theta))^{p_2-1} D^2t(\theta) S_1(\theta) \]
\[ +2p_2(t(\theta))^{p_2-1} Dt(\theta) DS_1(\theta) + (t(\theta))^{p_2} D^2S_1(\theta), \] (3.3)

We can easily verify that \(D^2S_2(\theta)\) is negative on \((\theta, b]\) by the concavity of \(t(\theta)\) and the definition of \(p_2\). The proof is complete.

As might be noticed in the proof above, concavity of \(t(\theta)\) is a stronger assumption than needed since theorem only requires that \(t(\theta)\) satisfies \((Dt(\theta))^2 - t(\theta) D^2t(\theta) > 0\). By the two theorems above, we can develop a Newton-type algorithm with monotone convergence assuming that an appropriate "linearization index" \(p\) is selected.

As an application of Theorem 1, an algorithm using the nonlinear Jacobi and nonlinear Gauss-Seidel with the quadratic approximation can be also adapted to solve a multivariate nonlinear likelihood problem by considering it as several univariate nonlinear problems.

Although linearization can dramatically improve conventional maximization algorithms, there is a serious lack of methods for the selection of appropriate linearization. In statistics, Callanan and Harville(1991) suggest linearization of the score function for obtaining maximum likelihood estimates of variance components, but they proposed to
derive the linearization factor by intuition from studying the balanced data case. In the following discussion, we shall consider the univariate likelihood function \( l(\theta) \); the results presented can be extended to the general multi-parameter likelihood case using methods indicated above. For the univariate likelihood function \( l(\theta) \), we use plots of \( Dl(\theta) \) obtained from simulated data for selecting the linearization index for a given \( Dl(\theta) \) and \( t(\theta) \). Once the linearization index and \( t(\theta) \) is obtained, we can develop fast and efficient maximum likelihood (ML) algorithms by applying Lemma 1. We introduce the procedure in detail in Section 3.3.

### 3.3 Selection Procedure for Univariate Linearization

In Section 3.2, we have assumed the existence of a function \( t(\theta) \) that satisfies the condition of Theorem 1. In fact, there are not many suitable selections of \( t(\theta) \) on a given parameter space. In other words, the selection totally depends on the parameter space.

For example, when \( \theta \) belongs to \( \Omega \equiv (0, \infty) \) and the loglikelihood is strictly concave or concave on the majority of the parameter space, \( t(\theta) = \theta \) provides a good linearization function in practice. If \( \theta \) belongs to \( \Omega \equiv (0, 1) \), although \( t(\theta) = \theta \) can still be a selection, \( t(\theta) = -\log(1 - \theta) \) is a better selection. Similarly, \( t(\theta) = \exp(\theta) \) might be more suitable on \( \Omega \equiv (-\infty, \infty) \).

Since the parameter space can be established using a proper reparameterization of \( \Omega \), finding \( t(\theta) \) can be considered as obtaining a reparameterization of the loglikelihood surface to one that is concave over most of the new parameter space. To determine the best \( t(\theta) \), several reparameterizations must be attempted.

Assuming that we have selected the best \( t(\theta) \), we use Theorem 1 to determine \( p \). Although Theorem 1 is indispensable, \( p_1 + p_2 \), provided by the theorem, is not a sharp upper-bound for the linearization index \( p \). That is, the smallest \( p \) which guarantees concavity of \( DL_2(\theta) \) on \((\hat{\theta}, b)\) is a value in \([p_1, p_1 + p_2] \). In the absence of other information
that will provide a sharp upper-bound for \( p \). We suggest below a statistical procedure for finding an optimal linearization index:

0 select a large enough \( p \) which guarantees concavity of \( Dl(\theta) \) on the subspace.

1 generate \( m \) sets of data given the number of observations from the specified density.

2 compute large enough (say, 200) grid of points of \( D^2l(\theta) \) over the range of \( \theta \) for each data set.

3 continue until the value of \( D^2l(\theta) \) decreases at any grid point. Otherwise, decrement \( p \) and go to 1.

Suppose that \( \epsilon \) is the error probability that with a simulated data, we decide concavity of \( Dl(\theta) \) which is not truly concave on the specified area. Then the probability that we have selected the incorrect linearization index with the procedure above is \( \epsilon^m \). When we use 10-30 random data sets, thus, the concavity of \( Dl(\theta) \) is almost guaranteed with our procedure.

3.4 Univariate Linearizations Examples

3.4.1 Univariate t-model with Unknown Degrees of Freedom

Consider a sample of size \( n \) from the \( k \) dimensional \( t_v(\mu, \Sigma) \) model.

\[
y_i|\tau_i, \mu, \Sigma \sim N(\mu, \Sigma/\tau_i) \tag{3.4}
\]

\[
\tau_i|v \sim \chi^2_v/v, \quad i = 1, \ldots, n, \tag{3.5}
\]

where \( \mu \) and \( \Sigma \) are assumed to be a known \( k \times 1 \) vector and a \( k \times k \) symmetric positive definite matrix. For estimating \( v \) via maximum likelihood of one dimensional t-model, we have to maximize

\[
\log l(\theta|y) = -0.5n \log(|\Sigma|) + 0.5nv\log(v) + n \log \Gamma\left(\frac{v+1}{2}\right) - n \log \Gamma\left(\frac{v}{2}\right)
\]
\[-0.5(v + 1) \sum_{i=1}^{n} \log(v + \delta_i),\]

where \(\Gamma(\cdot)\) denotes the gamma function, \(\theta = (\mu, \Sigma, v)\), and \(\delta_i = (y_i - \mu)^T \Sigma^{-1} (y_i - \mu), i = 1, \cdots, n\). To investigate the loglikelihood surface in terms of \(v\), we simulate \(m = 30\) number of data sets from the one dimensional \(t_\nu(0, 1)\) with varying sample sizes. As mentioned Section 3.3, simulation number \(m = 30\) is large enough for reducing error probability that we choose wrong linearization index.

Figure 3.1 shows simulated traces for the original likelihood, its first, and second derivative functions for selected data sets. In the neighborhood of the MLE of the degrees of freedom \(\hat{v}\), first derivative function is constant or steeply quadratic and hence the second derivative is close to 0. This shape of score function leads to nonconvergent quadratic approximation algorithms and very slow ELBQA algorithms. This is perhaps the reason why Liu and Rubin(1995) suggest direct linear search for finding MLE of \(v\).

First, the linearization index of \(p = 2.6\) with \(t(\theta) = u\), i.e., linearization factor \(K(\theta) = v^{2.6}\), is tested as in Figure 3.2. The linearized score function appears to be almost linear, but it fails to preserve concavity on the desired subset of \(\Omega\). With linearization factor \(K(\theta) = v^3\), however, linearized score function \(Dl_L(\theta) = K(\theta) Dl(\theta)\) displays more linear behaviour and is concave on \(\{\theta; Dl(\theta) < 0\}\) with the same data sets as those used for generating Figure 3.3. We also find that the linearized Hessian function is bounded below and decreasing over the space such that \(Dl_L(\theta)\) is negative, provided that the loglikelihood function is bounded above. Hence by setting \(B(\theta) = D^2l_L(\theta)\) and using large initial values for \(v\), a fast ELBQA algorithm can be developed for this optimization problem.
Figure 3.1 Original Loglikelihood, Its First Derivative, and Its Second Derivative Functions as a Function of Degrees of Freedom $v$
Figure 3.2  Original Loglikelihood, Its Linearized Score, and Its Linearized Hessian Functions as a Function of Degrees of Freedom $v$ with $p=2.6$
Figure 3.3  Original Loglikelihood, Its Linearized Score, and Its Linearized Hessian Functions as a Function of Degrees of Freedom $v$ with $p=3.0$
3.4.2 Two Component Finite Normal Mixture Model

The normal (Gaussian) finite mixture model is indispensable in applied statistics. The loglikelihood of normal (Gaussian) finite mixture model with two components is

$$l(\theta) = \sum_{i=1}^{n} \log \left( p \phi(y_i|\mu_1,\Sigma_1) + (1-p)\phi(y_i|\mu_2,\Sigma_2) \right),$$

where $\phi(\cdot|\mu,\Sigma)$ is the density function of $N(\mu,\Sigma)$, and $\theta = (p,\mu_1,\mu_2,\Sigma_1,\Sigma_2)$. EM algorithm for this model (Dempster et al., 1977) is known to be an efficient method for finding MLEs of $\theta$ since users of any quadratic approximation method are confronted with the dimensionality problem and with failure of monotone convergence (Everitt and Hand, 1980). However, the slow convergence has been pointed out as one of the disadvantages of the EM algorithm for this problem. For convenience of discussion in the univariate parameter case, $\mu_j$ and $\Sigma_j$, $j = 1,2$ are assumed to be known, and the interest is in obtaining MLE of $p$.

As Figure 3.4 shows the loglikelihood surface as a function of $p$, second derivative of loglikelihood is close to zero in the neighborhood of the maximum, and hence a slow EL-BQA algorithm is expected. Furthermore we have to perform constrained optimization due to the restricted space of $p$.

In order to overcome the slow speed of the EM algorithm, a reparameterized quadratic approximation algorithm for obtaining MLE of $p$ was used by Shin and Marasinghe (1997). They reparameterized $p$ to $\tau = \log \frac{p}{1-p}$ since the first derivative function of loglikelihood in terms of $\tau$ is locally linear as shown Figure 3.5 shows, and any adjustment for constrained optimization is not needed.

But the first derivative function of loglikelihood is not monotonely decreasing for all $\tau$ and then it may lead to failure of the monotone convergence of the algorithm due to the second derivative becoming positive. Thus, instead of the second derivative, they used a lower bound of the Hessian function which is negative for all $\tau$ and is close to the second derivative in the neighborhood of a mode as indicated in the last column of
Figure 3.5. For the case that $\mu_j$ and $\Sigma_j$, $j = 1, 2$ are assumed to be unknown, Shin and Marasinghe successfully developed new ML algorithms for two component finite normal model by using this Hessian function, that performed better than EM algorithm.

By graphical inspection, however, we found better score and Hessian functions by reparameterizing $p$ to $\alpha = p/(1 - p)$ and using the linearization factor $\alpha^{2.8}$ as shown in Figure 3.6. Since the linearized Hessian function is nonincreasing like in the t-model, Newton-Raphson becomes an ELBQA algorithm by the Corollary 1 in Subsection 2.2.1.

Figure 3.4 The Surfaces of Loglikelihood, First, and Second Derivative as a Function of $p$ for Various True $p = 0.2, 0.4, 0.7$, and 0.9
Figure 3.5  The Surfaces of Loglikelihood, First, Second Derivative (Thick Lines) and Modified Hessian Function (Thin Lines) as a Function of $\tau$ for Various True $p = 0.2, 0.4, 0.7, \text{ and } 0.9$
Figure 3.6 The Surfaces of Loglikelihood, Linearized Score, and Linearized Hessian as a Function of $\alpha = p/(1 - p)$ for Various True $p = 0.2, 0.4, 0.7, \text{and} 0.9$
3.4.3 Neural Network Prediction Model

Neural networks have recently gained its importance in statistics as well as in the artificial intelligence area. Although the model was derived from structure of brain activity, it may be considered as a projection pursuit regression with logistic function as a smooth function. (Ripley, 1994) Without giving any more detailed explanation concerning neural network prediction (see Faraggi and Simon, 1995 for a detailed account), we consider it here as a statistical model in which dependent variable $y_i$ and predictor variable $x_i, i = 1, \ldots, n$ have the following relationship:

$$y_i = \alpha_0 + \sum_{h=1}^{H} \alpha_h /[1 + \exp(-w_h^T x_i)] + \epsilon_i, \quad (3.7)$$

where $\epsilon_i \sim N(0, \sigma^2), x_i = (x_{i0}, \ldots, x_{ip}), w_h = (w_{h0}, \ldots, w_{hp}), i = 1, \ldots, n,$ and $H$ is an integer determined in advance. The term related to each $h$ is called hidden node in the neural network area. (Thus, in the model above, we have $H$ hidden nodes) Here each element of $x_i, i = 1, \ldots, n$ is scaled so that its value lies between 0 and 1. In our experience the scaling of $x_i$'s appears to produce a smoother likelihood surface than nonscaled version.

As the model (3.7) indicates, $\alpha_h, h = 0, \ldots, H$ are computed in one iteration by a linear regression procedure when the values of 'weight vectors' $w_h$ are found. But obtaining $w_h$s maximizing loglikelihood requires an iterative optimization method.

Most common method which is known as backpropagation is a kind of steepest ascent method which needs only score function with identity matrix $I$ as Hessian matrix. Main reason using this method is that it is least affected by curse of dimensionality. For example, suppose that we use a quadratic approximation methods. Then each iteration of the method for a neural network model requires $H(p + 1) \times H(p + 1)$ Hessian matrix even though computation for $\alpha_h$s is not included. In this way, a user will surely face 'memory not sufficient' error when he has large number of $H$ or $p$. 
But backpropagation method becomes very slow at the end of convergence since it does not use second derivatives at all. In this section, we will study the possibility of improvement of the algorithm by univariate linearization.

\(\alpha_h, h = 0, \ldots, H\) and other weight \(w_{hj}\) except for \(w_{h1}\) are assumed to be known values in our discussion for setting a univariate problem. Using the same type of simulations as in the two models previously considered, Figure 3.7 is constructed and displays the traces of the loglikelihood, its first and second derivatives in terms of \(w_{h1}\). This parameterization produces a nonconcave likelihood surface and hence strong nonlinearity of \(Dl(\theta)\). In this strong nonlinearity, linearization index should be big for concavity of \(Dl_L(\theta)\).

Another parameterization \(\pi_{h1} = \exp(w_{h1})/(1 + \exp(w_{h1}))\), produces a smoother surface for \(Dl(\theta)\) as shown in Figure 3.8. Even with this linear-like surface, however, concavity of \(Dl(\theta)\) is not guaranteed as the second row of Figure 3.8 shows. With \(t(\theta) = -\log(1 - \pi_{h1})\) and \(p = 0.7\), the concavity of \(Dl_L(\theta)\) is shown to be occupied in Figure 3.9.
The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $w_{h1}$ for Various Seeds
Figure 3.8 The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $\pi_{h1} = \exp(w_{h1})$ for Various Seeds
Figure 3.9 The Surfaces of Loglikelihood, Its First Derivative, and Second Derivative in Terms of $\pi_{h1} = \exp(w_{h1})$ for Various Seeds
4 DESIGN OF A GCM ALGORITHM

4.1 Introduction

Although the development of a GCM algorithm depends heavily on the given probabilistic model, we illustrate some practical considerations involved in the design of a GCM algorithm using an example in this chapter.

Specific recommendations regarding how the parameter should be partitioned, selection of a reparameterization, and the method of approximation of the likelihood cannot be given since they are completely dependent on the statistical model since the correlations among parameters and the parameter space itself are different among models. This situation is similar to that of ECM or ECME algorithms (ECME will be introduced later). How to select parameter partition to be used for the CM steps in ECME or ECM, and how to decide if a CM-step maximizes $A_{EM}(\theta|\theta')$ or $l(\theta)$ in ECME are entirely dependent on the specified model and therefore decisions regarding the various choices could only be made using past experience and experimenting with available options.

But assuming that a general model is given, we can establish guidelines for designing a GCM algorithm by taking into account the performance of conventional algorithms, correlations of parameters among partitions in the sense of asymptotic variance-covariance matrix of MLE, and the need for balancing of speeds among different cycles of a GCM algorithm.

Section 4.2 presents a review of conventional algorithms for a given model as a prior study for development of new GCM algorithm. Section 4.3 includes a discussion of
options available in the selection of a partition of the parameter and for reparameterization. Some guidance of the selection of an approximation function for each partitioned block parameter vector is provided in Section 4.4, and the need for an R-step for each cycle in Section 4.5. Section 4.6 presents the developments of new GCM algorithms and results of simulation studies comparing them to conventional algorithms for five different ML estimation problems, i.e., ML estimation for random effect model, two-variance-component model, univariate t-model, contingency table with missing cell, and two-component Gaussian finite mixture model.

4.2 Review of Conventional Algorithms

Investigating algorithms already available for a given model is a good starting point in order to detect problems one might encounter in developing a GCM algorithm such as curse of dimensionality, slow convergence rate, tendency to diverge, etc.

Example Variance components model

Consider the problem of maximum likelihood estimation in a two-variance-component model, without loss of generality. Let \( y, \beta, b, \) and \( e \) be \( n \times 1, k \times 1, q \times 1, \) and \( n \times 1 \) vectors, respectively. The two-variance-component model is the following:

\[
y = X\beta + Zb + e, \tag{4.1}
\]

where \( b \) is a random vector independent of \( e, \) and distributed as \( N(0, \sigma_1^2 I_q), \) and the distribution of \( e \) is \( N(0, \sigma_2^2 I_n). \) The vector \( \beta \) is unknown fixed and \( X \) and \( Z \) are known \( n \times k \) and \( n \times q \) matrices. Although finding MLE for this model by quadratic approximation methods like Newton-Raphson and Fisher scoring requires computing \( H(\theta) \) and its inverse, linear search for monotone convergence and so on, Goodnight and Hemmerle(1979) reduces the computational burden via the use of the W-transformation. However, in order to avoid the possibility that the sequences from quadratic approximation methods converge to negative values, constraints adjustment must be incorporated into the usual
maximization procedure (Harville, 1977). Linearizations and reparameterizations of the quadratic approximation by Callanan and Harville (1991) leads to faster convergence of the algorithm for finding REML estimates, but its computational burden is heavy. EM algorithm for variance components has a simple iteration function which is equivalent to that of the quadratic approximations with diagonal Hessian matrix so that it requires only the first derivatives of the loglikelihood function to be computed. Furthermore, the sequence of variance components from EM never converges to a negative value. However, when one of the components converges to zero, the algorithm becomes very slow. Since MLE of residual variance component, $\sigma^2$ is in general, not zero, we can maximize $l(\theta)$ in terms of $\sigma^2$ given other parameters usually by applying several iterations of a quadratic approximation method instead of maximizing $A_{EM}(\theta|\theta')$ in each iteration. This type of scheme is called an ECME algorithm (Liu and Rubin, 1994). Consider the ECME as a GCM with 2 cycles (EM approximation for $\sigma^2$ and a quadratic approximation for $\sigma^2$), and with an R-step for $\sigma^2$. This GCM algorithm is an unbalanced iterative method since the quadratic approximation cycle has very small convergence rate while the EM approximation cycle has a relatively large convergence rate.

### 4.3 Partition and Reparameterization

The next step is to find a suitable partitioning of the parameter vector. Sound partitioning requires dividing the parameters into several block vectors which are not "correlated" with each other in the sense of asymptotic variance-covariance matrix of their MLE's. For an uncorrelated partitioning, reparameterization is necessary since conventional parameterization is usually correlated. Finding a suitable reparameterization is aided by the use of prior knowledge and/or graphical methods. A consideration for reparameterization is the resulting gain in computational efficiency. Reparameterizations that require heavy computational burden should be avoided.
Example continued: Although Callanan and Harville (1991) used sound reparameterizations in the variance components model, they can lead to a heavy computational burden. Since the correlation between ML estimates of $\sigma_1^2$ and $\sigma_2^2$ is not significant in general, we will continue to use this conventional parameterization and partition the parameter vector into a residual component and the other component as done in the ECME algorithm for this problem.

4.4 Selection of an Approximation Function

Determining an appropriate approximation for the reduced likelihood function corresponding to each block of the partitioned parameters is very important in establishing cycles of GCM. As in Chapter 1, an approximation can be selected by using the criterion \([a]-[e]\), but is mostly dependent on the probabilistic model. Thus theoretical knowledge, prior experience, and possibly results from previous simulation studies of the model may be necessary to help this selection process.

Example continued: As with the ECME algorithms explained above, we use the quadratic approximation, especially the Fisher scoring method for the residual component $\sigma_2^2$, and the EM approximation for the other component. In the context of GCM definition, the EM approximation is applied for cycle 1 corresponding to $\sigma_1^2$ while the quadratic approximation is used for cycle 2 corresponding to $\sigma_2^2$.

4.5 Decision for R-step

Deciding the number of repetitions for the R-step in each cycle is based on balancing convergence rates of the cycles of the GCM since convergence rate of GCM algorithm is asymptotically similar to that of the slowest cycle.

Example continued: In order to balance convergence rate between the two cycles, Unlike ECME, we propose to include an R-step for the EM approximation, but not the
R-step for the quadratic one. This differs from the ECME where for practical purpose, the number of repetitions of R-step is obtained from comparing convergence rates, and is about 1 or 2.

4.6 Numerical Examples

The version of ECME used in this section applies several iterations of a quadratic approximation method for maximizing the observed loglikelihood over a part of the parameter vector in each iteration. The number of the inner iterations within a iteration is, however, sufficient to be 1 or 2 for better efficiency of the ECME algorithm.

The purpose for comparing GCM algorithms to EM-type algorithms in this section is to illustrate the elements that need to be considered for designing a more efficient GCM algorithm. Especially comparisons with the ECME algorithm should be viewed in this sense since ECME already is GCM algorithm. Most of the software used in this section for performing the computations are developed in Fortran-77, and Splus language.

