On the Numerical Evaluation of Optimal Variance Components Estimators in Crossed Classification Credibility

Xiaotong Wang

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

April, 2005

Your file Votre référence
ISBN: 0-494-04457-8
Our file Notre référence
ISBN: 0-494-04457-8

Abstract

On the Numerical Evaluation of Optimal Variance Components Estimators in Crossed Classification Credibility

Xiaotong Wang

Credibility theory is a set of quantitative tools which allows insurers to perform prospective experience rating on a risk or group of risks in a heterogeneous portfolio. Credibility theory promotes a mechanism for the implementation of risk management strategies by differentiating between good and poor risks. It is used in the setting of rates for classification systems. Furthermore, application of credibility theory would also help to improve the ability of insurance companies to modify their price structure with the changing economic environment.

Dannenburg [7] introduced a crossed classification credibility (CCC) model applicable to contracts that could be affected by many risk factors, that can not be modeled as nested or hierarchical relationships. In practice, the main problem in crossed classification credibility is the estimation of structure parameters. In a two-way CCC model, these parameters are the collective mean m, the time variance s^2, and the variance components $b^{(1)}, b^{(2)}$, and $b^{(12)}$.

Dannenburg has already proposed estimators of these structure parameters. However, they have no known optimality property except unbiasedness. Goulet [24] proposed minimum variance unbiased (optimal) estimators for the mean and variance components.

This thesis focuses on the implementation of the variance components estimators calculation. Analysis of algorithms and numerical evaluation procedures are studied to achieve computing efficiency. Through a simulation study, optimality of these estimators is then assessed in comparison to estimators proposed by Dannenburg [7].

Contents

List of Figures viii

List of Tables x

1 Introduction to Credibility Theory 1

1.1 What is Credibility Theory 1

1.2 Previous Work on Credibility Theory 4

1.3 Problems 6

1.4 Thesis Outline 9

2 Two-Way CCC Model and Optimal Parameter Estimation 11

2.1 A Two-Way Crossed Classification Credibility Model 12

2.2 Estimation of the Structure Parameters in the CCC Model . 15

2.3 Optimal Estimation of the Variance Components 17

3 Simulation Study 22

3.1 Previous Work and Existing Problems 22

3.2 Structure of Matrix C and Attributes 23

3.3 Simulation of Variance Components 24

3.3.1 Structure of the Simulation on Variance Components . 25

3.3.2 Model of $b^{(12)}$ 26

3.3.3 Model of $b^{(1)}$ 27
3.3.4 Model of $b^{(2)}$ 27
3.4 Properties of Fixed-Point Algorithm 27
3.5 Problem Analysis and Resolving Method 36
3.5.1 Solving a System of Linear Equations 36
3.5.2 Cholesky Decomposition Method 38
3.5.3 Application of Standard Subroutine Packages 40
3.5.4 Algorithm of Lapack Cholesky Routines 41
3.5.5 Integration with Matlab C Math Library 44
3.5.6 Application of Back Slash 45
3.5.7 Memory Allocation 48
3.5.8 Application of R Math Library 48
3.6 Accuracy of Solution 49
3.7 Speed Test 49
3.8 Integration with R 50

4 Software Implementation 57
4.1 Problem Description 57
4.2 Input Parameters 58
4.3 Output Requirement 59
4.4 Simulation of Portfolios 59
4.5 Flow Chart of the Simulation Study on Variance Components Estimators 60
4.6 Implementation Strategies 61
4.6.1 Application of Multiple Level Pointer 61
4.6.2 Transformation of Data Structures 69
4.6.3 Elimination of Redundant Calculations 69