In Subsection 4.6.1, A GCM algorithm which is a PX-EM algorithm incorporating overrelaxation is developed and compared with the original PX-EM algorithm for the random effects model. Subsection 4.6.2 presents two modifications of the ECME algorithm in GCM framework and another GCM algorithm as an application of univariate linearization, for the variance component model. Subsection 4.6.3 introduces an ERCM algorithm which is a GCM algorithm and is compared with ECM. This ERCM is developed for ML problem in the contingency table with incomplete cells. In Subsection 4.6.4, modifications of the ECME algorithm for the univariate t-model are derived for ML estimation of unknown degrees of freedom. Finally, Subsection 4.6.5 presents a newly developed ELBQA algorithm for ML estimation in the Gaussian finite mixture model and compared it with the EM algorithm.
4.6.1 Random Effects Model with overrelaxation

In this example, we apply the overrelaxation method to an EM-type algorithm, the Parameter-Expanded EM (PX-EM) algorithm (Liu et al, 1997) to obtain MLEs of the random effects model. Briefly, PX-EM algorithm consists of an E-step, an M-step, and an adjustment step (A-step). Instead of deriving the algorithm from the conventional model as in EM, the E-step and the M-step in PX-EM algorithm use the parameter-expanded model where an identifiable parameter is added to the conventional model. And the A-step transforms the new parameterization back to the conventional parameterization to obtain the MLE's of the parameters of interest. This simple change in the algorithm improves the speed of the EM algorithm dramatically for several important models (Liu et al. 1997).

Random effects model is usually given by:

\[ y_i = X_i^T \beta + Z_i^T b_i + \epsilon_i, \quad (4.2) \]

where \( y_i, i = 1, \ldots, N \) is an \( n_i \times 1 \) vector, \( q \times 1 \) vectors \( b_i \sim N(0, \Sigma_b) \) which are independent of each other, \( \epsilon_i \sim N(0, \sigma^2) \), \( b_i \)'s are also independent of \( \epsilon_i \), \( \beta \) is the usual fixed effects, and \( X_i \) and \( Z_i \) are \( p \times n_i \) and \( q \times n_i \) known design matrices, respectively. Therefore \( y_i \sim N(X_i^T \beta, \sigma^2 I_{n_i} + Z_i^T T Z_i) \), which leads to the following loglikelihood.

\[ l(\theta) = -0.5 N \log 2\pi - 0.5 \sum_{i=1}^{N} \log |V_i| - 0.5 \sum_{i=1}^{N} (y_i - X_i^T \beta)^T V_i^{-1} (y_i - X_i^T \beta), \]

where \( V_i = \sigma^2 I_{n_i} + Z_i^T T Z_i \). And from Lindstrom and Bates(1988), first and second derivatives of the loglikelihood are

\[ \frac{\partial l(\theta)}{\partial \beta} = 0.5 \sum_{i=1}^{N} X_i V_i^{-1} (y_i - X_i^T \beta), \]

\[ \frac{\partial l(\theta)}{\partial \sigma^2} = -0.5 \sum_{i=1}^{N} tr(V_i^{-1}) + 0.5 \sum_{i=1}^{N} (y_i - X_i^T \beta)^T V_i^{-1} V_i^{-1} (y_i - X_i^T \beta), \]

\[ \frac{\partial l(\theta)}{\partial vec(T)} = -0.5 \sum_{i=1}^{N} vec(Z_i V_i^{-1} Z_i^T) \]
\[ +0.5 \sum_{i=1}^{N} \text{vec}[Z_i V_i^{-1}(y_i - X_i^T \beta)(y_i - X_i^T \beta)^T V_i^{-1} Z_i^T] \]

\[ \frac{\partial^2 l(\theta)}{\partial \beta \partial \beta^T} = -0.5 \sum_{i=1}^{N} X_i V_i^{-1} X_i^T, \]

\[ \frac{\partial^2 l(\theta)}{\partial \beta \partial \sigma^2} = -0.5 \sum_{i=1}^{N} X_i V_i^{-1} V_i^{-1}(y_i - X_i^T \beta), \]

\[ \frac{\partial^2 l(\theta)}{\partial (\sigma^2)^2} = 0.5 \sum_{i=1}^{N} tr(V_i^{-1} V_i^{-1}) \]

\[ -0.5 \sum_{i=1}^{N} (y_i - X_i^T \beta)^T V_i^{-1} V_i^{-1}(y_i - X_i^T \beta), \]

\[ \frac{\partial^2 l(\theta)}{\partial \beta \partial \text{vec}(T)^T} = -0.5 \sum_{i=1}^{N} X_i V_i^{-1} Z_i^T \otimes (y_i - X_i^T \beta)^T V_i^{-1} Z_i^T \]

\[ -0.5 \sum_{i=1}^{N} (y_i - X_i^T \beta)^T V_i^{-1} Z_i^T \otimes X_i V_i^{-1} Z_i^T \]

\[ \frac{\partial^2 l(\theta)}{\partial \text{vec}(T) \partial \sigma^2} = 0.5 \sum_{i=1}^{N} \text{vec}[Z_i V_i^{-1} V_i^{-1} Z_i^T] \]

\[ -0.5 \sum_{i=1}^{N} \text{vec}[Z_i V_i^{-1} V_i^{-1}(y_i - X_i^T \beta)(y_i - X_i^T \beta)^T V_i^{-1} Z_i^T] \]

\[ -0.5 \sum_{i=1}^{N} \text{vec}[Z_i V_i^{-1}(y_i - X_i^T \beta)(y_i - X_i^T \beta)^T V_i^{-1} V_i^{-1} Z_i^T] \]

\[ \frac{\partial^2 l(\theta)}{\partial \text{vec}(T) \partial \text{vec}(T)^T} = 0.5 \sum_{i=1}^{N} Z_i V_i^{-1} Z_i^T \otimes Z_i V_i^{-1} Z_i^T \]

\[ - \sum_{i=1}^{N} Z_i V_i^{-1} Z_i^T \otimes Z_i V_i^{-1}(y_i - X_i^T \beta)(y_i - X_i^T \beta)^T V_i^{-1} Z_i^T, \]

where \( \text{vec}(A) \) is a vector whose \((j - 1)n + i\) element is the \(ij\) element of a \(m \times n\) matrix \(A\), and \(\otimes\) is the Kronecker product. As Lindstrom and Bates(1988) point out, conventional Newton’s method without constraint on \(T\) may produce nonpositive definite estimate of \(T\) and its convergence often fails. Furthermore, EM algorithm derived by setting \(b_i\)'s as missing data, might lead to a convergence rate so slow that it is nearly impossible for the iterate to reach a local maximum. In this regards, Lindstrom and Bates(1988) develop an improved Newton’s algorithm maximizing a profile likelihood function where \(T\) is transformed to \(L\), the Cholesky factor of \(T\), i.e., \(T = LL^T\). The
PX-EM algorithm for the random effects model can be considered as the EM-version of this improved algorithm. It does not yet outperform the modified Newton’s algorithm due to Lindstrom and Bates (1988) with its fast convergence rate, but the wide range of possible initial values of parameters makes the algorithm competitive with the Newton’s algorithm.

For simulation studies to investigate the effects of overrelaxation, we consider the following model used by Meng and van Dyk (1997):

\[ y_i = x_{i1} \beta_1 + x_{i2} \beta_2 + z_{i1} b_{i1} + z_{i2} b_{i2} + \epsilon_i, \quad (4.3) \]

where \( x_{i1} = 1, x_{i2} = i \left( \begin{array}{c} b_{i1} \\ b_{i2} \end{array} \right) \sim N_2(0, \left( \begin{array}{cc} 4 & 0 \\ 0 & 9 \end{array} \right)) \) and \( \epsilon_i \sim N(0, \sigma^2) \) with \( b_i = (b_{i1}, b_{i2})^T \) and \( \epsilon_i \) independent.

Defining \( X_i = (x_{i1}, x_{i2})^T, \beta = (\beta_1, \beta_2)^T, \) and \( Z_i = (z_{i1}, z_{i2})^T \) leads to the matrix expression of (4.3)

\[ y_i = X_i^T \beta + Z_i^T b_i + \epsilon_i, \quad (4.4) \]

where \( b_i \sim N(0, T) \), \( \epsilon_i \sim N(0, \sigma^2) \), \( b_i \perp \epsilon_i \), and \( T = \left( \begin{array}{cc} 4 & 0 \\ 0 & 9 \end{array} \right) \). Since the joint distribution of \((y_i, b_i)\) belongs to exponential family, EM algorithm can be implemented easily, but terribly slow when \( \sigma^2 \) is large (Liu et al, 1997).

Instead of using (4.4), PX-EM algorithm for this problem is derived from its PX model,

\[ y_i = X_i^T \beta_+ + Z_i^T A b_i + \epsilon_i, \quad (4.5) \]

where \( A \) is 2 \times 2 matrix form of the expanded parameter, \( b_i \sim N(0, T_*) \), \( \epsilon_i \sim N(0, \sigma^2_*) \), and \( b_i \perp \epsilon_i \). Under the model (4.5), the E-step and the M-step are obvious and the assumption that the model (4.5) is identical to (4.4) leads to the A-step of PX-EM where \( \beta = \beta_*, \sigma^2 = \sigma^2_*, \) and \( T = AT_+ A^T \) (For more details, see Liu et al (1997)).
Thus, each iteration of PX-EM consists of following three steps (Liu et al., 1997):

**E-step** Compute conditional expectations of sufficient statistics of missing data:

\[
E(b_i | y_i, \theta^{(t)}) = \frac{y_i - X_i^T \theta^{(t)}}{\sigma^2(t) + Z_i^T T^{(t)} Z_i} T^{(t)} Z_i,
\]

\[
E(b_i | y_i, \theta^{(t)}) = E(b_i | y_i, \theta^{(t)})[E(b_i | y_i, \theta^{(t)})]^T + T^{(t)} - \frac{T^{(t)} Z_i Z_i^T T^{(t)}}{\sigma^2(t) + Z_i^T T^{(t)} Z_i},
\]

where \( \theta = (\beta, \sigma^2, T) \).

**M-step** Since we can write (4.5) in the linear regression form

\[
y_i = X_i^T \beta + (\text{vec}(Z_i b_i^T))^T \text{vec}(A) + \epsilon_i,
\]

we can find \( \theta^{(t+1)} = (\beta, \sigma^2, \text{vec}(A)) \) by the usual regression procedure. And Compute

\[
T^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} E(b_i | y_i, \theta^{(t)}).
\]

**A-step** set \( \beta^{(t+1)} = \beta^{(t+1)}, \sigma^{2(t+1)} = \sigma^{2(t+1)}, \) and \( T^{(t+1)} = AT^{(t+1)}A^T \).

Computationally, the E-step and the M-step can be performed using the sweep operator. Sweep operator (Beaton, 1964) transforms

\[
U = \begin{bmatrix}
U_{11} & U_{12} \\
U_{21} & U_{22}
\end{bmatrix}
\]
to the form

\[
\begin{bmatrix}
-U_{11}^{-1} & U_{11}^{-1} U_{12} \\
U_{21} U_{11}^{-1} & U_{22} - U_{21} U_{11}^{-1} U_{12}
\end{bmatrix}
\]

by repeating elementary row/column operations without using extra storage. This operator mainly plays a main role in regression analysis by computing estimates of linear regression coefficient, residual sum of squares, and the variance-covariance matrix of the coefficient vector, simultaneously. For the E-step, \( E(b_i | y_i) = V_i^{-1} Z_i^T T (y_i - X_i^T \beta) \) and \( V(b_i | y_i) = T - T Z_i V_i^{-1} Z_i^T T \) are easily computed by setting

\[
U = \begin{bmatrix}
V_i & Z_i^T T \\
T Z_i & T
\end{bmatrix}
\]

and by performing sweeps to obtain

\[
\begin{bmatrix}
-V_i^{-1} & V_i^{-1} Z_i^T T \\
T Z_i V_i^{-1} & T - T Z_i V_i^{-1} Z_i^T T
\end{bmatrix}
\]
Since the M-step of PX-EM above is the same as finding least squares solution of linear regression with coefficient \((\beta, vec(A))\), by setting \(U = \sum_{i=1}^{N} E(D_i^T D_i | y_i, \theta)\) where \(D_i = \left(X_i^T, (vec(Z_i b_i^T))^T, y_i\right)\), the parameter vector \((\beta, vec(A), \sigma^2)\) can be easily computed. The sweep operator has, however, somewhat of a bad reputation in some cases producing unreliable values in least square problems. For increased accuracy, it is possible to do several additional iterations of PX-EM using the singular value decomposition, a reliable numerical method instead of using the sweep operator.

Overrelaxation is included with PX-EM in addition to the three steps above, to provide a new GCM algorithm. Just add the following step: Overrelaxation step

\[ \theta^{*\left(t+1\right)} = \omega^{(t)} \theta^{(t+1)} + (1 - \omega^{(t)}) \theta^{(t)}, \]  

(4.7)

where \(\theta^{*\left(t+1\right)}\) is the improved updates and \(\omega^{(t)}\) is depending on \(\lambda^{(t)} = (l(\theta^{\left(t+1\right)}) - l(\theta^{(t)}))/\left(l(\theta^{(t)}) - l(\theta^{\left(t-1\right)})\right)\) as follows:

\[ \omega^{(t)} = \begin{cases} 
1 & \text{for } t \text{ such that } \lambda^{(t)} - \lambda^{(t-1)} > 0.001, \\
1 + \lambda^{(t_0)} & \text{for } t \geq t_0,
\end{cases} \]  

(4.8)

where \(t_0\) is the first \(t\) holding \(\lambda^{(t)} - \lambda^{(t-1)} < 0.001\).

That is, each iteration of new GCM algorithm has one cycle whose

**Approximation step** is established by conditional expectation of complete-data \((b_i\) and \(y_i, i = 1, \cdots, N\) loglikelihood given observed data \(y_is\) with parameter-expanded model (4.5),

**One CM step** finds parameter vector value maximizing the conditional expectation of complete data loglikelihood above, and

**Overrelaxation step** as in (4.7).

We compared the log-scaled number of iterations between PX-EM and PX-EM with overrelaxation with 100 sets of randomly generated 100 observations from model (4.4) for each value of \(\sigma^2 = 0.05, 0.1, 0.25, 1, 5, 10, 20, 25\). Only the number of iterations rather
than CPU time needs to be compared due to the fact that overrelaxation does not require any significant amount of computation.

We use 0.001 as the negligible difference between previous and current numerical convergence rates which is the criterion for starting overrelaxation step. As expected, the sequences from new GCM algorithm are monotonely increasing in our all simulations. The simulation study is summarized in Figure 4.1. Over all 8 simulations, overrelaxation method leads to slightly faster convergence than PX-EM. The new algorithm takes only 60 percent of the number of iterations required for convergence of the pure PX-EM algorithm.

**4.6.2 Variance Components Model (Two components)**

In order to avoid any confusion, we assume that the phrase 'variance components model' imply the model (4.1) throughout this paper. A complete description and extension of this model is in Chapter 5. Implementing PX-EM as done in the case of random effects model is not applicable in this model, even in the model with two variance components since we have to solve equation $AT^2 = \sigma^2_I$ in each iteration in order to update the estimator of variance of the $b_1$ vector. Instead we shall compare ECME with a GCM algorithm incorporating overrelaxation (labeled here as GCM-1) which was introduced as an example of the steps involved in designing a GCM algorithm. In order to avoid slow progress of simulation study, We shall exclude the EM algorithm which is known to be slower than ECME from our simulation study.

The use of overrelaxation in GCM-1 eliminates the problem associated with other EM-type algorithms of it becoming too slow when the MLE of $\sigma^2_I$ is close to 0. Another approach to overcome this problem is by using partial univariate Aitken acceleration, that is, accelerating the sequence of only a part of the parameter vector, $\{\sigma^{2(t)}_1\}$ rather than the whole parameter vector, $\{\sigma^{2(t)}_1, \sigma^{2(t)}_2\}$ by using the convergence rate of the inner iteration for $\{\sigma^{2(t)}_1\}$. We shall label the resulting algorithm GCM-2 The partial Aitken
Figure 4.1 Simulation Study 1 $\log(PX - EM)$ and $\log(PX - EM - OR)$ Indicate the Log-scaled Number of Iterations of Pure PX-EM and PX-EM with Overrelaxation, Respectively. the Solid Line Denotes the Line $\log(PX - EM) = \log(PX - EM - OR)$
acceleration is more likely to retain the monotone convergence of the sequence of GCM iterates than the Aitken acceleration of the complete parameter vector. In 500 simple simulation studies, it is seen that GCM-2 converges monotonely 100% of the time while about 3% of full (univariate) Aitken acceleration fails to do so.

The standard ECME algorithm for the variance components model consists of two cycles as follows:

**Cycle 1 for \( \sigma_1^2 \):** EM approximation with no R-step.

\[
q \sigma_1^{2(1)} = q \sigma_1^{2(0)} + \sigma_1^{4(0)} y^T P^{(0)} ZZ^T P^{(0)} y - tr[\sigma_1^{4(0)} Z^T (V^{(0)})^{-1} Z], \quad (4.9)
\]

where \( V^{(0)} = ZZ^T \sigma_1^{2(0)} + I_n \sigma_2^{2(0)} \).

and \( P^{(0)} = (V^{(0)})^{-1} - (V^{(0)})^{-1} X [X^T (V^{(0)})^{-1} X]^{-1} X^T (V^{(0)})^{-1} \).

**Cycle 2 for \( \sigma_2^2 \):** Quadratic approximation (Fisher scoring) with no R-step (i.e., \( r = 1 \))

\[
\sigma_2^{2(2)} = \sigma_2^{2(0)} + \frac{0.5}{tr[(V^{(1)})^{-1} (V^{(1)})^{-1}]} \{ y^T P^{(1)} P^{(1)} y - tr[(V^{(1)})^{-1}] \}. \quad (4.10)
\]

where \( V^{(1)} = ZZ^T \sigma_1^{2(1)} + I_n \sigma_2^{2(0)} \) and

\[
P^{(1)} = (V^{(1)})^{-1} - (V^{(1)})^{-1} X [X^T (V^{(1)})^{-1} X]^{-1} X^T (V^{(1)})^{-1}.
\]

Of course, MLE of \( \beta \) can be obtained after convergence of the algorithm by setting

\[
\hat{\beta} = [X^T (V^{(\infty)})^{-1} X]^{-1} X^T (V^{(\infty)})^{-1} y,
\]

where \( V^{(\infty)} = ZZ^T \sigma_1^2 + I_n \sigma_2^2 \), \( \sigma_1^2 \) and \( \sigma_2^2 \) are corresponding MLEs. GCM-1 also has two cycles with Cycle 1 having an R-step (with number of repetition=2). The additional overrelaxation step given in (4.7) with \( \theta = (\sigma_1^2, \sigma_2^2) \) takes place after both cycles.

The GCM-2 algorithm is the same as GCM-1 except that a partial Aitken acceleration step is inserted between **Cycle 1** and **Cycle 2** as following:

**Cycle 1 for \( \sigma_1^2 \):** EM approximation with one R-step (\( r = 2 \)).

\[
q \sigma_1^{2(1)} = q \sigma_1^{2(0)} + \sigma_1^{4(0)} y^T P^{(0)} ZZ^T P^{(0)} y - tr[\sigma_1^{4(0)} Z^T (V^{(0)})^{-1} Z], \quad (4.11)
\]
where $V^{(0)} = ZZ^T \sigma_1^{2(0)} + I_n \sigma_2^{2(0)}$, 

and $P^{(0)} = (V^{(0)})^{-1} - (V^{(0)})^{-1} X [X^T (V^{(0)})^{-1} X]^{-1} X^T (V^{(0)})^{-1}$.

**Cycle 2 (partial Aitken step)**

$$
\sigma_1^{2(2)} = \omega^{(t)} \sigma_1^{2(2*)} + (1 - \omega^{(t)}) \sigma_1^{2(1*)},
$$

(4.12)

where $\sigma_1^{2(1*)}$ is the update of the first run of Cycle 1, $\sigma_1^{2(2*)}$ is the update of the second run of Cycle 1, and $\omega^{(t)}$ depends on $\phi^{(t)} = (\sigma_1^{2(2*)} - \sigma_1^{2(1*)}) / (\sigma_1^{2(1*)} - \sigma_1^{2(0)})$ as follows:

$$
\omega^{(t)} = \begin{cases} 
1 & \text{for } t \text{ such that } \phi^{(t)} - \phi^{(t-1)} > 0.001, \\
\frac{1}{1 + \phi^{(t)}} & \text{otherwise,}
\end{cases}
$$

(4.13)

**Cycle 3 for $\sigma_2^2$: Quadratic approximation (Fisher scoring) with no R-step (i.e., $r=1$)**

$$
\sigma_2^{2(3)} = \sigma_2^{2(0)} + \frac{0.5}{\text{tr}([V^{(2)})^{-1} (V^{(2)})^{-1}]} \{y^T P^{(2)} y - \text{tr}([V^{(2)})^{-1}]}.
$$

(4.14)

where $V^{(2)} = ZZ^T \sigma_1^{2(2)} + I_n \sigma_2^{2(0)}$ and 

$$
P^{(2)} = (V^{(2)})^{-1} - (V^{(2)})^{-1} X [X^T (V^{(2)})^{-1} X]^{-1} X^T (V^{(2)})^{-1}.
$$

**Overrelaxation step**

On experimenting with the variance components model we found that each of 

$$
\frac{\partial l(\theta)}{\partial \sigma_i^2}, \quad i = 1, 2
$$

can be linearized using the linearization factor $\beta_i^4$. The details about this linearization is described in Chapter 5. This discovery leads to another GCM algorithm, GCM-3, which only has one cycle of Jacobi-type quadratic approximation with no R-step (repeating...
number=1) whose $t$th iteration is as follows:

$$
\sigma_1^{2(t+1)} = \sigma_1^{2(t)} - \frac{\sigma_1^{4(t)} \frac{\partial l(\theta)}{\partial \sigma_1^2} \bigg|_{\sigma_1^2=\sigma_1^{2(t)}}}{2\sigma_1^{2(t)} \frac{\partial l(\theta)}{\partial \sigma_1^2} \bigg|_{\sigma_1^2=\sigma_1^{2(t)}} + \sigma_1^{4(t)} \frac{\partial l(\theta)}{(\sigma_1^2)^2} \bigg|_{\sigma_1^2=\sigma_1^{2(t)}}^{2(t)}}
$$

$$
\sigma_2^{2(t+1)} = \sigma_2^{2(t)} - \frac{\sigma_2^{4(t)} \frac{\partial l(\theta)}{\partial \sigma_2^2} \bigg|_{\sigma_2^2=\sigma_2^{2(t)}}}{2\sigma_2^{2(t)} \frac{\partial l(\theta)}{\partial \sigma_2^2} \bigg|_{\sigma_2^2=\sigma_2^{2(t)}} + \sigma_2^{4(t)} \frac{\partial l(\theta)}{(\sigma_2^2)^2} \bigg|_{\sigma_2^2=\sigma_2^{2(t)}}^{2(t)}}
$$

Consider the following simulation study, where we randomly generate data from (4.1) with $n = 30$, $q = 10$, known $\beta = 0$, $Z = ((Z_{ij})), Z_{ij} \sim \text{unif}(0, 1)$ $b \sim N(0, I_q)$, with a value selected for $\sigma_2^2$ from 0.05, 0.5, 1, 5, and 25. Initial value selected for $(\sigma_1^2, \sigma_2^2)$ is (10, 10) and the criterion of convergence is $||l(\theta^{(t)}) - l(\theta^{(t-1)})|| < 10^{-8}$. Table 4.1 shows the comparison of performance among the ECME, GCM-1, GCM-2 and GCM-3 algorithms for each selected value of $\sigma_2^2$ in terms of the number of iterations averaged over 10 simulations.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>0.05</th>
<th>0.5</th>
<th>1</th>
<th>5</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECME</td>
<td>23.2</td>
<td>20.7</td>
<td>21.6</td>
<td>80</td>
<td>4565.7</td>
</tr>
<tr>
<td>GCM-1</td>
<td>16.7</td>
<td>14.2</td>
<td>15.4</td>
<td>27.6</td>
<td>2367.6</td>
</tr>
<tr>
<td>GCM-2</td>
<td>17.2</td>
<td>17.5</td>
<td>17.6</td>
<td>15.6</td>
<td>23.4</td>
</tr>
<tr>
<td>GCM-3</td>
<td>5.3</td>
<td>5.4</td>
<td>5.9</td>
<td>7.4</td>
<td>11.6</td>
</tr>
</tbody>
</table>

Since the computational times for each iteration of the four algorithms are similar to each other, comparisons are made using the number of iterations to determine a local maximum. As was expected, if the true value of $\sigma_2^2$ is large relatively to that of $\sigma_1^2$, ML estimator of $\sigma_1^2$ approaches 0, and thus ECME and GCM-1 algorithms, which are simple modifications of the EM algorithm, become very slow. But GCM-2 does not appear to lose its speed even in these cases. An interesting fact is that GCM-2 is not always faster than GCM-1 in spite of the fact that GCM-2 resulted from adding an acceleration procedure to one of the iterative steps of GCM-1. The reason for this is that performing
an inner acceleration leads to a slow start of overrelaxation procedure so that GCM-2 becomes slower than GCM-1 when GCM-1 is sufficiently fast. We obtained similar result for this model in the case of more than two components.

The results of GCM-3 are spectacular. It does not require computation of the inverse of Hessian matrix, except for the diagonals elements. Also it does not require a linear search in each iteration to ensure monotone convergence. In spite of this, GCM-3 converges very quickly to the local maximum similar to a well-designed quadratic approximation method, e.g., Fisher Scoring, as shown in Table 4.1. Unlike common quadratic approximation algorithms, however, it does not converge to negative variance estimates. In conclusion, it appears that GCM-3 is the winner, but more extended comparison between GCM-3 and other quadratic algorithms is needed.

4.6.3 Contingency Table with Incomplete Cells

The $2 \times 2 \times 2$ contingency table without three-way interaction is a typical example for showing usefulness of the ECM algorithm. As in Table 4.2, our data from Little and Rubin(1987, p. 187) consists of two parts, first $\{y_{ijk}^a\}$ is completely classified, and second of which $\{y_{ijk}^b\}$ is partially classified.

Since there is no closed form solution for MLE of cell probabilities with this type of data, an iterative method must be used. Following the notations from Meng and Rubin(1993), let $\theta_{ijk}$ be the cell probability in cell $ijk$, $(i, j, k = 1, 2)$ and $x_{ijk}$ be the complete data in cell $ijk$. Because complete-data loglikelihood is linear in $x_{ijk}$'s, computing

$$x_{ijk}^{(t)} = E(x_{ijk}|y_{ijk}^a, y_{ijk}^b, \theta^{(t)}) = y_{ijk}^a + y_{ijk}^b \frac{\theta_{ijk}^{(t)}}{\sum_j \theta_{ijk}^{(t)}} \quad (4.15)$$

is only necessary for the E-step in the $t$th iteration. And the following are the three CM-steps in the $t$th iteration:

$$CM_1: \theta_{ijk}^{(t+\frac{1}{2})} = \theta_{ijk}^{(t)} \frac{x_{ij}^{(t)}}{N},$$
\[ CM_2 : \theta^{(t+\frac{1}{2})}_{ijk} = \theta^{(t+\frac{1}{2})}_{(i)jk} \frac{x^{(t)}_{ijk}}{N}, \]
\[ CM_3 : \theta^{(t+\frac{1}{2})}_{ijk} = \theta^{(t+\frac{1}{2})}_{(i)jk} \frac{x^{(t)}_{ijk}}{N}, \]

where \(N\) is the total count, \(\theta_{(i)jk} = \frac{\theta_{ijk}}{\sum_i \theta_{ijk}}\), and \(x_{ijk} = \sum_i x_{ijk}\). (Derivation of this form of M-step follows from the constrained complete-data likelihood equation of Meng and Rubin(1991)) The standard ECM algorithm follows the E-step by the 3-CM steps. The first GCM algorithm proposed (labeled ERCM) includes an R-step (with \(r = 2\)), i.e., the 3-CM steps are repeated once. Since the EM approximation function is easily verified to be concave, the second GCM algorithm (labeled ERCMO) is a modification of ERCM that implements an overrelaxation step.

Table 4.2 A 2 \times 2 \times 2 contingency table with partially classified observations from Little and Rubin(1987, p. 187)

<table>
<thead>
<tr>
<th>Clinic (C)</th>
<th>Prenatal care (P)</th>
<th>Survival (S)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Died</td>
<td>Survived</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>(a) Completely classified cases</td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>A</td>
<td>Less</td>
<td>3</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td></td>
<td>More</td>
<td>4</td>
<td>293</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Less</td>
<td>17</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td></td>
<td>More</td>
<td>2</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(N^{(c)} = 715) cases</td>
<td></td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>(a) Partially classified cases</td>
<td></td>
<td></td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>?</td>
<td>Less</td>
<td>10</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>More</td>
<td>5</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(N^{(b)} = 255) cases</td>
<td></td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

We compare the above three algorithms, ECM, ERCM, and ERCMO with this example given in Table 4.2. The result is shown in Table 4.3 The table does not indicate much of improvement in efficiency of ERCM compared to the ECM algorithm. Also ERCM with overrelaxation algorithm reduces the total computation time of ECM algorithm only by 10%. The reason is that computation of the E-step in this model is not much heavier than that of all of the CM steps combined.

ERCMO algorithm can be applied for the supplemented algorithm for asymptotic variance-covariance matrix. In this model, it is hard to find analytic asymptotic variance-
Table 4.3  Comparisons of the number of iterations and CPU time per iteration among ECM, ERCM, and ERCM with overrelaxation (ERCMO)

<table>
<thead>
<tr>
<th></th>
<th>ECM</th>
<th>ERCM</th>
<th>ERCMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>35</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td>Time</td>
<td>0.07142857</td>
<td>0.1166667</td>
<td>0.1210526</td>
</tr>
</tbody>
</table>

covariance matrix $D^2l(\theta^*)$ due to the complexity of the model, where $\theta^*$ is MLE of the parameter vector. Thus, van Dyk et al (1995) use the equations

$$D_{MECM} = I - (I - DM_{CM})(I - DM_{EM}),$$

and

$$D_{MEM} = I - [D^{20}A_{EM}(\theta^*|\theta^*)]^{-1}D^2l(\theta^*)$$

to have

$$D^2l(\theta^*) = D^{20}A_{EM}(\theta^*|\theta^*)(I - DM_{CM})^{-1}(I - DM_{EM}),$$

(4.16)

where $D_{MECM}$, $D_{MEM}$, and $D_{CM}$ are the convergence rate matrices of ECM, EM and CM algorithms, respectively. (CM algorithm can be considered as ECM algorithm with complete data and thus, no E-step.) Thus, by measuring numerical matrices of convergence rate of ECM and CM, we can compute the asymptotic variance-covariance matrix which is the inverse of $-D^2l(\theta^*)$

Since the convergence matrix of ERCM with overrelaxation is similar in structure to that of ECM and is given by

$$D_{MERCW(o)} = I - \omega(I - DM_{CM}DM_{CM})(I - DM_{EM}),$$

$I_{obs}(\theta) = -D^2l(\theta)$ can be computed as

$$I_{obs}(\theta) = -\frac{1}{\omega}D^{20}A(\theta|\theta)(I - DM_{CM}DM_{CM})^{-1}(I - DM_{ERCW(o)}).$$

Since an overrelaxation-installed algorithm appeared to be faster than usual algorithm with asymptotically twice speed, the supplemented algorithm for asymptotic variance-covariance matrix using ERCMO is expected to converge about twice as fast as the supplemented algorithm using ECM (van Dyk et al, 1995).
4.6.4 Univariate T-model with Unknown Degrees of Freedom

We consider the t-model denoted by \( t_v(\mu, \sigma^2) \) and defined as

\[
y_i | \tau_i, \mu, \sigma^2 \sim N(\mu, \sigma^2/\tau_i) \quad (4.17)
\]

\[
\tau_i | v \sim \chi^2_v / v, \quad i = 1, \ldots, n \quad (4.18)
\]

If we assume that the degrees of freedom, \( v \) is known, MLE estimation of \( \mu \) and \( \sigma^2 \) is very easy and fast by using the EM algorithm with optimal augmentation (Meng and van Dyk, 1997) or PX-EM (Liu et al, 1997). Both of these methods use different approaches to derive exactly the same procedure that results in the following algorithm:

The \( t \)th iteration of the EM algorithm with optimal augmentation, or the PX-EM algorithm for determining the MLEs of \( \mu \) and \( \sigma^2 \) with \( v \) fixed is

\[
\mu^{(t+1)} = \frac{\sum_{i=1}^{n} w_i^{(t)} y_i / \sum_{i=1}^{n} w_i^{(t)}}{\sum_{i=1}^{n} w_i^{(t)}},
\]

\[
\sigma^{2(t+1)} = \frac{\sum_{i=1}^{n} w_i^{(t)} (y_i - \mu^{(t+1)})^2 / \sum_{i=1}^{n} w_i^{(t)}}{\sum_{i=1}^{n} w_i^{(t)}},
\]

where \( w_i^{(t)} = \frac{1}{\delta_i^{(t)} + v} \), \( i = 1, \ldots, n \), and \( \delta_i^{(t)} = (y_i - \mu^{(t)})^2 / \sigma^{2(t)} \);

However, when \( v \) is unknown, we have to maximize the loglikelihood

\[
\log L(\theta | y) = -0.5n \log(\sigma^2) + 0.5nv \log(v) + n \log \Gamma\left(\frac{v+1}{2}\right) - n \log \Gamma\left(\frac{v}{2}\right)
\]

\[-0.5(v+1) \sum_{i=1}^{n} \log(v + \delta_i),
\]

where \( \Gamma(\cdot) \) denotes the gamma function, \( \theta = (\mu, \sigma^2, v) \), and \( \delta_i = (y_i - \mu)^2 / \sigma^2 \), \( i = 1, \ldots, n \). Estimation of \( \theta \), especially \( v \) by EM or by another GEM algorithm is time consuming (Liu, 1994). Thus Liu suggests the algorithm named Expectation Conditional Maximization Either (ECME) for this estimation problem. ECME algorithm has two cycles, one iteration of EM steps for \( \mu \) and \( \sigma^2 \) given \( v \), several iterations of an one dimensional search algorithm such as the interval halving method (Carnahan et al, 1969).
for \( v \) given \( \mu \) and \( \sigma^2 \). The several inner iterations for \( v \) are intended to maximize the observed loglikelihood in terms of \( v \) with \( \mu \) and \( \sigma^2 \) fixed in each iteration. The reason that a quadratic approximation method is not used for this purpose may be due to the fact that the conventional parameterization of \( v \) leads to the failure of monotone convergence and a different parameterization can not be easily determined.

We propose a GCM algorithm (labeled GCM-1) which consists of two cycles, one iteration of EM algorithm for \( \mu \) and \( \sigma^2 \), and one iteration of a linearized quadratic approximation step, i.e., Newton-Raphson for \( v \). When the linearization for this model was discussed in Chapter 3, the linearized quadratic approximation algorithm for \( v \) suggested \( v^3 \frac{\partial^3(\ell)}{\partial v^3} \) and \( 2v^2 \frac{\partial^2(\ell)}{\partial v^2} + v^3 \frac{\partial^3(\ell)}{\partial v^3} \) as the score function and Hessian function, respectively. This is incorporated in A-step of GCM-1.

Overrelaxation is also implemented in a second GCM algorithm (say, GCM-2) where the other steps are the same as GCM-1. Since Liu’s ECME algorithm obviously less efficient than our GCM algorithms, instead of using the ECME algorithm above, we redefine ECME algorithm as a GCM algorithm which has two cycles, one iteration of EM steps for \( \mu \) and \( \sigma^2 \), and two iterations of our quadratic approximation algorithm for \( v \).

Although values of \( \mu \) and \( \sigma^2 \) do not have an effect on the performance of the GCM algorithms, iterates of \( v \) could converge to \( \infty \). (with this infinite value of \( v \), the t-model implies normal distribution model). So we set up our algorithms to stop and set \( v = \infty \) whenever current iterate of \( v \) is over a large value, say 10000.

The ECME algorithm for this model has two cycles,

**Cycle 1**: EM approximation with no R-step

\[
\mu^{(1)} = \frac{\sum_{i=1}^{n} w_i^{(0)} y_i}{\sum_{i=1}^{n} w_i^{(0)}},
\]

\[
\sigma^2^{(1)} = \frac{\sum_{i=1}^{n} w_i^{(0)} (y_i - \mu^{(1)})^2}{\sum_{i=1}^{n} w_i^{(0)}},
\]
where \( w_i^{(0)} = \frac{\mu_i^{(0)} + v}{\delta_i^{(0)} + v}, i = 1, \ldots, n, \) and \( \delta_i^{(t)} = \frac{(y_i - \mu_i^{(t)})^2}{\sigma_i^{(t)}}; \)

**Cycle 2: Quadratic approximation with one R-step**

\[
v^{(1)} = v^{(0)} - s(\theta^{(1)})/h(\theta^{(1)}),
\]

where

\[
\theta^{(1)} = (\mu^{(1)}, \sigma^{2(1)}, v^{(0)}),
\]

\[
s(\theta) = v^3 \frac{\partial l(\theta)}{\partial v},
\]

and

\[
h(\theta) = 3v^2 \frac{\partial l(\theta)}{\partial v} + v^3 \frac{\partial^2 l(\theta)}{\partial v^2}.
\]

GCM-1 is the same as above with an R-step \((r=2)\) in Cycle 1. GCM-2 is also the same with an R-step \((r=2)\) in Cycle 1 and an overrelaxation step.

In order to perform a simulation study to compare the three algorithms, GCM-1, GCM-2, ECME, we generated 100 data sets of 100 random observation from the t-model with \( \mu = 5, \sigma^2 = 10 \) for each of selected values for the degrees of freedom, \( v = 2, 3, 5, 7, 10 \). The algorithm were executed until convergence was reached, when the MLEs of \( \theta \) was obtained for each data set.

Table 4.4 shows the results of the simulation which consist of two parts, the average number of iterations and total CPU time. As expected, GCM-2 which is implemented with overrelaxation has a slight advantage over GCM-1 and ECME for the various values of \( v \). Although the actual total CPU time depends on computing platform, all three algorithms display adequate speed of convergence.

### 4.6.5 Normal Finite Mixture Model with Two components

The normal (Gaussian) finite mixture model is indispensable in applied statistics. We develop a new GCM algorithm for the model with two components. The loglikelihood
Table 4.4 Comparisons of the number of iterations and total CPU Time among GCM-1, GCM-2, and ECME

<table>
<thead>
<tr>
<th></th>
<th>Number of Iteration</th>
<th>Total CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GCM-1</td>
<td>GCM-2</td>
</tr>
<tr>
<td>2</td>
<td>31.17</td>
<td>23.74</td>
</tr>
<tr>
<td>3</td>
<td>28.49</td>
<td>22.21</td>
</tr>
<tr>
<td>5</td>
<td>24.58</td>
<td>19.26</td>
</tr>
<tr>
<td>7</td>
<td>21.95</td>
<td>17.43</td>
</tr>
<tr>
<td>10</td>
<td>18.91</td>
<td>15.27</td>
</tr>
</tbody>
</table>

of the normal (Gaussian) finite mixture model with two components is

\[ l(\theta) = \sum_{i=1}^{n} \log \left( p \phi(y_i|\mu_1, \Sigma_1) + (1 - p) \phi(y_i|\mu_2, \Sigma_2) \right), \]  

(4.19)

where \( \phi(\cdot|\mu, \Sigma) \) is the density function of \( N(\mu, \Sigma) \), and \( \theta = (p, \mu_1, \mu_2, \Sigma_1, \Sigma_2) \).

The EM algorithm for this model (Dempster et al, 1977) has been known to be an efficient method for determining MLEs of \( \theta \) since users of any quadratic approximation method are confronted with the dimensionality problem and with failure of monotone convergence (Everitt and Hand, 1980). However, one of the disadvantages of the EM algorithm for this model is its slow convergence.

In order to overcome this deficiency of the EM algorithm, we develop a GCM algorithm which also has two cycles, one iteration of EM algorithm for \( \mu_j \) and \( \Sigma_j \), \( j = 1, 2 \), and one iteration of a reparameterized quadratic approximation algorithm for \( p \). We reparameterize \( p \) to \( \tau = \log \frac{p}{1-p} \) since the first derivative function of loglikelihood in terms of \( \tau \) is locally linear as shown in Figure 4.2, and any adjustment for constrained optimization can be avoided. But the second derivative function of loglikelihood is not negative for all \( \tau \) in its space and then it may not converge to a local maximum since positive second derivative leads to the next iterate \( \theta^{(t+1)} \) which results in \( l(\theta^{(t+1)}) < l(\theta^{(t)}) \).

Thus, instead of the second derivative, we use a lower bound of the second derivative as the Hessian function. As the last columns of Figure 4.2 indicates, this is negative for all \( \tau \) and is close to the second derivative in the neighborhood of a mode.
Figure 4.2 The Surfaces of Loglikelihood, First, Second Derivative (Thick Lines) and Negative Hessian Function (Thin Lines) in Terms of $\tau$ for Various True $p = 0.2, 0.4, 0.7,$ and $0.9$
An interesting feature is that the Gauss-Seidel iteration over the two cycles is not efficient in this case. That is, updating each of \( \tau^{(t)}, \mu_j^{(t)}, \) and \( \Sigma_j^{(t)}, j = 1,2 \) involves computations of the same quantities

\[
p^{(t-1)} \phi(y_i | \mu_1^{(t-1)}, \Sigma_1^{(t-1)}) \text{ and } (1 - p^{(t-1)}) \phi(y_i | \mu_2^{(t-1)}, \Sigma_2^{(t-1)}) \text{ for all } i = 1, \ldots, n,\]

and thus Gauss-Seidel iterations over \( \tau \) and \( (\mu_j, \Sigma_j), j = 1,2 \) require twice the amount of computation required by the Jacobi iterations while each Gauss-Seidel step wastes information that is needed for the other step. Obviously, the two approximations to loglikelihood use are not concave, thus overrelaxation methods cannot be used in this model.

The EM algorithm in detail for two components finite model is as follows:

\[
\begin{align*}
\mu_j^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} y_i / \sum_{i=1}^{n} w_{ij}^{(t)}}{\sum_{i=1}^{n} w_{ij}^{(t)}}, \\
\Sigma_j^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} (y_i - \mu_j^{(t+1)}) (y_i - \mu_j^{(t+1)})^T / \sum_{i=1}^{n} w_{ij}^{(t)}}{\sum_{i=1}^{n} w_{ij}^{(t)}}, \\
p^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} / n},
\end{align*}
\]

where

\[
\begin{align*}
w_{i1} &= \frac{p^{(t-1)} \phi(y_i | \mu_1^{(t-1)}, \Sigma_1^{(t-1)})}{p^{(t-1)} \phi(y_i | \mu_1^{(t-1)}, \Sigma_1^{(t-1)}) + (1 - p^{(t-1)}) \phi(y_i | \mu_2^{(t-1)}, \Sigma_2^{(t-1)})}, \\
w_{i2} &= 1 - w_{i1}, \quad i = 1, \ldots, n, \quad j = 1, 2.
\end{align*}
\]

The proposed GCM algorithm is:

\[
\begin{align*}
\mu_j^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} y_i / \sum_{i=1}^{n} w_{ij}^{(t)}}{\sum_{i=1}^{n} w_{ij}^{(t)}}, \\
\Sigma_j^{(t+1)} &= \frac{\sum_{i=1}^{n} w_{ij}^{(t)} (y_i - \mu_j^{(t+1)}) (y_i - \mu_j^{(t+1)})^T / \sum_{i=1}^{n} w_{ij}^{(t)}}{\sum_{i=1}^{n} w_{ij}^{(t)}}, \\
\tau^{(t+1)} &= \tau^{(t)} - s(\theta^{(t)}) / h(\theta^{(t)}),
\end{align*}
\]
where

\[ s(\theta) = \sum_{i=1}^{n} w_{i1} - np, \]

\[ h(\theta) = -1.5[\sum_{i=1}^{n} w_{i1} - np] + [\sum_{i=1}^{n} w_{i1}^2 - np^2] \]

\[ p = \frac{\exp(\tau)}{(\exp(\tau) + 1)}. \]

In a simulation study to compare the GCM algorithm with the EM algorithm, we generate 100 sets of 100 observations from the finite mixture of \( N(-d, 1) \) and \( N(d, 1) \) with probability \( 1/3 \), and \( 2/3 \), respectively for each value of \( d = 0.25, 0.5, 1, 2 \). For convenience, we set up the models used for both of algorithms to have the same variance.