5 Results of the Simulation Study 72

5.1 Results and Comparison with Dannenburg's Estimators . . . 72

5.1.1 Simulation Results for $I = J = 4$ 73

5.1.2 Simulation Results for $I = J = 6$ 76

5.1.3 Simulation Results for $I = J = 8$ 79

5.1.4 Simulation Results for $I = J = 10$ 81

5.2 Graphical Comparison of Standard Deviations 81

5.3 Conclusions 84

Conclusion and Future Work 89

Bibliography 94

List of Figures

3.1 Model of Structural Parameter $b^{(12)}$ 28

3.2 Model of Structural Parameter $b^{(1)}$ 29

3.3 Model of Structural Parameter $b^{(2)}$ 30

4.1 Simulation Flow Chart 62

5.1 Graphs of Small Values Parameters_ $b^{(1)}$ 82

5.2 Graphs of Small Values Parameters_ $b^{(2)}$ 83

5.3 Graphs of Small Values Parameters_ $b^{(12)}$ 83

5.4 Graphs of Moderate Values Parameters_ $b^{(1)}$ 85

5.5 Graphs of Moderate Values Parameters_ $b^{(2)}$ 85

5.6 Graphs of Moderate Values Parameters_ $b^{(12)}$ 86

5.7 Graphs of Large Values Parameters_ $b^{(1)}$ 86

5.8 Graphs of Large Values Parameters_ $b^{(2)}$ 87

5.9 Graphs of Large Values Parameters_ $b^{(12)}$ 87

List of Tables

5.1 Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 4$, 1000 *runs.* 73

5.2 Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 4$, 1000 *runs.* 73

5.3 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 4$, 1000 *runs.* 74

5.4 Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 4$, 5000 *runs.* 74

5.5 Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 4$, 5000 *runs.* 74

5.6 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 4$, 5000 *runs.* 75

5.7 Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 6$, 1000 *runs.* 76

5.8 Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 6$, 1000 *runs.* 76

5.9 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 6$, 1000 *runs.* 77

5.10 Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 6$, 5000 *runs.* 77

5.11 Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 6$, 5000 *runs.* 77

5.12 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 6$, 5000 *runs.* 78

5.13 Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 8$, 1000 *runs.* 79

5.14 Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 8$, 1000 *runs.* 79

5.15 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 8$, 1000 *runs.* 80

5.16 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.* 81

5.17 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.* 81

5.18 Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.* 82

Chapter 1

Introduction to Credibility Theory

1.1 What is Credibility Theory

Credibility theory is a set of quantitative tools which allows an insurer to perform prospective experience rating on a risk or group of risks [29]. Therefore, credibility theory is an effective solution for determining insurance premiums for contracts in a heterogeneous portfolio.

In order to establish a base premium, the insurer's first concern is to ascertain that the premium is sufficiently large to fulfill its obligations. When this is established, the insurer can then distribute the premium fairly among its insureds.

According to the classical manual rating system, the premium is distributed to reflect the expected experience of the entire rating class and implicitly assumes that the risks are homogeneous. However, no classification system is perfect, and there always remains some heterogeneity in the risk levels after all the underwriting criteria

are accounted for. Consequently, some policyholders will be better risks than that assumed in the underlying manual rate. Of course, a policyholder who takes steps to reduce loss should be rewarded. Conversely, a rate increase should be applied to a poor risk.

To resolve the risk heterogeneity problem, an experience rating system is used to assign to each individual risk an appropriate premium rate. The correct premium for any period depends exclusively on the (unknown) claims distribution of the individual risk for the same period [5]. However, these systems have a somewhat limited scope in insurance because they require the accumulation of a significant volume of experience. For example, experience rating does not apply to traditional individual life insurance (we only die once) and homeowners insurance. Therefore, experience rating is especially suited to certain types of insurance, such as workers compensation, automobile insurance and reinsurance.

The question arises: should insurance companies charge the insured the collective net premium, or should they construct a tariff system based solely on the individual claims experience? There are arguments against using either of these extremes. Charging the collective premium m seems unfair for most risks. Such a tariff system tends to chase away "good" risks and attract "bad" risks. On the other hand, charging the pure risk premium is against the aim of insurance, since the risk is not spread over a group of similar policies, and every insured pays for his own claims. As a compromise

between hypothesis and observation, one might charge:

$$P = zX + (1 - z)m \,, \tag{1.1}$$

where P is the individual premium; X is the mean from the individual experience; m is the collective mean; $0 \leq z \leq 1$ is the credibility factor assigned to the observation and $1 - z$ refers to the complement of credibility. If the body of observed data is large and not likely to vary much from one period to another, then z will be closer to 1. In this case, the policyholder's own experience has more credibility. On the other hand, if the observation consists of limited data, then z will be closer to zero and more weight will be given to other information. Credibility theory provides the tools to calculate z as well as the global premium m.