Figure 4.3 shows comparison of log-scaled number of iterations for each data set between GCM and EM algorithm for selected values of \( d \). Average number of iterations required for convergence of EM and GCM are given in Table 4.5.

**Table 4.5** Comparisons of the number of iterations among GCM and EM

<table>
<thead>
<tr>
<th>True ( \mu_2 - \mu_1 )</th>
<th>EM</th>
<th>GCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>5399.72</td>
<td>2003.98</td>
</tr>
<tr>
<td>1</td>
<td>4714.12</td>
<td>1036.16</td>
</tr>
<tr>
<td>2</td>
<td>589.38</td>
<td>194.95</td>
</tr>
<tr>
<td>4</td>
<td>14.89</td>
<td>18.27</td>
</tr>
</tbody>
</table>

As the figure and the table illustrate, GCM has a speed advantage over the EM algorithm when the mixture consists of two distributions one of which is distant from the other. Since both algorithms are very fast when the two distributions of the mixture are distant from each other, we can conclude that the proposed GCM algorithm is preferable.
Figure 4.3 Comparisons of Log-scaled Number of Iterations between EM and GCM with 100 Random Data Sets
5 LINEAR VARIANCE COMPONENT MODEL

5.1 Conventional Methods for Variance Component Model

Let $y$, $\beta$, $b_i$, $i = 1, \cdots, u$, and $e$ be $n \times 1$, $k \times 1$, $q_i \times 1$, and $n \times 1$ vectors, respectively. The variance components model has the following structure:

$$y = X\beta + \sum_{i=1}^{u} Z_i b_i + e,$$

(5.1)

where each $b_i$, $i = 1, \cdots, u$ is a random vector independent of each other, independent of $e$, and distributed as $N(0, \sigma_i^2 I_{q_i})$, while the distribution of $e$ is $N(0, \sigma_e^2 I_n)$. The vector $\beta$ is fixed and $X$ and $Z_i$ are known $n \times k$ and $n \times q_i$ matrices.

We want to estimate $\theta = (\beta, \sigma_0, \sigma_1, \cdots, \sigma_u)^T$ by finding a point maximizing the loglikelihood

$$l(\theta) = -0.5n \log 2\pi - 0.5 \log |V| - 0.5(y - X\beta)^T V^{-1} (y - X\beta),$$

(5.2)

where $V = \text{var}(y) = \sum_{i=1}^u \sigma_i^2 Z_i Z_i^T + \sigma_e^2 I_n$. Since

$$\frac{\partial V}{\partial \sigma_i^2} = Z_i Z_i^T,$$

$$\frac{\partial \log |V|}{\partial \sigma_i^2} = \text{tr}(V^{-1} \frac{\partial V}{\partial \sigma_i^2}),$$

and

$$\frac{\partial V^{-1}}{\partial \sigma_i^2} = -V^{-1} \frac{\partial V}{\partial \sigma_i^2} V^{-1},$$

(5.3)

$Dl(\theta)$ consists of the elements

$$\frac{\partial l(\theta)}{\partial \beta} = X^T V^{-1} y - X^T V^{-1} X \beta,$$

$$\frac{\partial l(\theta)}{\partial \sigma_i^2} = -0.5 \text{tr}(V^{-1} Z_i Z_i^T) + 0.5(y - X\beta)^T V^{-1} Z_i Z_i^T V^{-1} (y - X\beta),$$

(5.4)
where $Z_0 = I_n$ and $i = 0, 1, \ldots, u$. The elements of $D^2 l(\theta)$ and $I_{obs}(\theta) = -E[D^2 l(\theta)]$ are given by:

\[
\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta} = -X^T V^{-1} X,
\]

\[
\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \sigma_i} = -X^T V^{-1} Z_i Z_i^T V^{-1} (y - X \beta),
\]

\[
\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2} = 0.5 \text{tr}(V^{-1} Z_i Z_i^T V^{-1} Z_i Z_i^T)
\]

\[
(y - X \beta)^T V^{-1} Z_i Z_i^T V^{-1} Z_i Z_i^T V^{-1} (y - X \beta),
\]

\[
- E\left[\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta}\right] = -\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta}, - E\left[\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \sigma_i}\right] = 0,
\]

and

\[
- E\left[\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2}\right] = 0.5 \text{tr}(V^{-1} Z_i Z_i^T V^{-1} Z_i Z_i^T).
\]

As reviewed in Chapter 1, Newton’s algorithm is defined by the iteration function

$M_q(\theta) = \theta - [D^2 l(\theta)]^{-1} Dl(\theta)$ where we substitute expressions above for $D^2 l(\cdot)$ and $Dl(\cdot)$ to obtain Newton’s iteration for the above model. One effective modification of Newton’s method is Fisher Scoring method whose iteration function is $M_f(\theta) = \theta - [I_{obs}(\theta)]^{-1} Dl(\theta)$.

To formulate the steps for the EM algorithm for the above optimization problem, set

$(y, b_1, \ldots, b_u)^T$ as the complete data vector and $b_1, \ldots, b_u$ as the missing data vector. Then the MLEs of variance components can be estimated as follows:

Since $\text{cov}(y, b_i) = E[\sum_{i=1}^{u} (Z_i^T b_i + e) b_i] = \sigma_i^2 Z_i$, the density of the complete data vector is

\[
(2\pi)^{-\frac{1}{2}(n+\sum_{i=1}^{u} n_i)} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} Q\right), \quad (5.5)
\]

where

\[
Q = [(y - X \beta)^T, b_1^T, \ldots, b_u^T] \Sigma^{-1} [(y - X \beta)^T, b_1^T, \ldots, b_u^T]^T \quad (5.6)
\]

and

\[
\Sigma = \begin{bmatrix}
V & \{\sigma_i^2 Z_i\}_{i=1}^{u} \\
\{\sigma_i^2 Z_i^T\}_{i=1}^{u} & \{\sigma_i^2 I_{qi}\}_{i=1}^{u}
\end{bmatrix}. \quad (5.7)
\]
\[ \{ \cdot \}_i \] denotes block diagonal matrix with \( \cdot_i, i = 1, \cdots, u \) matrices as elements. To simplify (5.5) in terms of the variance components we first use the standard result

\[
\begin{vmatrix} A & B \\ C & D \end{vmatrix} = |D||A - BD^{-1}C| \tag{5.8}
\]

to derive

\[
|\Sigma| = \prod_{i=1}^{u} |\sigma_i^2I_{q_i}| |V - \sum_{i=1}^{u} \sigma_i^2Z_iZ_i^T| = \prod_{i=1}^{u} (\sigma_i^2)^{q_i} |\sigma_0I_n| = \prod_{i=0}^{u} (\sigma_i^2)^{q_i}, \tag{5.9}
\]

where \( q_0 = n \). And since

\[
(X'X)^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & (X_2'X_2)^{-1} \end{bmatrix} + \begin{bmatrix} I \\ -(X_2'X_2)^{-1}X_2X_1 \end{bmatrix} \times^{-1} (X_1'X_2) \times^{-1} [I - X_1'X_2(X_2'X_2)]
\tag{5.10}
\]

where \( M_2 = I - X_2(X_2'X_2)^{-1}X_2' \), and

\[
(X'X) = \begin{bmatrix} (X_1'X_1) & (X_1'X_2) \\ (X_2'X_1) & (X_2'X_2) \end{bmatrix}, \tag{5.11}
\]

\[
\Sigma^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & \{\sigma_i^{-2}I_{q_i}\}_{i=1}^{u} \end{bmatrix} + \begin{bmatrix} I \\ \{-Z_i^T\}_{i=1}^{u} \end{bmatrix} \sigma_0^{-2}I_n \times \{I - Z_i\}_{i=1}^{u}. \tag{5.12}
\]

Thus, we have

\[
l(x|\theta) = -\frac{1}{2} \sum_{i=0}^{u} q_i \log 2\pi - \frac{1}{2} \sum_{i=0}^{u} q_i \log \sigma_i^2 - \frac{1}{2} \sum_{i=0}^{u} \frac{b_i^Tb_i}{\sigma_i^2}, \tag{5.13}
\]
where \( x = (y', b_1^T, \ldots, b_u^T)' \) and \( b_0 = e \). From this, it is easy to derive the MLE based on the complete data as

\[
\hat{\sigma}_i^2 = \frac{b_i^T b_i}{q_i}, \quad i = 1, \ldots, u
\]

\[
\hat{\sigma}_0^2 = \frac{e^T e}{n},
\]

and

\[
\hat{\beta} = (X'X)^{-1}X'(y - Zb). \tag{5.15}
\]

To finalize the iteration steps for the EM algorithm we need the conditional expected values of \( b'b \) and \( Zb \) given \( y \). The derivation is straightforward using standard multivariate normal results:

\[
x_1|x_2 \sim N[\mu_1 + V_{12}V_{22}^{-1}(x_2 - \mu_2), V_{11} - V_{12}V_{22}^{-1}V_{21}] \tag{5.16}
\]

where

\[
x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim N\left( \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \right). \tag{5.17}
\]

Then the conditional distribution of \( b \) given \( y \) is

\[
b_i|y \sim N[\sigma_i^2 Z_i'V_i^{-1}(y - X_i\beta), \sigma_i^2 I_q - \sigma_i^4 Z_i'V_i^{-1}Z_i], \tag{5.18}
\]

so that

\[
E(b_i|y) = \sigma_i^2 Z_i'V_i^{-1}(y - X_i\beta) \tag{5.19}
\]

and

\[
E(b_i^T b_i|y) = E(b_i|y)^T E(b_i|y) + tr(\sigma_i^2 I_q - \sigma_i^4 Z_i'V_i^{-1}Z_i) \tag{5.20}
\]

using the fact that

\[
E(y'Ay) = tr(AV) + \mu'A\mu \tag{5.21}
\]

for a random vector \( y \) with \( E(y) = \mu \) and \( var(y) = V \). Similarly we have

\[
E(e|y) = \sigma_0^2 V^{-1}(y - X\beta) \tag{5.22}
\]
and

\[ E(e'e|y) = E(e|y)'E(e|y) + tr(\sigma_0^2I_n - \sigma_0^4V^{-1}). \]  \hspace{1cm} (5.23) 

Hence the EM algorithm for MLE of variance components is the following:

1. Set starting values \( \sigma_i^{2(0)}, i = 0, \ldots, u, \) and \( \beta^{(0)} \)

2. \textbf{(E step)} Calculate conditional expected values of the sufficient statistics.

\[
E(b_i^Tb_i|y)_{\theta^{(t)}} = \sigma_i^{4(t)}(y - X\beta^{(t)})'(V^{(t)})^{-1}Z_iZ_i^T(V^{(t)})^{-1}(y - X\beta^{(t)}) \\
+ \text{tr}(\sigma_i^{2(t)}I - \sigma_i^{4(t)}Z_i^T(V^{(t)})^{-1}Z_i), \quad i = 0, \ldots, u \]  \hspace{1cm} (5.24) 

and

\[
E(y - \sum_{i=1}^{u} Z_ib_i|y)_{\theta^{(t)}} = y - \sum_{i=1}^{u} \sigma_i^{2(t)}Z_iZ_i^T(V^{(t)})^{-1}(y - X\beta^{(t)}) \\
= X\beta^{(t)} + \sigma_0^{2(t)}(V^{(t)})^{-1}(y - X\beta^{(t)}), \hspace{1cm} (5.25) 
\]

where \( \theta = (\beta', \sigma_1^2, \sigma_2^2)' \).

3. \textbf{(M step)} Maximize the likelihood of the complete data, based on (5.14) and (5.15)

\[
\sigma_i^{2(t+1)} = \frac{1}{q_i}E(b_i^Tb_i|y)_{\theta^{(t)}}, \quad i = 0, \ldots, u \]  \hspace{1cm} (5.26) 

and

\[
\beta^{(t+1)} = (X'X)^{-1}X'E(y - \sum_{i=1}^{u} Z_ib_i|y)_{\theta^{(t)}}. \]  \hspace{1cm} (5.27) 

4. Go to 2 until convergence is reached.

Laird(1982) suggested using GLS estimator, \( \beta \), instead of (5.27), that is,

\[
\beta^{(t+1)} = (X'(V^{(t)})^{-1}X)^{-1}X'(V^{(t)})^{-1}y, \hspace{1cm} (5.28) 
\]

The above algorithm was presented as an ECME algorithm later (McLachlan and Krishnan, 1997). In this case, we avoid calculating \( \beta^{(t+1)} \) at each iteration. This modified version is as follows:
1. Set starting values $\sigma_i^{(0)}, i = 0, \cdots, u$

2. (E step) Calculate conditional expected values of the sufficient statistics.

$$
E(b_i^Tb_i|y)_{\theta=\theta^{(t)}} = \sigma_i^{(t)}(y - X\beta^{(t)})'(V^{(t)})^{-1}Z_i V^{(t)}V^{(t)}y
\quad i = 0, \cdots, u
$$

3. (M step) Maximize the likelihood of the complete data, based on (5.14) and (5.15)

$$
\sigma_i^{2(t+1)} = \frac{1}{q_i}E(b_i^Tb_i|y)_{\theta=\theta^{(t)}}, \quad i = 0, \cdots, u
$$

4. If convergence is reached, set $\sigma_i^2 = \sigma_i^{2(t)}$, $i = 0, \cdots, u$, and

$$
\hat{\beta} = (X'(V^{(t)})^{-1}X)^-X'(V^{(t)})^{-1}y;
$$

otherwise go to 2.

In summary, iteration function of Laird’s ECME algorithm is $M_E(\theta) = \theta - [H(\theta)]^{-1}Dl(\theta)$ where $H(\theta)$ is the block diagonal matrix of $(X^TV^{-1}X, q_0/\sigma_0^2, \cdots, q_u/\sigma_u^2)$.

Restricted Maximum Likelihood(REML) estimation is a modified ML estimation method in which the likelihood of $A^Ty$ is maximized instead of that of $y$, where $A^T$ is a $n-k$ full row rank matrix such that $A^TX = 0$. Thus for such a $A^T$ matrix, restricted loglikelihood is

$$
l(\theta) = -0.5(n-k)\log 2\pi - 0.5 \log |A^TV\lambda| - 0.5y^TA(A^TV^{-1}A)^{-1}A^Ty,
$$

The formulae necessary for quadratic approximation methods like Newton’s method and Fisher Scoring are as follows:

$$
\frac{\partial l(\theta)}{\partial \sigma_i^2} = -0.5tr(PZ_i^TZ_i) + 0.5y^TPZ_iZ_i^TPy,
$$

$$
\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2} = 0.5tr(PZ_iZ_i^TPZ_jZ_j^T) - y^TPZ_iZ_i^TPZ_jZ_j^TPy,
$$

$$
- E[\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2}] = 0.5tr(PZ_iZ_i^TPZ_jZ_j^T),
$$
where \( P = V^{-1} - V^{-1}X(X^TV^{-1}X)^{-1}X^TV^{-1} \) and \( i, j = 0, \ldots, u \).

For the EM algorithm, we also have the following nice result:

\[
E(b_i|b_i^t|\theta)_{\theta=\theta^{(t)}} = \sigma_i^{4(t)}(y^tP^{(t)}Z_{i'}P^{(t)}y) + tr(\sigma_i^{2(t)}I_n - \sigma_i^{4(t)}Z_{i'}(P^{(t)})Z_i), \tag{5.33}
\]

which leads to the following EM algorithm for REML:

1. Set starting values \( \sigma_i^{2(0)}, i = 0, 1, \ldots, r \)

2. (E step) Calculate conditional expected values of the sufficient statistics for \( i = 0, 1, \ldots, r \)

\[
E(b_i|b_i^t|A'y)|_{\theta=\theta^{(t)}} = \sigma_i^{4(t)}(y^tP^{(t)}Z_{i'}P^{(t)}y) + tr(\sigma_i^{2(t)}I_n - \sigma_i^{4(t)}Z_{i'}P^{(t)}Z_i) \tag{5.34}
\]

where \( P = V^{-1} - V^{-1}X(X^TV^{-1}X)^{-1}X^TV^{-1} \) and \( A' \) is full row rank \( N - p \) such that \( A'X = 0 \).

3. (M step) Maximize the likelihood of the complete data.

\[
\sigma_i^{2(t+1)} = \frac{1}{q_i}E(b_i|b_i^t|A'y)|_{\theta=\theta^{(t)}}, \tag{5.35}
\]

4. If convergence is reached, set \( \hat{\sigma}_i^2 = \sigma_i^{2(t)} \). Otherwise go to 2.

For computing ML or REML estimates in variance components model, Newton’s algorithm does not work very well even with initial values \( \sigma_i^{2(0)}, i = 1, \ldots, u \) close to a local maximum. Further, when a local maximum is near zero, these converge to negative variance component estimates. This is a result of the likelihood surface being neither quadratic nor concave near the solution. Another quadratic approximation algorithm, Fisher Scoring method does not require linear search in each iteration, and with suitable initial values for monotone convergence, converges to a local maximum. But it also may produce MLE with negative variance component estimates.
Callanan and Harville (1991) introduced a modified versions of Newton's algorithms with different parameterizations with associated linearization factors, that produced very fast convergence. In special cases, their algorithm converged in one iteration. But their parameterizations lead to increased computational burden in each iteration of the algorithms. And their methods may also converge to negative variance component estimates.

EM algorithm for variance components obviously leads to monotone convergence and is fast when there is a large number of observations and when the MLE of any variance component is not close to zero. And even if MLEs of variance components include zero, the sequences for the variance components from EM do not converge to negative values. Also, while the quadratic approximation algorithm requires the computations of at least a $u \times u$ Hessian matrix in each iteration, EM needs only the computation of the $u$ components of $Dl(\theta)$ in each iteration. A disadvantage of EM algorithm, however, is its slowness when the MLE of any variance component is zero. In that case, it is too slow to be of any practical use.

In order to improve EM algorithm, Liu and Rubin (1994) suggest ECME algorithm which was mentioned in Chapter 3. Since the MLE of residual variance component $\sigma_0^2$ is not zero in general, we can maximize $l(\theta)$ in terms of $\sigma_0^2$ by a quadratic approximation method in each iteration while the EM steps are performed for the other variance components $\sigma_i^2, i = 1, \cdots, u$. Although the ECME improves the speed of the EM algorithm, if MLE of a variance component is zero, then the algorithm degenerates to be as slow as EM. Furthermore, when the MLE of residual variance component $\sigma_0^2$ is zero, the algorithm may lead to a negative value for the MLE of the residual variance component.

Among other possibilities, nonlinear Jacobi and nonlinear Gauss-Seidel algorithms using quadratic approximation methods can be useful when the initial value is far from a local maximum, Fisher Scoring as well as Newton's method fail monotone convergence, but NJA or NGS might converge monotonely. However, they may still be in danger of converging to a negative variance component estimates.
5.2 New GCM Algorithms for Variance Component Estimation

True missing data denotes the missing data that occurs in \( y \) and \( X \). In this section, we discuss only the case that there is no true missing data. Without loss of generality, we shall not consider discussion about REML estimation in this section. ECME algorithm for the variance components model is, for all practical purposes, defined to consist of two cycles as follows:

**Cycle 1 for \( \sigma_i^2, i = 1, \cdots, u \):** EM approximation with no R-step.

\[
q_i \sigma_i^{2(1)} = q_i \sigma_i^{2(0)} + \sigma_i^{4(0)} y^T P^{(0)} Z_i Z_i^T P^{(0)} y - tr[\sigma_i^{4(0)} Z_i^T (V^{(0)})^{-1} Z_i],
\]

where

\[
V^{(0)} = \sum_{i=1}^{u} Z_i Z_i^T \sigma_i^{2(0)} + I_{n \sigma_2^{2(0)}},
\]

and

\[
P^{(0)} = (V^{(0)})^{-1} - (V^{(0)})^{-1} X [X^T (V^{(0)})^{-1} X]^{-1} X^T (V^{(0)})^{-1}.
\]

**Cycle 2 for \( \sigma_0^2 \):** Quadratic approximation (Fisher scoring) with no R-step (\( r = 1 \)).

\[
\sigma_0^{2(2)} = \sigma_0^{2(0)} + \frac{0.5}{tr[(V^{(1)})^{-1} (V^{(1)})^{-1}]} \{ y^T P^{(1)} P^{(1)} y - tr[(V^{(1)})^{-1}]\}.
\]

where

\[
V^{(1)} = \sum_{i=1}^{u} Z_i Z_i^T \sigma_i^{2(1)} + I_{n \sigma_0^{2(0)}},
\]

and

\[
P^{(1)} = (V^{(1)})^{-1} - (V^{(1)})^{-1} X [X^T (V^{(1)})^{-1} X]^{-1} X^T (V^{(1)})^{-1}.
\]

The reason for the absence of an R-step in Cycle 2 is due to the fact that more than one inner iteration for Cycle 2 is not helpful in improving towards a local maximum even though maximization by several inner iterations might seem appropriate for the definition of ECME. The first proposed GCM algorithm (labeled GCM-1) GCM-1 also has two cycles with Cycle 1 having an R-step (\( r = 2 \) and an additional overrelaxation step with \( \theta = (\sigma_0^2, \sigma_1^2, \cdots, \sigma_u^2) \).
The second proposed GCM algorithm (labeled GCM-2) is the same as GCM-1 except that there is a partial Aitken acceleration step between Cycle 1 and Cycle 2 of GCM 1 as following:

partial Aitken step

$$\sigma^{(2)} = \omega(t) \sigma^{(2*)} + (1 - \omega(t))\sigma^{(1*)},$$

(5.38)

where $\sigma^{(1*)}$ is the update of $\sigma = (\sigma_1^2, \cdots, \sigma_u^2)^T$ after the first run of Cycle 1, $\sigma^{(2*)}$ is the update of $\sigma$ after the second run of Cycle 1, and $\omega(t)$ depends on $\phi(t) = ||\sigma^{(2*)} - \sigma^{(1*)}||/||\sigma^{(1*)} - \sigma^{(0)}||$ as follows:

$$\omega(t) = \begin{cases} 
1 & \text{for } t \text{ such that } \phi(t) - \phi(t-1) > 0.001, \\
\frac{1}{1 - \phi(t)} & \text{otherwise,}
\end{cases}$$

(5.39)

Note that this partial Aitken step fit to our definition of cycle in GCM algorithm since the step uses quadratic approximation step with Hessian $\frac{1}{\sigma}$ and the corresponding maximization step. Thus, GCM-2 is still justified as a GCM algorithm. Surprisingly, the algorithm works well without failure of monotone convergence as shown in numerical example of Chapter 4.