Credibility theory is used to set the rates for insurance classification systems. Outside the usual domain of application mentioned above, credibility theory has also been developed and successfully applied in cases, such as valuation methods of pension fund assets, which involve a blend of initial cost and current market value, as well as dividend interest and expense rates, the latter of which are treated as a blend of initial assumption and recent results [28].

Generally, credibility theory tells us that it is optimal to give only partial weight to past experience and give the remaining weight to an estimator produced from other information.

1.2 Previous Work on Credibility Theory

Credibility theory is based on the basic economic theory of risk and statistical estimation. Because it concerns itself models with the adaptive response of insurance price systems to dynamic environments, this theory covers most fields of actuarial science. Credibility theory is split into two branches: limited fluctuation credibility and greatest accuracy credibility.

Limited fluctuation credibility theory was developed around the time of the First World War in connection with premium adjustment systems in workmen's compensation insurance. The first concept (full credibility) appeared in a paper by A.H. Mowbray [32] in the first volume of the Proceedings of the Casualty Actuarial Society. In 1918, Whitney posited that the objective of credibility theory is the calculation of the balance between established class experience and risk experience. This inaugurated the development of partial credibility. Whitney's method was the first step towards greatest accuracy credibility, based on the homogeneity of the portfolio.

After three decades dominated by limited fluctuation studies, the post World War II era saw the revival of Whitneys theory of random effect. Combined with suitable elements of statistical decision theory, Whitney's ideas rapidly developed into a huge

body of models and methods – the greatest accuracy approach – which originated from a series of papers by Arthur L. Bailey [1, 2]. The experience rating problem was now seen as a matter of estimating the random variable $\mu(\Theta)$ with some function $g(\cdot)$ of the individual data $X_1, ..., X_T$, the objective being to minimize the mean squared error $E[(\mu(\theta) - g(X_1, X_2, ..., X_T))^2]$. This is what is now referred to as the Bayesian premium. Drawing upon this theorem, Bühlmann [3] became interested in the idea of approximating a posterior mean by a linear function of the prior mean and the mean of the claims data.

In the 1970's, with the rapid development of credibility theory, Bühlmann and Straub [4] generalized Bühlmann's classical model by assigning weights to the observations and by introducing estimators to the structure parameters. In fact, however, the greatest accuracy resolution to the credibility problem had essentially already been set out two decades earlier by Bailey [1, 2]. Like many other scientific works ahead of their time, however, they did not receive wide recognition. This theory came prior to and, therefore, could not benefit from modern statistical decision theory. In the following years, the Bühlmann–Straub model, the hierarchical model of Jewell [27] and the linear regression model of Hachemeister [26] were generated.

If the 1970's mark the period when credibility research focused on model generation, it was during the 1980's that the estimation of structure parameters became the focus. This decade saw, for example, the formulation of optimal parameter estimation by

De Vylder and Goovaerts [14, 12], robust parameter estimation by Künsch [30] and Gisler, Reinhard [20]. Important papers produced during the 1980's include De Vylder [9, 10, 11], Norberg [35], Gisler [19] and Dubey, Gisler [17].

A more recent innovation in credibility theory is the variance components model introduced by Dannenburg in 1995 [7] to describe his crossed classification credibility model.

1.3 Problems

In many cases, contracts in an insurance porfolio can not be grouped in a hierarchical model. If risks in an automobile portfolio are categorized by sex and age of insureds, neither of these two factors can be listed at a prior level. These two risk factors are not nested in general. Certain risk characteristics are specific for sex or age, but others for interaction between them. In order to cope with this kind of data structrure, a so–called crossed classification credibility model was introduced by Dannenburg in 1995 [7].

In the crossed classification credibility model, Dannenburg uses the variance components model to present the insured's claim ratio. The risk can be written as a sum of uncorrelated random variables, each representing the contribution of a risk factor or of an interaction between risk factors to the variance of the risk. For example, the

claim ratio in a two-way crossed classification model can be written as follows:

$$X_{ijt} = m + \Xi_i^{(1)} + \Xi_j^{(2)} + \Xi_{ij}^{(12)} + \Xi_{ijt}^{(123)}\,, \tag{1.2}$$