So far, we have not introduced any new algorithm. Beginning with the concepts of the ECME algorithm, two new GCM algorithms, GCM-I and GCM-2, were developed. We shall utilize linearization methods discussed in Chapter 2 to derive another new algorithm. Figure 5.1, and 5.2 are the results of simulation experiments performed as part of the linearization procedures discussed in Chapter 2.

In considering estimation of variance components as univariate problems for the purpose of linearization as discussed in Chapter 2, Figure 5.1 illustrates the flatness of the first derivative that leads to the non-effectiveness of quadratic approximation algorithms. In fact, with the score function $Dl(\theta)$, Newton's algorithm converges only when the initial values $\sigma_i^2(0), i = 0, \cdots, u$ are very close to the local maximum.
Figure 5.1 Original Loglikelihood, Its First Derivative, and its Second Derivative Functions in Terms of $\sigma_i^2$
Figure 5.2 Original Loglikelihood, Its Linearized Score Function, and Its
Hessian Functions with Linearization Factor $\sigma_1^4/q_1$ in Terms of
$\sigma_1^2$
The last column of Figure 5.2 depicts the linearized Hessian $D^2l_L(\theta)$. An algorithm that uses this Hessian and the score function will obviously produce a fast algorithm. When we use the linearization factor $\sigma_i^4$ (i.e., linearization index, $p = 2$) for each corresponding element of $Dl(\theta)$, the linearized score function $Dl_L(\theta) = \sigma_i^4 Dl(\theta)$ is nearly linear on the subspace $\Omega_- = \{\theta; Dl(\theta) < 0\}$ as seen in Figure 5.2. Note that the EM algorithm discussed earlier uses the above score function but not the above Hessian. To show this, by combining (5.24) and (5.26), and observing that

$$\frac{\partial l(\theta)}{\partial \sigma_i^2} = \frac{0.5}{\sigma_i^4} E(b_i^Tb_i | y, \theta) - \frac{0.5q_i}{\sigma_i^2}$$

$$E(b_i^Tb_i | y, \theta) = \sigma_i^4 (y - X\beta)^T V^{-1} Z_i V^{-1}(y - X\beta) + tr(\sigma_i^2 I - \sigma_i^4 Z_i^TV^{-1}Z_i),$$

we obtain

$$\sigma_i^{2(t+1)} = \sigma_i^{2(t)} + [\sigma_i^{4(t)} \frac{\partial l(\theta)}{\partial \sigma_i^2} | \sigma_i^2 = \sigma_i^{2(t)}]/q_i \quad (5.40)$$

In the above iteration function, $\sigma_i^{4(t)} \frac{\partial l(\theta)}{\partial \sigma_i^2}$ corresponds to the score function and $q_i$ to the Hessian, of a quadratic approximation algorithm. Comparing this to the algorithm based on the above linearization, we see that the score functions are identical. However, the Hessian function (shown on the last column of Figure 5.2 is not constant in the neighborhood of the solution as expected in the above EM algorithm (i.e., $q_i$ is a constant). And this nonlinearity of $Dl_L(\theta)$ in the neighborhood is more serious as the solution is closer to zero. This explains the reason for the EM algorithm to become very slow when a solution is close to zero.

On the basis of this linearization study, we propose a new GCM algorithm which is equivalent to nonlinear Jacobi using a linearized quadratic approximation method (NJALQ) as follows:

$$\theta^{(t+1)} = \theta^{(t)} - [W(\theta^{(t)})]^{-1} Dl_L(\theta^{(t)}), \quad (5.41)$$

where $Dl_L(\theta) = Dl(\theta) K(\theta)$, $W(\theta)$ is block diagonal elements of $D^2l_L(\theta)$ corresponding to $(\beta, \sigma_0^2, \ldots, \sigma_n^2)$, and $K(\theta) = blockdiag([X^TV^{-1}X]^{-1}, \sigma_0^4, \ldots, \sigma_n^4)$, respectively. More
specifically, the $t$th iteration of NJALQ algorithm is

$$
\beta^{(t+1)} = (X^T V^{-1(t)} X)^{-1} X^T V^{-1(t)} y,
$$

(5.42)

$$
\sigma^2_{j}^{(t+1)} = \frac{\sigma^2_{j}^{(t)} \frac{\partial l(\theta)}{\partial \sigma^2_{j}} |_{\sigma^2_{j}=\sigma^2_{j}^{(t)}} + \sigma^2_{j}^{(t)} \frac{\partial^2 l(\theta)}{\partial \sigma^2_{j}^2} |_{\sigma^2_{j}=\sigma^2_{j}^{(t)}}}{2\sigma^2_{j}^{(t)}}
$$

for $j = 0, \cdots, u.$

Of course, the first step of (5.42) can be omitted and computed once at convergence by setting $V^{-1}(y - X\beta) = V^{-1}y - V^{-1}X[X^T V^{-1}X]^{-1}X^T V^{-1}y$. The only difference between this algorithm and EM is that $q_i$ in EM (see (5.40) above) is replaced by

$$
\frac{2\sigma^2_{j}^{(t)} \frac{\partial l(\theta)}{\partial \sigma^2_{j}} |_{\sigma^2_{j}=\sigma^2_{j}^{(t)}} + \sigma^2_{j}^{(t)} \frac{\partial^2 l(\theta)}{\partial \sigma^2_{j}^2} |_{\sigma^2_{j}=\sigma^2_{j}^{(t)}}}{2\sigma^2_{j}^{(t)}}
$$

(5.43)

5.2.1 Computation of First and Second Derivatives of Loglikelihood

Whether one uses a quadratic approximation method, EM algorithm, or a new GCM algorithm, it is required to obtain the first and/or second derivatives of loglikelihood. Since the procedure includes inversion of $n \times n$ matrix $V$, usual matrix operations involving computation of the inversion are quite inefficient. Hemmerle and Hartley(1973), Hemmerle and Lorens(1976), and Goodnight and Hemmerle(1979), however, have established elegant computational methods involving the $W$ Transformation, to obtain necessary quantities for computing first and second order of the derivatives. We shall briefly introduce the algorithm in this section.

Defining $Z_{tot} = [Z_1, \cdots, Z_u]$, it is needed to compute the matrix

$$
W = \begin{bmatrix}
Z_{tot}^T V^{-1} Z_{tot} & Z_{tot}^T V^{-1} X & Z_{tot}^T V^{-1} y \\
X^T V^{-1} Z_{tot} & X^T V^{-1} X & X^T V^{-1} y \\
y^T V^{-1} Z_{tot} & y^T V^{-1} X & y^T V^{-1} y
\end{bmatrix}
$$

(5.44)

Hemmerle and Hartley(1973) found that

$$
H^{-1} = \sigma_0^2 V^{-1} = I - Z_{tot} (D^{-1} + Z_{tot}^T Z_{tot})^{-1} Z_{tot}^T,
$$
where $D = \text{diag}(\sigma^2_{1_{q_1} x_1}, \ldots, \sigma^2_{1_{q_1} x_1})$ is $\sum_{i=1}^{u} q_i \times \sum_{i=1}^{u} q_i$ diagonal matrix, and that $W$ can be established by setting

$$
\sigma^2_0 W = W_0 - \begin{bmatrix}
Z^T_{tot} Z_{tot} \\
X^T Z_{tot} \\
y^T Z_{tot}
\end{bmatrix} (D^{-1} + Z^T_{tot} Z_{tot})^{-1} [Z^T_{tot} Z_{tot}, X^T Z_{tot}, y^T Z_{tot}],
$$

where

$$
W_0 = \begin{bmatrix}
Z^T_{tot} Z_{tot} & Z^T_{tot} X & Z^T_{tot} y \\
X^T Z_{tot} & X^T X & X^T y \\
y^T Z_{tot} & y^T X & y^T y
\end{bmatrix}.
$$

Goodnight and Hemmerle(1979) show that $W$ can be obtained by performing sweep operations(Appendix A) on the matrix

$$
M = \begin{bmatrix}
Z^T_{tot} Z_{tot} + D^{-1} & L \\
L^T & W_0
\end{bmatrix},
$$

where $L = [Z^T_{tot} Z_{tot}, X^T Z_{tot}, y^T Z_{tot}]$, and developed a modified sweep algorithm that achieves dimensional reduction so that the computation of $W$ as well as $|H| = |D^{-1} + Z^T_{tot} Z_{tot}||D|$ is done on a smaller matrix $W_0$ than the matrix $M$. After performing the $W$ transformation, the remaining computations involve only trivial matrix operations. Furthermore, when EM or GCM algorithms are used, inversion of any other matrix is not necessary for computing the derivatives.

### 5.3 Empirical Comparison of ML algorithms

In this section, several specific algorithms which are well-known, and GCM-3 algorithm are implemented and compared in three examples. For improving efficiency of the algorithms, the computations of $\sigma^2_0$ in each iteration are set identically to $\sigma^2_0(g(t+1)) = \frac{1}{n}(y - X\beta^{(t+1)})^T H^{-1}(t)(y - X\beta^{(t+1)})$. Assuming that $X$ has full column rank, the con-
Conventional algorithms for the variance components model are presented as follows:

**Newton-Raphson**

For \( t = 0, 1, \ldots \),

1. Set \( \beta_{(t+1)} = (X^T H^{-1(t)} X)^{-1} X^T H^{-1(t)} y \) and \( \sigma_0^{(t+1)} = \frac{1}{n}(y - X\beta_{(t+1)})^T H^{-1(t)}(y - X\beta_{(t+1)}) \)

2. Set \( \gamma_{(t+1)} = \gamma^{(t)} - \alpha[D^2 l(\gamma^{(t)})]^{-1} Dl(\gamma^{(t)}) \) and find \( \alpha \) by performing linear search so that \( l(\theta^{(t+1)}) \geq l(\theta^{(t)}) \).

3. If \( ||\theta^{(t+1)} - \theta^{(t)}|| > \epsilon \) for a given \( \epsilon \) then go to 1; otherwise accept current iterate \( \theta^{(t+1)} \) as the limit of the sequence \{\( \theta^{(t)} \)\},

where \( H = \frac{1}{\sigma_0^2} V, \gamma = (\sigma_1^2/\sigma_0^2, \ldots, \sigma_n^2/\sigma_0^2)^T, Dl(\gamma) \) and \( D^2 l(\gamma) \) are the first and the second derivatives, respectively in terms of \( \gamma \).

**Fisher Scoring**

For \( t = 0, 1, \ldots \),

1. Same as Step 1 above.

2. Set \( \sigma^{(t+1)} = \sigma^{(t)} - \alpha[E(D^2 l(\sigma^{(t)}))]^{-1} Dl(\sigma^{(t)}) \) and find \( \alpha \) by performing linear search so that \( l(\theta^{(t+1)}) \geq l(\theta^{(t)}) \).

3. If \( ||\theta^{(t+1)} - \theta^{(t)}|| > \epsilon \) for a given \( \epsilon \) then go to 1; otherwise accept current iterate \( \theta^{(t+1)} \) as the limit of the sequence \{\( \theta^{(t)} \)\},

where \( \sigma = (\sigma_1^2, \ldots, \sigma_n^2)^T, Dl(\sigma) \) and \( D^2 l(\sigma) \) are the first and the second derivatives, respectively in terms of \( \sigma \).
EM

For $t = 0, 1, \cdots$,

1. Same as Step 1 above.

2. Compute $\sigma^{(t+1)} = \sigma^{(t)} - [L_{EM}(\sigma^{(t)})]Dl(\sigma^{(t)})$.

3. If $||\theta^{(t+1)} - \theta^{(t)}|| > \epsilon$ for a given $\epsilon$ then go to 1; otherwise accept current iterate $\theta^{(t+1)}$ as the limit of the sequence $\{\theta^{(t)}\}$,

where $L_{EM}(\sigma)$ is a diagonal matrix whose elements are $(\sigma_1^4/q_1, \cdots, \sigma_u^4/q_u)$.

Newton-Raphson and Fisher Scoring need an additional step for avoiding negative values being generated for the iterate $\sigma^{(t+1)}$. Harville(1977) provides various methods to overcome this problem.

The following GCM-L algorithm is modified from GCM-3 in Subsection 4.6.2 for computational efficiency of $\sigma_0^2$.

GCM-L

For $t = 0, 1, \cdots$,

1. Same as Step 1 above.

2. Compute $\sigma^{(t+1)} = \sigma^{(t)} - [L_{GCM}(\sigma^{(t)})]Dl(\sigma^{(t)})$.

3. If $||\theta^{(t+1)} - \theta^{(t)}|| > \epsilon$ for a given $\epsilon$ then go to 1; otherwise accept current iterate $\theta^{(t+1)}$ as the limit of the sequence $\{\theta^{(t)}\}$,

where $L_{GCM}(\theta)$ is diagonal matrix whose elements are $(\sigma_1^4/d_1(\theta), \cdots, \sigma_u^4/d_u(\theta))$, $d_i(\theta) = 2\sigma_i^2\delta_i + \sigma_i^4\eta_i$, $\delta_i$ is the $i$th element of $Dl(\sigma)$, and $\eta_i$ is the $i$th diagonal element of $D^2l(\sigma)$. 
An EM algorithm incorporating overrelaxation is follows:

**EM-O**

Set \( \omega = 1 \). For \( t = 0, 1, \cdots \),

1. Same as Step 1 above.

2. Compute \( \mathbf{\sigma}^{(t+1)} = \mathbf{\sigma}^{(t)} - \omega [L_{EM}(\mathbf{\sigma}^{(t)})]Dl(\mathbf{\sigma}^{(t)}) \).

3. (*Overrelaxation Step*) Set

\[
\lambda^{(t)} = \| \theta^{(t+1)} - \theta^{(t+1)} \| / \| \theta^{(t+1)} - \theta^{(t+1)} \|
\]

If \( \omega = 1 \) and \( \lambda^{(t)} - \lambda^{(t-1)} < \epsilon_0 \) for a given \( \epsilon_0 \) and some \( t_0 \), then set \( \omega = (1 + \lambda^{(t_0)}) \).

4. If \( \| \theta^{(t+1)} - \theta^{(t)} \| > \epsilon \) for a given \( \epsilon \) then go to 1; otherwise accept current iterate \( \theta^{(t+1)} \) as the limit of the sequence \( \{ \theta^{(t)} \} \),

where \( L_{EM}(\mathbf{\sigma}) \) is a diagonal matrix whose elements are \( (\sigma_1^2 / q_1, \cdots, \sigma_u^2 / q_u) \).

Now, we compare the performance of four ML algorithms: Fisher Scoring (FS), EM, GCM-L, EM with overrelaxation (EM-O) using three examples of variance components models (or mixed models) with real data. The reason for the choice of Fisher Scoring rather than the original Newton-Raphson is because of the fact that FS is known to be the quadratic approximation method admitting the widest possible range of suitable initial values for this model.

The first example is an evaluation of the breeding value of a set of five sires in raising pigs. Each sire is mated to a random group of dams, each mating producing a litter of pigs whose characteristics are the criterion. The model for \( y_{ijk} \), average daily gain of pigs, is

\[
y_{ijk} = \mu + a_i + B_{ij} + e_{ijk} \quad (5.46)
\]
The $\alpha_i$ are constants associated with the sire effects and the $B_{ij}$ and the $\epsilon_{ijk}$ are random variables corresponding to dam and offspring effects. (Snedecor and Cochran, 1967). We tried the four algorithms first with rough initial values $\sigma_{1}^{(0)} = 1$, $\sigma_{0}^{(0)} = 1$, and $\sigma_{1}^{(0)} = 10$, $\sigma_{0}^{(0)} = 10$, and then with initial values set to values in output from SAS ANOVA ($\sigma_{0}^{(0)} = 0.0375, \sigma_{1}^{(0)} = 0.038$).

The convergence behavior of the algorithms with this data set is displayed in Table 5.1. With rough initial values, Fisher Scoring algorithm diverges while EM, EM-0, GCM-L converge. Using the total CPU time in seconds, GCM-L shows the best efficiency with these initial values. With the initial values set to those from SAS/ANOVA, however, speed and convergence of FS improve.

<table>
<thead>
<tr>
<th>initial values</th>
<th>EM</th>
<th>GCM-L</th>
<th>FS</th>
<th>EM-O</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{0}^{(0)} = 1, \sigma_{1}^{(0)} = 1$</td>
<td>3.4 (116)</td>
<td>1.033 (26)</td>
<td>0 (0)</td>
<td>2.517 (80)</td>
</tr>
<tr>
<td>$\sigma_{0}^{(0)} = 10, \sigma_{1}^{(0)} = 10$</td>
<td>3.433 (116)</td>
<td>1.1 (27)</td>
<td>0 (0)</td>
<td>2.517 (80)</td>
</tr>
<tr>
<td>$\sigma_{0}^{(0)} = 0.0375, \sigma_{1}^{(0)} = 0.038$</td>
<td>3.317 (113)</td>
<td>0.983 (24)</td>
<td>0.6 (10)</td>
<td>2.417 (77)</td>
</tr>
</tbody>
</table>

Somewhat different from the first example, the second is a model with unbalanced data from Harville and Fenech(1985). Let $y_{ijkl}$ represent the weight (at birth) of the $l$th of those lambs that are the offspring of the $k$th sire in the $j$th population line and of a dam belonging to the $i$th age category. This model is the following:

$$y_{ijkl} = \mu + \delta_i + \pi_j + s_{jk} + e_{ijkl} \quad (5.47)$$

where the age effects ($\delta_1, \delta_2, \delta_3$) and line effects ($\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$) are fixed effects, where the sire with line effects ($s_{11}, s_{12}, \ldots, s_{58}$) are random effects that are distributed independently as $N(0, \sigma_s^2)$, and where the random errors ($e_{1111}, e_{1121}, \ldots, e_{3582}$) are distributed as $N(0, \sigma^2)$ independently of each other and of the sire effects. Table 5.2 show the results.
In this example, results are similar for the three choices of the initial values. Note that indeed, the maximum of \((\sigma_0^2, \sigma_1^2)\) is \((2.9440619, 0)\) which is on the boundary of \(\Omega = \mathbb{R}^+ \times \mathbb{R}^+\). The iterates from EM, GCM-L, and EM-O did not converge to negative values even though the number of iterations for EM and EM-O were large in these cases as shown in Table 5.2.

Table 5.2  Total CPU times (number of iterations) of the four algorithms converging to \(\hat{\sigma}_0 = 2.9440619, \hat{\sigma}_1 = 0\) in the second example (total CPU time (number of iteration) = 0 (0) implies divergence of the algorithm)

<table>
<thead>
<tr>
<th>initial values</th>
<th>EM</th>
<th>GCM-L</th>
<th>FS</th>
<th>EM-O</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_0^{(0)} = 1, \sigma_1^{(0)} = 1)</td>
<td>&gt; 10000</td>
<td>1.35 (31)</td>
<td>0 (1)</td>
<td>&gt; 10000</td>
</tr>
<tr>
<td>(\sigma_0^{(0)} = 10, \sigma_1^{(0)} = 10)</td>
<td>&gt; 10000</td>
<td>1.40 (32)</td>
<td>0 (1)</td>
<td>&gt; 10000</td>
</tr>
<tr>
<td>(\sigma_0^{(0)} = 2.763, \sigma_1^{(0)} = 0.7672)</td>
<td>&gt; 10000</td>
<td>1.23333 (28)</td>
<td>0 (1)</td>
<td>&gt; 10000</td>
</tr>
</tbody>
</table>

In the two variance component case as in our examples, if \(\hat{\sigma}_1 = 0\), then \(\hat{\sigma}_0 = (y^T y - \beta^T X^T y)/n\), i.e., there is a closed form solution for the MLE of \(\sigma_0\). This is the reason for FS to converge to the maximum in one iteration in this example since the first iterate of \(\sigma_1\) is negative and is set to zero, i.e., \(\hat{\sigma}_1 = 0\). As before, GCM-L exhibits stability in its performance.

The last example is concerned with the prediction of county crop areas using satellite information (Battese, Harter, and Fuller, 1988). Let \(y_{ij}\) be the number of hectares of corn in the \(j\)th segment of the \(i\)th county, and let \(x_{1ij}\) and \(x_{2ij}\) represent the number of satellite pixels in a sample segment classified as corn and soybeans, respectively. This model is a nested-error regression model as following:

\[
y_{ij} = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + v_i + e_{ij}.
\]  \hfill (5.48)

where \(v_i\) is county effects. \(v_i\) and \(e_{ij}\) are random effect and random error, respectively, and are identically and independently distributed as Normal as in the other examples. Table 5.3 contains the results attained from applying the algorithms.
Note that in this example, the dimension of the corresponding $b$, vector of random effects, is relatively small compared to that of $y$. Since EM-type algorithms assume $b$ to be missing data, EM and EM-O algorithms converge quickly in this example. GCM-L being similar in structure to EM, also has a fast convergence rate and is even quicker than those algorithms and the FS algorithm.

Nonconvergence of GCM-L algorithm for the initial value $(\sigma_0^{(0)} = 1, \sigma_1^{(0)} = 1)$ is expected since GCM-L algorithm is guaranteed to converge only for initial values on \( \{\theta \in \Omega; Dl(\theta) < 0\} \).

<table>
<thead>
<tr>
<th>initial values</th>
<th>EM</th>
<th>GCM-L</th>
<th>FS</th>
<th>EM-O</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_0^{(0)} = 1$, $\sigma_1^{(0)} = 1$</td>
<td>0.633 (22)</td>
<td>0 (0)</td>
<td>0.733 (12)</td>
<td>0.567 (18)</td>
</tr>
<tr>
<td>$\sigma_0^{(0)} = 1000$, $\sigma_1^{(0)} = 1000$</td>
<td>0.683 (23)</td>
<td>0.583 (15)</td>
<td>0 (0)</td>
<td>0.567 (19)</td>
</tr>
<tr>
<td>$\sigma_0^{(0)} = 149.56$, $\sigma_1^{(0)} = 139.68$</td>
<td>0.517 (18)</td>
<td>0.383 (9)</td>
<td>0.517 (9)</td>
<td>0.433 (14)</td>
</tr>
</tbody>
</table>

Through the comparisons in the three examples, GCM-L algorithm outperforms the other algorithms in the sense of small convergence rate and wide range of initial values allowed that lead to convergence. In all of the examples, GCM-L is at least as fast as Newton's algorithm. This situation might change when $u$, the number of variance components becomes large. However, the correlations between variance components is generally expected to be low so that the difference between speeds of GCM-L and Newton's algorithm should not be large.

Furthermore, the proposed GCM algorithm could be used with these models in more complex situations such as in the presence of missing data and nonlinearity problems. In the next section, a structure with a missing data problem is considered.
5.4 New GCM Algorithms with True Missing Data in $y$

Assume some elements of $y$ are missing at random. For convenience, let $y_1$ and $y_2$ denote observed data and missing data, respectively. Also suppose that $Z_i^T$, and $Z_{i2}^T$ is the partitions of $Z_i$ corresponding to $y_1$ and $y_2$ while $X_i^T$, and $X_{i2}^T$ is the corresponding partitions of $X$. That is,

$$
y = (y_1^T, y_2^T)^T,$$

$$Z_i = (Z_{i1}^T, Z_{i2}^T)^T,$$

$$X = (X_1^T, X_2^T)^T,$$

and $V$ is decomposed into

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix},$$

where

$$V_{11} = \sum_{i=0}^{u} Z_{i1}Z_{i1}^T \sigma_i^2,$$

$$V_{12} = \sum_{i=0}^{u} Z_{i1}Z_{i2}^T \sigma_i^2,$$

$$V_{21} = \sum_{i=0}^{u} Z_{i2}Z_{i1}^T \sigma_i^2,$$

$$V_{22} = \sum_{i=0}^{u} Z_{i2}Z_{i2}^T \sigma_i^2.$$

The conventional EM algorithm for ML estimation of variance components sets $(y_2, b)$ as missing data and the corresponding conditional distributions of missing data given observed data are

$$b_i|y_1 \sim N(\sigma_i^2 Z_{i1}^T V_{11}^{-1}(y_1 - X_1 \beta), \sigma_i^2 I_q - \sigma_i^2 Z_{i1}^T V_{11}^{-1} Z_{i1}),$$

$$y_2|y_1 \sim N(X_2 \beta + V_{21} V_{11}^{-1}(y - X_1 \beta), V_{22} - V_{21} V_{11}^{-1} V_{12}),$$

respectively. Thus the $t$th iteration of the conventional EM algorithm is as follows:

$$\beta^{(t+1)} = E[(X^T V_{11}^{-1(t)} X)^{-1} X^T V_{11}^{-1(t)} y|y_1, \theta^{(t)}]$$

$$\sigma_i^{2(t+1)} = E[b_i^T b_i/q_i|y_1, \theta^{(t)}] \quad i = 1, \ldots, u,$$

$$\sigma_0^{2(t+1)} = \frac{1}{n} E[(y - X \beta^{(t)})^T(y - X \beta^{(t)})|y_1, \theta^{(t)}]$$
However, the complete data structure (or data augmentation) of this conventional algorithm is expected to cause slower convergence of the algorithm than the conventional EM algorithm without true missing data the speed of convergence of which was discussed in Section 5.1. This is because former algorithm has more missing data than the latter, and the convergence rate of an EM algorithm depends on the proportion of information amounts of missing data over those of complete data as shown in (2.25). Partial Aitken acceleration and overrelaxation may be applied to improve the speed of the EM algorithm as discussed in Subsection 4.6.2.

Another data augmentation scheme is to set only $y_2$ as missing data, that is, complete data is just $y = (y_1^T, y_2^T)^T$. Problem with this complete data structure is that there is no closed form of expression for $\sigma_i^2$, $i = 0, \cdots, u$ in the M-step of the resulting EM algorithm. One solution is to apply the nonlinear Jacobi algorithm using linearized quadratic approximation developed in Section 5.2. That is, as the E-step using (5.51), compute $E(W(\theta^{(t)})|y_1, \theta^{(t)})$ and $E(Dl_{L}(\theta^{(t)})|y_1, \theta^{(t)})$, perform an iteration of the nonlinear Jacobi algorithm given in (5.41).

Thus, the exact computation required in each iteration of this algorithm is

$$\theta^{(t+1)} = \theta^{(t)} - [E[W(\theta^{(t)})|y_1, \theta^{(t)}])^{-1} E[Dl_{L}(\theta^{(t)})|y_1, \theta^{(t)}],$$

(5.52)

where

$$Dl_{L}(\theta) = Dl(\theta)K(\theta),$$

$W(\theta)$ is block diagonal elements of $D^2l_{L}(\theta)$ for $(\beta, \sigma_0^2, \cdots, \sigma_u^2)$, and $K(\theta) = \text{blockdiag}([X^TV^{-1}X]^{-1}, \sigma_0^4, \cdots, \sigma_u^4)$.

Since the algorithm without true missing data is guaranteed to converge monotonely, this algorithm with true missing data is at least a GEM algorithm with monotone convergence as defined in Chapter 1 and has a faster convergence rate than the conventional EM.
Before deriving the required computational steps in detail, note that

\[
E(y|y_1, \theta) = (y_1^T, E(y_2^T|y_1, \theta))^T
\]

and that for a matrix

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\]

which is partitioned appropriately according to the dimensions of \(y_1\) and \(y_2\),

\[
E(y^T A y|y_1, \theta) = y_{imp}^T A y_{imp} + tr(A_{22}[V_{22} - V_{21}V_{11}^{-1}V_{12}]),
\]

where \(y_{imp} = E(y|y_1, \theta)\). Then by setting \(\beta = (X^T V^{-1}X)^{-1}X^T V^{-1}y\), we have

\[
E\left(\frac{\partial l(\theta)}{\partial \sigma_i^2}|y_1, \theta\right) = -0.5tr(V^{-1}Z_i Z_i^T) + 0.5 E(y^T P_i Z_i^T P_i y|y_1, \theta)
\]

\[
= -0.5tr(V^{-1}Z_i Z_i^T - P_2 Z_i Z_i^T P_2^T [V_{22} - V_{21}V_{11}^{-1}V_{12}])
\]

\[
+ 0.5 y_{imp}^T P_i Z_i^T P_i y_{imp},
\]

\[
E\left(\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2}|y_1, \theta\right) = 0.5tr(V^{-1}Z_i Z_i^T V^{-1}Z_i Z_i^T)
\]

\[
- tr(P_2 Z_i Z_i^T V^{-1}Z_i Z_i^T P_2^T [V_{22} - V_{21}V_{11}^{-1}V_{12}])
\]

\[
- y_{imp}^T P_i Z_i^T V^{-1}Z_i Z_i^T P_i y_{imp},
\]

where \(P = V^{-1} - V^{-1}X[X^T V^{-1}X]^{-1}X^T V^{-1}\), and \(P_1\) and \(P_2\) are partitions of \(P\) corresponding to that of \(y\), respectively, i.e., \(P_i, i = 1, 2\) has the same number of rows as \(y_i\).

Now, the \(t\)th iteration of the new GCM algorithm is

\[
\beta^{(t+1)} = (X^T V^{-1(t)}X)^{-1}X^T V^{-1(t)}y_{imp}, \quad (5.54)
\]

\[
\sigma_j^{2(t+1)} = \frac{\sigma_j^{2(t)} E\left(\frac{\partial l(\theta)}{\partial \sigma_i^2}|y_1, \theta\right)|_{\sigma_i^2 = \sigma_j^{2(t)}}}{2 \sigma_j^{2(t)} E\left(\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2}|y_1, \theta\right)|_{\sigma_i^2 = \sigma_j^{2(t)}} + \sigma_j^{2(t)} E\left(\frac{\partial l(\theta)}{\partial \sigma_j^2}|y_1, \theta\right)|_{\sigma_j^2 = \sigma_j^{2(t)}}}
\]

for \(j = 0, \ldots, u\).
Of course, the first step in (5.54) can be omitted from the each iteration and computed once only at convergence by setting \( V^{-1}(y - X\beta) = V^{-1}y - V^{-1}X[X^T V^{-1}X]X^T V^{-1}y \).

The computational burden of the new GCM algorithm includes the E-step (computing conditional mean and variance of missing data), computing \( P_2Z_iZ_i^TP_i^T \) and \( P_2Z_iZ_i^TV^{-1}Z_iZ_i^TP_i^T \). Note that the \( W \) transformation also needs to be computed as in the case of no true missing data. As usual, for the E-step in the case of the normal distribution, the sweep operator can be applied to transform

\[
V = \begin{bmatrix}
V_{11} & V_{12} \\
V_{21} & V_{22}
\end{bmatrix},
\]

into

\[
\begin{bmatrix}
-V_{11}^{-1} & V_{11}^{-1}V_{12} \\
V_{21}V_{11}^{-1} & V_{22} - V_{21}V_{11}^{-1}V_{12}
\end{bmatrix},
\]

Thus, \( E(y_2|y_1) = X_2\beta + V_{21}V_{11}^{-1}(y - X_1\beta) \), and \( Var(y_2|y_1) = V_{22} - V_{21}V_{11}^{-1}V_{12} \) are easily computed. Since \( tr(A^T B) = vec(A)^T vec(B) \) for any matrix \( A \) and \( B \) with appropriate dimension, computing trace of product of two matrices is trivial.

However, separate computations of \( P_2Z_iZ_i^TP_i^T \) and \( P_2Z_iZ_i^TV^{-1}Z_iZ_i^TP_i^T \) may require high dimensional matrix operations. For solving this problem, the \( W \) transformation, introduced in Subsection 5.2.1 is adopted as described below.

Let \( E_2 \) is right partition of an \( n \times n \) identity matrix partitioned by columns to have the same column dimensions as those of \( y_1 \) and \( y_2 \), i.e, \( I = (E_1, E_2) \) With the use of this matrix, we can express the new matrix \( W_{m0} \) as

\[
W_{m0} = \begin{bmatrix}
Z_{tot}^TZ_{tot} & Z_{tot}^TX & Z_{tot}^TY & Z_{tot}^TE_2 \\
X^TZ_{tot} & X^TX & X^TY & X^TE_2 \\
y^TZ_{tot} & y^TX & y^TY & y^TE_2 \\
E_2^TZ_{tot} & E_2^TX & E_2^TY & E_2^TE_2
\end{bmatrix}
\] (5.55)
and then compute the $W$ transformation as follows:

$$W_m = \begin{bmatrix} Z_{tot}^TV^{-1}Z_{tot} & Z_{tot}^TV^{-1}X & Z_{tot}^TV^{-1}y & Z_{tot}^TV^{-1}E_2 \\ X^T V^{-1}Z_{tot} & X^T V^{-1}X & X^T V^{-1}y & X^T V^{-1}E_2 \\ y^T V^{-1}Z_{tot} & y^T V^{-1}X & y^T V^{-1}y & y^T V^{-1}E_2 \\ E_2^TV^{-1}Z_{tot} & E_2^TV^{-1}X & E_2^TV^{-1}y & E_2^TV^{-1}E_2 \end{bmatrix}, \quad (5.56)$$

where $Z_{tot} = [Z_1, \ldots, Z_u]$. Since

$$P_2Z_i = E_2^TPZ_i$$

$$= E_2^TV^{-1}Z_i - E_2^TV^{-1}X(X^T V^{-1}X)^{-1} XV^{-1}Z_i,$$

with this extended version of the $W$ transformation, computation of the additional quantities is simplified.

In fact, $E_1$ and $E_2$ are also useful for obtaining conditional expectation and conditional variance of missing data. Assume $S = SWEEP[1, \ldots, q]V$ and $q$ is the number of missing data in $y$ (see Appendix A for sweep operator in detail). Then, we have the following formulas:

$$E(y_2|y_1) = E_2^TX\beta + E_2^TSE_1E_1^T(y - X\beta)$$

$$\text{Var}(y_2|y_1) = E_2^TSE_2.$$

So far, we have assumed that missing data occurs in bottom part of $y$ for convenience. Although we can reorder data so that this situation is always true, the naturally occurring order of missing data is sometimes important or impossible to change. The case of times series data is an example. In this generalized case, the computation required for the new GCM algorithm is nearly the same as in the missing data case discussed.

Let $k_1 = (k_{11}, \ldots, k_{1(n-q)})$ and $k_1 = (k_{21}, \ldots, k_{2q})$ be partition of $(1, \ldots, n)$ where $k_{2j}, j = 1, \ldots, q$ is the location of $j$th missing data in $y$ and $k_{1j}, j = 1, \ldots, n - q$ is the location of $j$th observed data in $y$. Set $E_2$ to consist of all $k_{2j}$th columns of $n \times n$
identity matrix, and \( E_1 \) of all \( k_1 \)-th columns of \( n \times n \) identity matrix. Then only sweeping \((k_{11}, \ldots, k_{1(n-q)})\) out of \( V \) instead of sweeping \((1, \ldots, n-q)\) produces exactly the same results as in monotone missing data case. Thus our new GCM algorithm for missing data requires only sweeping and the extended \( W \) transformation in each iteration.

5.4.1 Simple Example for Missing Data in \( y \)

Here, an example of the GCM algorithm for missing data in \( y \) is presented. The GCM algorithm is a missing data version of GCM-L which was discussed earlier. Any comparison of this GCM algorithm with other methods were not conducted since it was previously showed that GCM-L algorithm is more efficient than either EM or Newton's algorithms. The satellite data was used in previous empirical comparisons in no true-missing-data case. To experiment with missing data, three observations, 12th, 23th, 36th observations in natural order, are assumed to be missing. Assuming that \( k_1 = (k_{11}, \ldots, k_{1(n-q)}) \), \( k_1 = (k_{21}, \ldots, k_{2q}) \), \( E_1 \), and \( E_2 \) have the definitions described in Section 5.4, the following is the specific description of the algorithm:

**Missing Data Version of GCM-L**

For \( t = 0, 1, \ldots \),

1. Compute \( S = S \omega = [k_1] \omega V \), \( y_{imp}^{(t)} = E_1 E_1^T y + E_2 E_2^T (X^T \beta^{(t)} + SE_1 E_1^T (y - X \beta)) \), and \( V_{y_2}^{(t)} = E_2 S^{(t)} E_2 \)

2. Set \( \beta^{(t+1)} = [X^T H^{-1(t)} X]^{-1} X^T H^{-1(t)} y_{imp} \) and \( \sigma_0^{2(t+1)} = \frac{1}{n} (y_{imp} - X \beta^{(t+1)})^T H^{-1(t)} (y_{imp} - X \beta^{(t+1)}) + tr(E_2^T P(t) E_2 V_{y_2}^{(t)}) \)

3. Compute \( \sigma^{(t+1)} = \sigma^{(t)} - [L_{GCM}(\sigma^{(t)})]Dl(\sigma^{(t)}) \).

4. If \(||\theta^{(t+1)} - \theta^{(t)}|| > \epsilon\) for a given \( \epsilon \) then go to 1; otherwise accept current iterate \( \theta^{(t+1)} \) as the limit of the sequence \( \{\theta^{(t)}\} \),
where $L_{GCM}(\theta)$ is diagonal matrix whose elements are $(\sigma_i^1/d_i(\theta), \cdots, \sigma_n^1/d_n(\theta))$, $d_i(\theta) = 2\sigma_i^2d_i + \sigma_i^1\eta_i$, $d_i$ is the $i$th element of $E(Dl(\sigma)|y_1)$, and $\eta_i$ is the $i$th diagonal element of $E(D^2l(\sigma)|y_1)$.

Using this algorithm and the original version of GCM-L which was defined in Section 5.3, MLEs were found for three cases of data structures. The first case (ORG) is the original complete data in which 12th, 23th, and 36th observations were present. Scatterplot matrix in Figure 5 shows the structure of the complete data and indicates the artificially missing data. The second case of data (CC) is established by omitting the above three observations from the data entirely. The last case (MISS) is our interesting data structure where the responses are missing for the above three observations. For the first and the second cases, the original GCM-L algorithm is used and for the third case, the missing data version of GCM-L given above. Table 5.4 shows the results for three data sets.

Table 5.4  The results of ML estimation for example of missing data in $y$ with Battese et al's(1988) model

<table>
<thead>
<tr>
<th>Case</th>
<th>$\sigma_0$</th>
<th>$\sigma_1$</th>
<th>$\beta_j, j = 0, 1, 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>121.062</td>
<td>137.313</td>
<td>50.967, 0.328, -0.133</td>
</tr>
<tr>
<td>CC</td>
<td>116.772</td>
<td>139.643</td>
<td>50.259, 0.335, -0.138</td>
</tr>
<tr>
<td>MISS</td>
<td>117.070</td>
<td>138.798</td>
<td>50.251, 0.335, -0.138</td>
</tr>
</tbody>
</table>

Interestingly, the difference between MLEs of $\beta$ in two cases appear to be minimal while values of variance components estimates for MISS are located between those for CC and ORG. GCM algorithm for MISS case takes 30 iterations in finding the MLE. This implies that missing data does not heavily affect on the speed of GCM-L algorithm since the original version of GCM-L algorithm needed 15 iterations to obtain MLEs.
Figure 5.3 Scatter Plot Matrix of Satellite Data, var 0 = v, var 1 = 'Segment', var 2 = y, var 3 = x₁, and var 4 = x₂
5.5 Discussion

In this chapter, extensions of GCM algorithm for variance components models are studied. GCM-L, one of the GCM algorithms shows efficiency in performance with respect to the conditions [a]-[e] which are presented in Chapter 1. The algorithm does not require any linear search for monotone convergence of \( \{l(\theta^{(t)})\} \) in each iteration nor take unnecessary steps as EM does so that the GCM improves speed of convergence of the EM algorithm. Furthermore, any set of the initial values which are sufficiently large guarantee convergence of the sequence \( \{\theta^{(t)}\} \) by linearization theorem in Chapter 2.

The algorithm is specific to the model described. The performance of this scheme for other models is unknown. As evinced from examples shown so far, developing efficient algorithms using this technique generally depends on the characteristics of the specific model. The purpose of this dissertation is to investigate appropriate guidelines for developing improved GCM algorithms by applying it to specific examples.

GCM scheme appears to have great potential for improving conventional ML algorithms for many models. Chapter 6 introduces some examples which are expected to be improved with appropriate GCM algorithms and describes the corresponding GCM algorithms.
6 FRAMEWORK FOR MORE APPLICATIONS OF THE GCM ALGORITHMS FOR FUTURE STUDY

6.1 Introduction

In this chapter, we introduce several models for which GCM algorithms are developed for ML estimation of the parameters. Because of time limitations and unavailability of data, simulation studies or numerical studies with real data were not conducted for these models. However, GCM algorithms presented in this chapter can be used with appropriate data without any change or modifications while they can be extended for other similar models with suitable modifications in the algorithms.

As a generalization of the GCM algorithms for variance component model with missing data in y, we establish variance component model with missing data in X. This requires the assumption of a specific prior distributions for X. The GCM algorithm under a multivariate normal for X is developed in Section 6.2. Section 6.3 introduces a neural network with a specific variance component structure and derive corresponding GCM algorithm.

6.2 New GCM Algorithms with Missing Data in X for Variance Component Model

Suppose that, excluding the intercept column, the first q columns of X have missing data. Also assume that each independent variables corresponding to the each column of
X matrix are continuous variable which is approximately distributed as Gaussian and that the missing data in the variables occur at random.

To handle this missing data problem, we need to impose conditions, specifically prior probability distributional assumptions on the variables. Let \( x_1, \ldots, x_p \) denote the column vectors of \( X \) and \( x'_1, \ldots, x'_n \) the row vectors of \( X \). Traditional approach to missing data in \( X \), especially in regression problem, is to assume that the \( x'_i \)'s are independent of each other and that each \( x'_i \) is distributed as \( N(\mu_{x'_i}, \Sigma_{x'_i}) \).

But sometimes, \( x'_i \)'s, column vectors of \( X \), might not be correlated with each other presuming appropriate variable selection in \( X \), but the elements of \( x'_i \) might be correlated with each other in the same \( x'_i \). We may assume that this correlation has the same design structure as that of \( y \). Thus, a feasible assumption in this case is that \( x'_i \)'s are independent of each other and that each \( x'_i \) is distributed as \( N(\mu_{x'_i}, I) \), where \( \Sigma = \Sigma_x + \Sigma_y \) and \( \Sigma_x \) is the same notation as in (5.1). That is,

\[
x'_i = \mu_{x'_i} + \sum_{j=1}^u Z_j b_{j(x'_i)} + e_{x'_i},
\]

where each \( b_{j(x'_i)}, j = 1, \ldots, u \) is a random vector independent of each other, independent of \( e_{x'_i} \), and distributed as \( N(0, \sigma_j^2 I) \), and the distribution of \( e_{x'_i} \) is \( N(0, \sigma_0^2 I) \). In this section, all derivations for GCM algorithm are based on this prior.

The priors will give us the following joint distribution for \( (y, x_1, \ldots, x_q) \):

\[
\begin{bmatrix}
y \\
x_1 \\
\vdots \\
x_q
\end{bmatrix} ~ N
\begin{bmatrix}
\mu_y \\
\mu_{x_1} \\
\vdots \\
\mu_{x_q}
\end{bmatrix},
\begin{bmatrix}
V_y & \beta_1 V & \beta_2 V & \cdots & \beta_q V \\
\beta_1 V & V & 0 & \cdots & 0 \\
\vdots & 0 & \ddots & \ddots & \vdots \\
\beta_q V & 0 & \cdots & 0 & V
\end{bmatrix},
\]

where \( \mu_y = \beta_0 I + \sum_{i=1}^q \beta_i \mu_{x_i} + \sum_{i=q+1}^p \beta_i x_i \), and \( V_y = (1 + \sum_{i=1}^q \beta_i^2) V \). Note that each \( x_i \) is not correlated with each other, but only with \( y \) and \( \text{Cov}(y, x_i) = \beta_i V_{x_i} \).

If no data were missing, we would maximize \( \log(p(y|\theta, x_1, \ldots, x_p) \prod_{i=1}^p p(x_i|\theta_i)) = l(\theta) + \sum_{i=1}^q l_{x_i}(\mu_{x_i}) \). This problem can be solved by extending the GCM algorithms.
without true missing data which was discussed in Section 5.3. That is, the \( t \)th iteration of GCM (nonlinear Jacobi using linearized quadratic approximation) is, for \( i = 1, \cdots, q \),

\[
\mu_{x_i} = (I^T V^{-1} 1)^{-1} I^T V^{-1} x_i, \quad (6.3)
\]

and the exact same formula as in (5.41),

\[
\theta^{(t+1)} = \theta^{(t)} - [W(\theta^{(t)})]^{-1} DL_L(\theta^{(t)}), \quad (6.4)
\]

where

\[
DL_L(\theta) = DL(\theta) K(\theta),
\]

\[
W(\theta) \quad \text{is block diagonal elements of } D^2 l_L(\theta) \text{ for } (\beta, \sigma_0^2, \cdots, \sigma_q^2),
\]

and \( K(\theta) = \text{blockdiag}([X^T V^{-1} X]^{-1}, \sigma_0^4, \cdots, \sigma_q^4) \).

Since we are interested in \( \theta = (\beta, \sigma_0^2, \cdots, \sigma_q^2) \), with complete data, we only need (6.4) which is the same as that without true missing data.

In the case of missing data in \( X \) and no missing data in \( y \), we will assume with loss of generality that the first \( q \) predictor variables include the missing data. However, in this case, we have to compute conditional expectation and conditional variance of missing data given observed data. Since conditional distribution of \( x_1, \cdots, x_q \) given \( y \) is

\[
N \left( \begin{bmatrix}
\mu_{x_1|y} \\
\mu_{x_2|y} \\
\vdots \\
\mu_{x_q|y}
\end{bmatrix},
\begin{bmatrix}
V_{x_1|x_1|y} & V_{x_1|x_2|y} & \cdots & V_{x_1|x_q|y} \\
V_{x_2|x_1|y} & V_{x_2|x_2|y} & \cdots & V_{x_2|x_q|y} \\
\vdots & \vdots & \ddots & \vdots \\
V_{x_q|x_1|y} & V_{x_q|x_2|y} & \cdots & V_{x_q|x_q|y}
\end{bmatrix}\right), \quad (6.5)
\]

i.e.,

\[
E(x_i|y) = \mu_{x_i|y} = \mu_i 1 + \frac{\beta_i}{1 + \sum_{j=1}^q \beta_j^2} (y - \mu_y)
\]

\[
Var(x_i|y) = V_{x_i|x_i|y} = (1 - \frac{\beta_i^2}{1 + \sum_{j=1}^q \beta_j^2}) V
\]

\[
Cov(x_i, x_k|y) = V_{x_i|x_k|y} = \frac{\beta_i \beta_k}{1 + \sum_{j=1}^q \beta_j^2} V.
\]
By applying sweep operator (Appendix A) to these conditional variance-covariance matrices of \( x_i \) given \( y \), E-step which computed the first derivative function of observed loglikelihood function can be easily obtained.

Let \( x_{i(obs)} \) and \( x_{i(mis)} \), \( i = 1, \cdots, q \) be observed data and missing data portions of \( x_i \), respectively. And let \( k_{i(obs)} = (k_{i1(obs)}, \cdots, k_{in(obs)}) \) and \( k_{i(mis)} = (k_{i1(mis)}, \cdots, k_{im(mis)}) \) denote location vectors of observed data and missing data in \( x_i \), respectively, i.e., \( k_{ij(mis)} \)th element of \( x_i \) is \( j \)th in \( x_{i(mis)} \). Here, \( o_i \) is the number of elements in \( x_{i(obs)} \) and \( m_i \) the number of elements in \( x_{i(mis)} \). Set \( E_{2i} \) to consist of \( k_{i(mis)} \) columns of the \( n \times n \) identity matrix, and \( E_{1i} \) to consist of \( k_{i(obs)} \) columns of the \( n \times n \) identity matrix.

Lemma

For a symmetric matrix \( A, x_i, \) and \( x_j, i = 1, \cdots, q, j = 1, \cdots, q \)

\[
E(x_i|z_{obs}) = E_{1i}x_{i(obs)} + E_{2i}E(x_{i(mis)}|z_{obs})
\]

\[
E(x_i^T A x_j|z_{obs}) = E(x_i^T z_{obs}) A E(x_j|z_{obs}) + tr(E_{2j} A E_{2i} \text{Cov}(x_{i(mis)}, x_{j(mis)}|z_{obs}))
\]

where \( z_{obs} = (y, x_{1(obs)}, \cdots, x_q(obs)) \)

Proof

It’s trivial to show the result from the fact \( I = E_{1i} E_{1i}^T + E_{2i} E_{2i}^T \) for each \( i = 1, \cdots, q \)

Let \( X_{imp} \) denotes

\[
X_{imp} = E(X|z_{obs}) = (1, E(x_1|z_{obs}), \cdots, E(x_q|z_{obs}), x_{q+1}, \cdots, x_p)
\]

Lemma

\[
E(X^T V^{-1} X|y_{obs}), E((y - X \beta)^T V^{-1} Z_j Z_j^T V^{-1} (y - X \beta)|z_{obs}), \text{and}
\]

\[
E((y - X \beta)^T V^{-1} Z_j Z_j^T V^{-1} Z_j Z_j^T V^{-1} (y - X \beta)|z_{obs}) \]

can be expressed respectively as follows:

\[
X_{imp}^T V^{-1} X_{imp} + F,
\]

\[
(y - X_{imp} \beta)^T V^{-1} Z_j Z_j^T V^{-1} (y - X_{imp} \beta) + \beta^T G_j \beta,
\]

\[
(y - X_{imp} \beta)^T V^{-1} Z_j Z_j^T V^{-1} Z_j Z_j^T V^{-1} (y - X_{imp} \beta) + \beta^T K_j \beta,
\]
where all elements of (1, q + 1, \cdots, p) columns and rows of F, G_j, and K_j are zeros, and for \( i = 1, \cdots, q, k = 1, \cdots, q \), each \((i + 1, k + 1)\) element of F, G_j, and K_j is respectively

\[
tr(E_{2k}^T V^{-1} E_{2k} P x Z_j Z_j^T V^{-1} E_{2k} P x E_{2k} V x_{i(mia)} | z_{obs})
\]

For deriving the algorithm in detail, we have the following quantities: for \( i = 1, \cdots, q, j = 0, \cdots, u \),

\[
E(\frac{\partial l_i}{\partial \mu_i} | z_{obs}, \theta) = 1^T V^{-1} x_i^{imp} - 1^T V^{-1} 1 \mu_1,
\]

\[
E(\frac{\partial^2 l_i}{\partial \mu_i^2} | z_{obs}, \theta) = 1^T V^{-1} 1,
\]

\[
E(\frac{\partial l_i}{\partial \sigma_j^2} | z_{obs}, \theta) = -0.5 tr(V^{-1} Z_j Z_j^T - E_{2i} P x Z_j Z_j^T P x E_{2i} V x_{i(mia)} | z_{obs}) + 0.5(x_i^{imp})^T P x Z_j Z_j^T P x x_i^{imp},
\]

\[
E(\frac{\partial^2 l_i}{\partial \sigma_j^2} | z_{obs}, \theta) = 0.5 tr(V^{-1} Z_j Z_j^T V^{-1} Z_j Z_j^T - tr(E_{2i} P x Z_j Z_j^T V^{-1} Z_j Z_j^T P x E_{2i} V x_{i(mia)} | z_{obs}) - (x_i^{imp})^T P x Z_j Z_j^T V^{-1} Z_j Z_j^T P x x_i^{imp},
\]

\[
E(\frac{\partial l}{\partial \beta} | z, \theta) = X_{imp}^T V^{-1} y - X_{imp}^T V^{-1} X_{imp} \beta - F \beta,
\]

\[
E(\frac{\partial l}{\partial \sigma_j^2} | z, \theta) = -0.5 tr(V^{-1} Z_j Z_j^T) + 0.5(y - X_{imp} \beta)^T V^{-1} Z_j Z_j^T V^{-1}(y - X_{imp} \beta) - \beta^T G_j \beta,
\]

\[
E(\frac{\partial^2 l}{\partial \beta^2} | z, \theta) = -X_{imp}^T V^{-1} X_{imp} - F,
\]

\[
E(\frac{\partial^2 l}{\partial \sigma_j^2} | z, \theta) = 0.5 tr(V^{-1} Z_j Z_j^T V^{-1} Z_j Z_j^T) - (y - X_{imp} \beta)^T V^{-1} Z_j Z_j^T V^{-1} Z_j Z_j^T V^{-1}(y - X_{imp} \beta) - \beta^T K_j \beta,
\]
where $x_{i}^{\text{imp}} = E(x_{i}|z_{obs})$, $V_{x_{i}(\text{mis})}z_{obs} = Var(x_{i}(\text{mis})|z_{obs})$, $z_{obs} = (y, x_{1(\text{obs})}, \ldots, x_{q(\text{obs})})^{T}$, and $P_{z} = V^{-1} - V^{-1}11^{T}V^{-1}11^{T}V^{-1}$.

Using the formula above, for $i = 1, \ldots, q, j = 0, \ldots, u$, the $t$th iteration of the new GCM algorithm is

\begin{align*}
\mu^{(t+1)}_{i} &= (1^{T}V^{-1}1)^{-1}1^{T}V^{-1}x_{i}^{\text{imp}}, \\
\beta^{(t+1)} &= (X_{\text{imp}}^{T}V^{-1(t)}X_{\text{imp}} + F)^{-1}X_{\text{imp}}^{T}V^{-1(t)}y, \\
\sigma^{2(t+1)}_{j} &= \sigma^{2(t)}_{j} - \frac{\sigma^{4(t)}_{j} E(\frac{\partial l_{i}(\theta)}{\partial \sigma^{2}_{j}}|z, \theta)|_{\sigma^{2} = \sigma^{2(t)}}}{2\sigma^{2(t)}_{j} E(\frac{\partial l_{i}(\theta)}{\partial \sigma^{2}_{j}}|z, \theta)|_{\sigma^{2} = \sigma^{2(t)}} + \sigma^{4(t)}_{j} E(\frac{\partial^{2}l_{i}(\theta)}{(\partial \sigma^{2}_{j})^{2}}|z, \theta)|_{\sigma^{2} = \sigma^{2(t)}}},
\end{align*}

where

\begin{align*}
\frac{\partial l_{i}(\theta)}{\partial \sigma^{2}_{j}} &= \frac{\partial l(\theta)}{\partial \sigma^{2}_{j}} + \sum_{i=1}^{q} \frac{\partial l_{x_{i}}(\theta_{x_{i}})}{\partial \sigma^{2}_{j}} \\
\frac{\partial^{2}l_{i}(\theta)}{(\partial \sigma^{2}_{j})^{2}} &= \frac{\partial^{2}l(\theta)}{(\partial \sigma^{2}_{j})^{2}} + \sum_{i=1}^{q} \frac{\partial^{2}l_{x_{i}}(\theta_{x_{i}})}{(\partial \sigma^{2}_{j})^{2}}.
\end{align*}

In addition to sweeping a $qn \times qn$ matrix, the computation for the model with missing data in $X$ includes a $W$ transformation which is even more extended than the one needed with missing data in $y$.

That is, performing $W$ transformation of

\begin{equation}
W_{m0} = \begin{bmatrix}
Z_{\text{tot}}^{T}Z_{\text{tot}} & Z_{\text{tot}}^{T}X & Z_{\text{tot}}^{T}y & Z_{\text{tot}}^{T}E_t \\
X^{T}Z_{\text{tot}} & X^{T}X & X^{T}y & X^{T}E_t \\
y^{T}Z_{\text{tot}} & y^{T}X & y^{T}y & y^{T}E_t \\
E_{t}^{T}Z_{\text{tot}} & E_{t}^{T}X & E_{t}^{T}y & E_{t}^{T}E_t
\end{bmatrix},
\end{equation}

leads to the easy access the following quantities

\begin{equation}
W_{m} = \begin{bmatrix}
Z_{\text{tot}}^{T}V^{-1}Z_{\text{tot}} & Z_{\text{tot}}^{T}V^{-1}X & Z_{\text{tot}}^{T}V^{-1}y & Z_{\text{tot}}^{T}V^{-1}E_t \\
X^{T}V^{-1}Z_{\text{tot}} & X^{T}V^{-1}X & X^{T}V^{-1}y & X^{T}V^{-1}E_t \\
y^{T}V^{-1}Z_{\text{tot}} & y^{T}V^{-1}X & y^{T}V^{-1}y & y^{T}V^{-1}E_t \\
E_{t}^{T}V^{-1}Z_{\text{tot}} & E_{t}^{T}V^{-1}X & E_{t}^{T}V^{-1}y & E_{t}^{T}V^{-1}E_t
\end{bmatrix},
\end{equation}
where $Z_{tot} = [Z_1, \cdots, Z_u]$ and $E_t = [E_{21}, \cdots, E_{2q}]$.

The GCM algorithm described above applies in the case of the selected prior only so that the algorithm may need to be adjusted for specific data, in which case a different prior may be appropriate.

### 6.2 Neural Network Prediction Model

The neural network was constructed in order to mimic the behavior of human brain in recognizing the pattern of an object. Since late 80's, neural networks have become one of popular methods for modeling artificial intelligence, especially, machine learning, in various areas in engineering where sophisticated monitoring of phenomena or events were needed. There are more than ten official journals and thousands of books for topics related with this method. As a statistician, Ripley(1994) presents an introduction to neural network and a wide ranging comparison between neural network methodology and other statistical methods. He pointed out that neural networks are not always better than other statistical methods, especially nonparametric nonlinear regression methods like MARS and projection pursuit regression. It performs better in some cases, but worse in others.

The feed forward neural network with a one hidden layer is the most popular one among the various neural networks. This neural network has three layer, input layer, hidden layer, and output layer. Assume that we have a data set $\{y_i, x_i\}_{i=1}^n$ and the hidden layer of neural network has $H$ nodes. For $h = 1, \cdots, H$, each $h$th node in the hidden layer of the neural network is assigned the value obtained from a nonlinear function $f_h(w_h^T x)$ of weighted sum $w_h^T x$ of input data $x_i$ of input layer where $x_{i0}$, first element of $x_i$ is 1 for intercept, and $w_h$ is unknown coefficient vector. Then output layer has a value obtained from its nonlinear function

$$f_0(\beta_0 + \sum_{h=1}^H \beta_h f_h(\cdot)) \quad (6.8)$$
of weighted sum $\beta_0 + \sum_{h=1}^{H} \beta_h f_h(\cdot)$ in the hidden layer. That is, each pair of $y_i$ and $x_i$, $i = 1 \cdots, n$ is assumed to have the following relationship:

$$y_i = f_0(\beta_0 + \sum_{h=1}^{H} \beta_h f_h(w_h^T x_i)) + \epsilon_i,$$

(6.9)

where $\epsilon_i$, $i = 1, \ldots, n$ is independent of each other and assumed to be distributed with $N(0, \sigma_0^2)$ where $\sigma_0^2$ is unknown. This assumption on distribution of error term $\epsilon_i$ is unusual, but is necessary when we consider the estimation of parameters in this neural network as ML estimation (Faraggi and Simon, 1995). Also, note that $f_h, h = 0, \ldots, H$ are assumed to be known.

6.2.1 GCM Algorithm for Neural Network Prediction Model

Although many types of neural networks have been developed so far, in this section we consider only neural network prediction (NNP) model, a type of feed forward neural networks which has a linear function as $f_0(\cdot)$ and the function $\frac{\exp(\cdot)}{1+\exp(\cdot)}$ as $f_h(\cdot), h = 1, \ldots, H$. Formally, NNP model has the following structure:

$$y_i = \beta_0 + \sum_{h=1}^{H} \frac{\beta_h}{1 + \exp(-w_h^T x_i)} + \epsilon_i$$

(6.10)

where $\epsilon_i$, $i = 1, \ldots, n$ has the same properties above. The NNP has been proved to approximate any measurable function in measure and $L_p$ as the number of hidden nodes becomes large (Ripley, 1994).

In fact, assuming normally distributed random error for neural networks is for the statistical convenience of being able to fit the model as a nonlinear regression (Faraggi and Simon, 1995). In this setting, it is obvious that ML estimation of the parameters of the model is the same as least square (LS) estimation except that one ignores the degrees of freedom for estimating $\sigma_0^2$. Note that there are many other possible assumptions for error which may lead to different estimation methods. We, however, concentrate only on ML estimation with normal errors in this section.
Interpretation of each parameter of this neural network is not necessary nor possible. The estimation is only necessary for prediction of a multivariate nonlinear function. The absence of parameter inference is one reason that many statistician have avoided the study of neural networks.

From equation (6.10), the loglikelihood is

$$l(\theta) = -0.5n \log \sigma_0^2 - \frac{0.5}{\sigma_0^2} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{h=1}^{H} \frac{\beta_h}{1 + \exp(-w_h^T x_i)})^2$$  

(6.11)

and its first and second derivatives are as follows:

$$\frac{\partial l(\theta)}{\partial \beta} = \frac{0.5}{\sigma_0^2} \mathbf{D}^T (\mathbf{y} - \mathbf{D}\beta),$$

$$\frac{\partial l(\theta)}{\partial w_{hj}} = \frac{0.5}{\sigma_0^2} c^T_j \mathbf{y} - \mathbf{D}\beta), \quad \text{for } j = 0, \cdots, p, \ h = 1, \cdots, H,$$

$$\frac{\partial l(\theta)}{\partial \sigma_0^2} = -0.5 \frac{0.5}{\sigma_0^2} (\mathbf{y} - \mathbf{D}\beta)^T (\mathbf{y} - \mathbf{D}\beta),$$

$$\frac{\partial^2 l(\theta)}{\partial \beta \partial \beta^T} = -\frac{0.5}{\sigma_0^2} \mathbf{D}^T \mathbf{D},$$

$$\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial w_{kl}} = -\frac{0.5}{\sigma_0^2} c^T_j d_k$$

for $k \neq h$,

$$= -\frac{0.5}{\sigma_0^2} c^T_j d_k + \frac{0.5}{\sigma_0^2} c^T_j (\mathbf{y} - \mathbf{D}\beta)$$

for $k = h$,

$$\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial \sigma_0^2} = -\frac{0.5}{\sigma_0^2} c^T_j c_{hj}$$

for $k \neq h$,

$$= -\frac{0.5}{\sigma_0^2} c^T_j c_{hj} + \frac{0.5}{\sigma_0^2} e^T_{hj} (\mathbf{y} - \mathbf{D}\beta)$$

for $k = h$,

$$\frac{\partial^2 l(\theta)}{\partial \beta \partial \sigma_0^2} = -\frac{0.5}{\sigma_0^2} \mathbf{D}^T (\mathbf{y} - \mathbf{D}\beta),$$

$$\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial \sigma_0^2} = -\frac{0.5}{\sigma_0^2} c^T_j (\mathbf{y} - \mathbf{D}\beta),$$

where $\mathbf{D}$ is an $n \times (H + 1)$ matrix whose element in $i$th row and $(h + 1)$ column is

$$\frac{1}{1 + \exp(-w_h^T x_i)} \quad \text{for } h = 1, \cdots, H,$$

(6.12)

(Obviously every element in the first column of $\mathbf{D}$ is 1 for intercept term), $\mathbf{d}_h$ is $n \times 1$ vector corresponding to $(h + 1)$th column of $\mathbf{D}$, $c_{hj}$ is $n \times 1$ vector whose $i$th element is

$$c_{hji} = \frac{x_{ij} \beta_h \exp(-w_h^T x_i)}{(1 + \exp(-w_h^T x_i))^2},$$
and $e_{hji}$ is $n \times 1$ vector whose $i$th element is

$$e_{hji} = -\frac{\beta_h x_{ij} x_{hi} [\exp(0.5 w_h^T x_i) - \exp(-0.5 w_h^T x_i)]}{[\exp(0.5 w_h^T x_i) + \exp(-0.5 w_h^T x_i)]^3}$$

(6.13)

Originally, estimation for the feed forward neural network is implemented by backpropagation method, a kind of steepest ascent which has as its iteration function

$$M_{st}(\theta) = \theta - s Dl(\theta).$$

Backpropagation chooses its step size $s$ as a small value to ensure convergence of $\{\theta^{(t)}\}$ to a local maximum. This strategy makes the algorithm obviously very slow.

On the other hand, statisticians often use a quadratic approximation method with linear search like the modified Newton's or a quasi-Newton method for improvement in the speed of convergence. However, when the number of hidden nodes is large, this approach results in the 'curse of dimensionality', that is, the problem of quadratically increasing memory needed, as the dimension of the model increases. For example, when dimension of $x_i$ is 5 and number of nodes hidden nodes of neural network prediction model is 4, $20 + 5 + 1 = 26$ parameters must be estimated and a $26 \times 26$ Hessian matrix evaluated at each iteration of a quadratic approximation method. Linear search for monotone convergence at each iteration adds to the already heavy computational burden of this method.

To overcome some of the disadvantages of the previous two methods, we present below a GCM algorithm, which is a composite of nonlinear Gauss-Seidel and nonlinear Jacobi using linearized quadratic approximation method which is mentioned in Chapter 2.

Given $\theta^{(0)} = (\beta^{(0)}, \pi_1^{(0)}, \cdots, \pi_H^{(0)}, \sigma_0^{(0)})$, the $t + 1$th iteration of the algorithm is

$$\beta^{(t+1)} = (D^{(t)} D^{(t)})^{-1} D^{(t)} y,$$

$$\pi_{hj}^{(t+1)} = \frac{D l_L(\theta^{(t)})}{D^2 l_L(\theta^{(t)})} \quad \text{for} \ h = 1, \cdots, H, \ j = 0, \cdots, p.$$
where

\[ \pi_{hj} = \exp(w_{hj})/(1 + \exp(w_{hj})) \]

\[ Dl(\theta) = \frac{\partial l(\theta)}{w_{hj}} \left( \frac{1}{1 - \pi_{hj}} \right) \]

\[ D^2l(\theta) = \frac{\partial^2 l(\theta)}{w_{hj}^2} \left( \frac{1}{1 - \pi_{hj}} \right) + \frac{\partial^2 l(\theta)}{w_{hj}^2 \pi_{hj}^2} \]

\[ Dl_{\theta}(\theta) = Dl(\theta)(-\log(1 - \pi_{hj}))^{0.7} \]

\[ D^2l_{\theta}(\theta) = 0.7 Dl(\theta)(-\log(1 - \pi_{hj}))^{-0.3}(\frac{1}{1 - \pi_{hj}}) + D^2l(\theta)(-\log(1 - \pi_{hj}))^{0.7}. \]

At convergence of the above algorithm, setting \( \hat{D} = D^{(t)} \) and \( \hat{\beta} = \beta^{(t)} \), MLE of \( \sigma_0^2 \) is computed by

\[ \hat{\sigma}_0^2 = \frac{1}{n}(y - \hat{D}\hat{\beta})^T(y - \hat{D}\hat{\beta}). \quad (6.14) \]

Even this algorithm has dimensionality problem when the number of nodes is very large. In this case, we can use several iteration of a linear iterative method like Gauss-Seidel to solve the linear equation \( \frac{\partial l(\theta)}{\partial \beta} = 0 \) to overcome the curse of dimensionality.

### 6.2.2 Neural Network Prediction with Variance Components

Whether data we consider is from an observational study or an experimental design, it's possible to have more than one source of random variation according to the data (Gumpertz and Pantula, 1992). In this case, if each of the random effects can be considered as independent, identical distributed, especially normally distributed, linear effect, we can establish the following model structure:

\[ y_i = \beta_0 + \sum_{h=1}^{H} \frac{\beta_h}{1 + \exp(-w_h^T x_i)} + \sum_{j=1}^{u} z_{ij}^T b_j + \epsilon_i \quad (6.15) \]

where each \( b_j, j = 1, \ldots, u \) is a random vector independent of each other, independent of \( \epsilon = (\epsilon_1, \ldots, \epsilon_n)^T \), and distributed as \( N(0, \sigma_0^2 I_n) \), and the distribution of \( \epsilon \) is \( N(0, \sigma_0^2 I_n) \). Each \( Z_j \), each row of which is \( z_{ji} \), is a known \( n \times q_i \) matrix. Now the ML estimation of \( \theta = (\beta, w_1, \ldots, w_H, \sigma_0, \sigma_1, \ldots, \sigma_u)^T \) is based on the following loglikelihood

\[ l(\theta) = -0.5n \log 2\pi - 0.5 \log |V| - 0.5(y - D\beta)^T V^{-1}(y - D\beta), \quad (6.16) \]
where $V = \text{var}(y) = \sum_{t=1}^{u} \sigma_i^2 Z_iZ_i^T + \sigma_0^2 I_n$. Then $Dl(\theta)$ consist of

$$
\frac{\partial l(\theta)}{\partial \beta} = 0.5D^TV^{-1}y - 0.5D^TV^{-1}D\beta,
$$

$$
\frac{\partial l(\theta)}{\partial w_{hj}} = 0.5c_{hj}^TV^{-1}(y - D\beta), \quad \text{for } j = 0, \ldots, p, \ h = 1, \ldots, H,
$$

$$
\frac{\partial l(\theta)}{\partial \sigma_i^2} = -0.5\text{tr}((V^{-1}Z_iZ_i^T) + 0.5(y - D\beta)^TV^{-1}Z_iZ_i^TV^{-1}(y - D\beta), \quad (6.17)
$$

where $Z_0 = I_n$ and $i = 0, 1, \ldots, u$. And the elements of $D^2l(\theta)$ and $I_{obs}(\theta) = -E[D^2l(\theta)]$ are as follows:

$$
\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta} = -0.5D^TV^{-1}D,
$$

$$
\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \sigma_i} = -D^TV^{-1}Z_iZ_i^TV^{-1}(y - D\beta),
$$

$$
\frac{\partial^2 l(\theta)}{\partial \sigma_i \partial \sigma_j} = 0.5\text{tr}(V^{-1}Z_iZ_i^TV^{-1}Z_iZ_i^T) - (y - D\beta)^TV^{-1}Z_iZ_i^TV^{-1}Z_iZ_i^T(y - D\beta),\quad (6.17)
$$

$$
-E\left[\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta}\right] = -\frac{\partial^2 l(\theta)}{\partial \beta^T \partial \beta},
$$

$$
-E\left[\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial \beta_k}\right] = 0.5c_{hj}^TV^{-1}d_k,
$$

$$
-E\left[\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial \sigma_i}\right] = 0.5c_{hj}^TV^{-1}c_{kl}
$$

$$
-E\left[\frac{\partial^2 l(\theta)}{\partial w_{hj} \partial \sigma_i^2}\right] = 0,
$$

and $-E\left[\frac{\partial^2 l(\theta)}{\partial \sigma_i^2 \partial \sigma_j^2}\right] = 0.5\text{tr}(V^{-1}Z_iZ_i^TV^{-1}Z_iZ_i^T)$. 
It's obvious that first and second derivatives of loglikelihood except for \( w_h, h = 1, \ldots, H \) are exactly the same as those of the linear variance components model. This means that the useful results of the GCM algorithms discussed in Chapter 5 regarding the linear variance components model can be used without any modification for estimation of \((\beta, \sigma_0^2, \ldots, \sigma_u^2)\).

As for estimating \( w_h, h = 1, \ldots, H \), several algorithms are possible. In this section, we present two algorithms. One approach is to use the same reparameterization and the same linearization to the above loglikelihood as the one used in the case of the loglikelihood for the simple neural network prediction model.

The other is to perform a complete-data approximation to the loglikelihood assuming that \( b_i \)'s are missing data and to apply the same reparameterization and the same linearization to the conditional expectation of complete-data likelihood equation as he one mentioned before. That is, assuming complete data \((y, b_1, \ldots, b_u)\), we have

\[
\frac{\partial l_{\text{com}}(\theta)}{\partial w_{hj}} = \frac{0.5}{\sigma_0^2} c^T_{hj}(y - D\beta - \sum_{i=1}^{u} Z_i b_i), \quad \text{for } j = 0, \ldots, p, \quad h = 1, \ldots, H, \]

\[
\frac{\partial^2 l_{\text{com}}(\theta)}{\partial w_{hj} \partial w_{ki}} = \frac{-0.5}{\sigma_0^2} c^T_{hj} c_{ki} \quad \text{for } k \neq h.
\]

\[
= -\frac{0.5}{\sigma_0^2} c^T_{hj} c_{hi} + \frac{0.5}{\sigma_0^2} e^T_{hj}(y - D\beta - \sum_{i=1}^{u} Z_i b_i) \quad \text{for } k = h,
\]

where \( l_{\text{com}}(\theta) \) denotes the complete-data loglikelihood, and conditional expectation of \( b_i \)'s given observed data \( y \) are easily computed as

\[
E(b_i|y) = \sigma_i^2 Z_i^T V^{-1}(y - D\beta).
\]

Now one can solve \( \frac{\partial l(\theta)}{\partial w_{hj}} = 0 \) by applying the GCM algorithm using the same reparameterization and linearization to the conditional expectation of complete-data loglikelihood given observed data.

Thus we present two GCM algorithms (GCM-1 and GCM-2) below as follows:

Assuming \( \theta = (\beta, \sigma_0^2, \ldots, \sigma_u^2, \pi_{11}, \ldots, \pi_{1p}, \ldots, \pi_{H1}, \ldots, \pi_{Hp}) \), the \( t \)th iteration of GCM-1
Algorithm is

$$\theta^{(t+1)} = \theta^{(t)} - [W(\theta^{(t)})]^{-1} Dl_L(\theta^{(t)}),$$  \hspace{1cm} (6.18)

where $Dl_L(\theta) = Dl(\theta)K(\theta)$, $W(\theta)$ is block diagonal elements of $D^2l_L(\theta)$ for

$$\theta = (\beta, \sigma_0^2, \cdots, \sigma_u^2, \pi_{11}, \cdots, \pi_{1p}, \cdots, \pi_{H1}, \cdots, \pi_{Hp}),$$

and

$$K(\theta) = \text{blockdiag}([D^TV^{-1}D]^{-1}, \sigma_0^4, \cdots, \sigma_u^4, (-\log(1 - \pi_{11}))^{0.7}, \cdots, (-\log(1 - \pi_{1p}))^{0.7},$$

$$\cdots, (-\log(1 - \pi_{H1}))^{0.7}, \cdots, (-\log(1 - \pi_{Hp}))^{0.7}).$$

After partitioning $\theta$ into $\theta_1 = (\beta, \sigma_0^2, \cdots, \sigma_u^2)$ and $\theta_2 = (\pi_{11}, \cdots, \pi_{1p}, \cdots, \pi_{H1}, \cdots, \pi_{Hp})$, the $t$th iteration of GCM-2 has the following two steps:

$$\theta_1^{(t+1)} = \theta_1^{(t)} - [W_1(\theta_1^{(t)})]^{-1} Dl_{L1}(\theta_1^{(t)}),$$
$$\theta_2^{(t+1)} = \theta_2^{(t)} - [E[W_2(\theta_2^{(t)})|y, \theta^{(t)}]]^{-1} E[Dl_{L2(com)}(\theta_2^{(t)})|y, \theta^{(t)}],$$  \hspace{1cm} (6.19)

$$Dl_{L1}(\theta_1) = Dl(\theta_1)K_1(\theta_1),$$
$$W_1(\theta_1) \text{ is block diagonal elements of } D^2l_{L1}(\theta_1) \text{ for } (\beta, \sigma_0^2, \cdots, \sigma_u^2),$$
$$K_1(\theta_1) = \text{blockdiag}([D^TV^{-1}D]^{-1}, \sigma_0^4, \cdots, \sigma_u^4),$$
$$Dl_{L2(com)}(\theta_2) = Dl_{com}(\theta_2)K_2(\theta_2),$$
$$W_2(\theta_2) \text{ is block diagonal elements of } D^2l_{L2(com)}(\theta_2) \text{ for } (\pi_{11}, \cdots, \pi_{1p}, \cdots, \pi_{H1}, \cdots, \pi_{Hp}),$$

and

$$K_2(\theta_2) = \text{blockdiag}((-\log(1 - \pi_{11}))^{0.7}, \cdots, (-\log(1 - \pi_{1p}))^{0.7},$$

$$\cdots, (-\log(1 - \pi_{H1}))^{0.7}, \cdots, (-\log(1 - \pi_{Hp}))^{0.7}).$$

6.3 Discussion

GCM technique seeks to improve the convergence rate, and reduce computations while maintaining monotone convergence without using linear search in each iteration.
In this dissertation, improved algorithms were obtained for the repeated measure model, the t-model, the normal finite mixture, and the variance component model with/without missing data. But as observed in the contingency table example, GCM should be used prudently for improving algorithmic efficiency.

Most of statistical literature dealing with maximum likelihood estimation concentrate on two algorithmic approaches: Newton-type and EM-type. Although any EM-type algorithm is related to the missing data concept, it has been a competitor to Newton-type algorithms due to its flexibility and simplicity. The GCM algorithm is an approach to exploit the merits of the two algorithms by applying one of them to each of separate partitions of the parameters.

We have to emphasize that this dissertation is just the tip of the iceberg. There are many statistical models, especially generalized linear models, to which new algorithms for MLE need to be found and for which purpose GCM approach seems ideal. Also, GCM algorithm can play a main role in least squares problems and other estimation problems, e.g., M-estimation that involves optimization. Alternatively, a GCM algorithm may be derived for more complex models like generalized variance components model where random coefficients are not independent of each other or over-dispersed binomial mixed effects model. From the computing aspect, GCM approach might be more flexible to be adapted for high performance computing machines and for global maximization using, say, interval analysis, than other Newton-type algorithms.

As statisticians introduce new models in order to improve the analysis of their data, availability of more generally applicable algorithm for ML estimation becomes an important computational issue. In many situations we have observed that the standard Newton-type or EM-type algorithms may turn out to be not as efficient as desired. In these cases, the possibility of finding MLE's must be investigated. We believe that the flexibility and general applicability of GCM affords a new approach for obtaining new and more efficient computational algorithms.
APPENDIX A SWEEP OPERATOR FOR COMPUTING CONDITIONAL EXPECTATION AND VARIANCE-COVARIANCE MATRIX OF GAUSSIAN DISTRIBUTION

The sweep operator introduced by Beaton(1964) is indispensable for performing computations associated with the missing data problem of Gaussian models. In this Appendix, properties of the method will be reviewed with regards of to its use in computing conditional expectation and conditional variance. For basic applications of the sweep operator in regression computation, see Goodnight(1979).

Define \( SWEEP[k]V \) for \( V = ((v_{ij})) \), the \( n \times n \) variance-covariance matrix of \( y \) as follows:

Step 0 \( \text{pivot} = v_{kk} \)

Step 1 for \( i = 1 \) to \( n \); for \( j = 1 \) to \( n \)

\[
\text{if}(i \neq k \text{ and } j \neq k) \text{ then set } v_{ij} = v_{ij} - \frac{v_{ik}v_{kj}}{\text{pivot}}
\]

endfor; endfor;

Step 2 for \( i = 1 \) to \( n \)

set \( v_{ik} = v_{ik}/\text{pivot} \)

set \( v_{ki} = v_{ki}/\text{pivot} \)
endfor

• Step 3 set \( v_{kk} = -1/u_{kk} \)

The \((i, j)\) element of \( SWEEP[k]V \) is

\[
\text{cov}(y_i, y_j|y_k) \quad \text{for } i \neq k \text{ and } j \neq k,
\]

\[
\text{cov}(y_i, y_k)/\text{var}(y_k), \text{ slope parameter in simple regression of } y_i \text{ on } y_k \text{ for } i \neq k \text{ and } j = k,
\]

\[
\text{cov}(y_k, y_j)/\text{var}(y_k), \text{ slope parameter in simple regression of } y_j \text{ on } y_k \text{ for } i = k \text{ and } j \neq k.
\]

or \(-1/\text{var}(y_k)\) for \(i = j = k\).

Furthermore, it is easy to show by induction that the \((i, j)\) element of

\[
SWEEP[k_1]SWEEP[k_2] \cdots SWEEP[k_c]V
\]

for some positive integer \(c\) is

\[
\text{cov}(y_i, y_j|y_{k_1}, \cdots, y_{k_c}) \quad \text{for } i \notin k = \{k_1, \cdots, k_c\} \text{ and } j \notin k,
\]

the slope parameter corresponding to \(y_j\) in regression of \(y_i\) on \(y_{k_1}, \cdots, y_{k_c}\) for \(i \notin k\) and \(j \in k\),

the slope parameter corresponding to \(y_i\) in regression of \(y_j\) on \(y_{k_1}, \cdots, y_{k_c}\) for \(i \in k\) and \(j \notin k\),

or the element corresponding to \(y_i\) of \([-\text{var}(y_{k_1}, \cdots, y_{k_c})]\)^{-1}, inverse matrix of variance-covariance matrix of \(y_{k_1}, \cdots, y_{k_c}\) for \(i \in k\) and \(j \in k\).

For example, \(SWEEP[1]SWEEP[2] \cdots SWEEP[c]V\) changes

\[
V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \quad \text{into} \quad \begin{bmatrix} -V_{11}^{-1}V_{12} \\ V_{21}V_{11}^{-1}V_{22} - V_{21}V_{11}^{-1}V_{12} \end{bmatrix},
\]
where $V_{11}$ is $c \times c$ variance-covariance matrix of $y_1, \ldots, y_c$, $V_{12}$ is $c \times (n-c)$, $V_{21}$ is $(n-c) \times c$, and $V_{22}$ is $(n-c) \times (n-c)$ submatrix of $V$. Then, from the matrix resulting from the sweep operations above, it is trivial to compute

$$E[y_2|y_1] = E[y_2] + V_{21} V_{11}^{-1} [y_1 - E(y_1)],$$

$$Var[y_2|y_1] = V_{22} - V_{21} V_{11}^{-1} V_{12},$$

where $y_1 = (y_1, \ldots, y_c)^T$ and $y_2 = (y_{c+1}, \ldots, y_n)^T$. Although this example can be directly applied to 'monotone missing data' problem where missing data occurs only in bottom rows of $y$, the generalization of sweep operator for non-monotone missing data is also possible by using properties of the operator defined above. Applications of this case are discussed in Chapter 4.
APPENDIX B  DEFINITIONS RELATED TO THE SPACE-FILLING CONDITION (MENG AND RUBIN, 1993)

Other than the simple CM steps in definition of GCM algorithm, more complex CM steps can be performed by using space-filling condition which is presented in this appendix. Most of notations and descriptions in this appendix were copied from Meng and Rubin(1993).

Without loss of generality, GCM algorithm considered here is assumed to have only one cycle. Let

$$G = \{g_s(\theta); s = 1, \cdots, S\} \quad \text{(B.1)}$$

be a set of $S$ pre-selected(vector) functions of $\theta$.

**Definition 1.** For an approximation function $A(\theta|\theta)$, each of $S(\geq 1)$ CM-steps in the $t$th iteration finds $\theta^{(t+s/S)}$ such that

$$Q(\theta^{(t+s/S)}|\theta^{(t)}) \geq Q(\theta|\theta^{(t)}),$$

for all $\theta \in \Theta_s(\theta^{(t+(s-1)/S)}) \equiv \{\theta \in \Theta; g_s(\theta) = g_s(\theta^{(t+(s-1)/S)})\} \quad \text{(B.2)}$$

for $s = 1, \cdots, S$.

**Definition 2.** Let $T_s(\theta)(s = 1, \cdots, S)$ be the set of all feasible directions at $\theta \in \Theta$ with respect to the constraint space

$$\Theta_s(\theta) = \{\zeta \in \Theta; g_s(\zeta) = g_s(\theta)\}, \quad \text{(B.3)}$$

that is,

$$T_s(\theta) = \{\eta \in R^d; \exists \{\theta_n\} \subset \Theta_s(\theta) \text{ such that } \eta - \lim_{n \to \infty} \frac{\theta_n - \theta}{||\theta_n - \theta||}\}. \quad \text{(B.4)}$$
We say $G = \{g_s, s = 1, \cdots, S\}$ is space-filling at $\theta \in \Theta$ if

$$T(\theta) \equiv \text{closure}\{\sum_{s=1}^{S} a_s \eta_s; a_s \geq 0, \eta_s \in T_s(\theta)\} = \mathbb{R}^d$$ (B.5)

* We assume that $g_s(\theta)(s = 1, \cdots, S)$ is differentiable and the corresponding gradient, $\nabla g_s(\theta)$, is of full rank at $\theta \in \Theta_0$, the interior of $\Theta$.

* Under conditions above, one can show that (B.5) is equivalent to

$$J(\theta) \equiv \cap_{s=1}^{S} J_s(\theta) = 0,$$ (B.6)

where $J_s(\theta)$ is the column space of the gradient of $g_s(\theta)$, that is,

$$J_s(\theta) = \{\nabla g_s(\theta) \lambda; \lambda \in \mathbb{R}^{d_s}\}$$

and $d_s$ is the dimensionality of the vector function $g_s(\theta)$. And (B.6) is a direct consequence of the following identity,

$$J(\theta) = \{\xi; \xi^T \eta \leq 0 \text{ for all } \eta \in T(\theta)\},$$ (B.7)

which itself follows directly from the polar and bipolar theorems in the literature of constrained optimization, e.g. Fletcher(1980, Ch.9), Lay(1982, Ch.9). The advantage of expression (B.6) over (B.5) is that it can be verified directly in many applications.
BIBLIOGRAPHY


Lansky, D., and Casella, G. (1990?) “Improving the EM algorithm”, Cornell University Biometrics Unit BU-1081-M.