where $\Xi_i^{(1)}, \Xi_j^{(2)}, \Xi_{ij}^{(12)}$, and $\Xi_{ijt}^{(123)}$ are the so–called random effects and assumed to be mutually independent random variables; $\Xi_i^{(1)}, \Xi_j^{(2)}$ represent the variability due to risk factor 1 and 2 respectively; $\Xi_{ij}^{(12)}$ represents the variability due to the interaction between the two risk factors; $\Xi_{ijt}^{(123)}$ represents the variability in the insured's claims over time. Expected values of all the random variables are zero. Variances of the random variables are as follows:

$$\mathrm{Var}[\Xi_i^{(1)}] = b^{(1)},$$

$$\mathrm{Var}[\Xi_j^{(2)}] = b^{(2)},$$

$$\mathrm{Var}[\Xi_{ij}^{(12)}] = b^{(12)},$$

$$\mathrm{Var}[\Xi_{ijt}^{(123)}] = \frac{s^2}{w_{ijt}}.$$

Provided that the sum of the four variances $b^{(1)}, b^{(2)}, b^{(12)}$ and s^2/w_{ijt} makes up the total variance of the contract's experience, they are called the variance components. Indeed, independent of the random effects

$$\mathrm{Var}[X_{ijt}] = b^{(1)} + b^{(2)} + b^{(12)} + \frac{s^2}{w_{ijt}}\,. \tag{1.3}$$

In practice, the main problem in crossed classification credibility is the estimation of the structure parameters. In a two-way crossed classification credibility model, these

parameters are the collective mean m, the time variance s^2 and the variance components $b^{(1)}, b^{(2)}$ and $b^{(12)}$. Estimators of these structure parameters can be found in Dannenburg [7]. However, they have no known optimality property other than unbiasedness. Goulet [24] uses minimum variance to optimize estimators of the structure parameters in a two-way CCC model. Numerical tests have also been proposed in the latter paper.

However, calculation of the optimal variance components estimators requires considerable computation. The matrix present in all estimators (see section 2.2) grows very rapidly when the number of categories for every risk factor increases. It grows in the order of $(IJ)^2$. Another problem with optimal variance components estimators is application of the fixed point estimation method. Computation of such a fixed point is an iterative process repeated until successive values of a calculation are close to each other. This process requires a great deal of computing resources. Moreover, each iteration involves the construction of a large matrix and its inversion.

In order to prove the optimal theory, Goulet [24] implemented Dannenburg's estimators, optimal estimators and generalized ANOVA in APL. However, due to limitation of the programming language, only simulations of few repetitions (300) and small portfolios were made in optimal estimators. These simulations are thus too small to generate significant results. Languages S, S plus and R have also been applied to implement the simulation. However, these programs have proven ineffective for large

scale calculations as well.

A simulation using C/C++ has been attempted. However, this work led to a dead end because the calculations proved too complicated, and appropriate methodologies and libraries could not be found. Therefore, the task taken up in this study of implementing this simulation is very important as a mean to demonstrate the optimal property of variance components estimators.

1.4 Thesis Outline

The objective of this thesis is to implement the calculation of variance components estimators in C, to look for an effective optimal algorithm and to develop tools in order to increase computing speed, and to derive data for further studies in the CCC model.

This study will begin by introducing the two way crossed classification credibility model that will be the basis of simulation in variance components estimators. Estimators of structure parameters proposed by Dannenburg will then been presented. Finally, Goulet's optimal estimation of the variance components [24] and optimal properties will be introduced by comparing Dannenburg's and optimal theories.

Chapter 3 analyzes existing problems, and provides an effective solving method. By

integrating successfully C language and Matlab C library, the simulation allocates and frees memory efficiently, so that the program will not only continue running in spite of a large number of iterative calculations, but will also improve the running speed significantly. The implementation procedure will be introduced in this chapter. Some algorithms analysis, and structural description of the simulation will also be presented. Furthermore, the process of integrating the application with R will be discussed in this chapter.

In Chapter 4, a flow chart and implementation strategies of the simulation study will be introduced. During the process of implementation, all kinds of difficulties arose. This chapter highlights the resolving strategies employed to overcome main problems.

The results of the simulation study for different portfolios will be given in Chapter 5. Graphs for small, moderate, and large values of parameters will be presented in order to display the solutions and trends clearly. By comparing Dannenburg's estimation with Goulet's optimal estimation, the optimal property of Goulet's optimal estimators in variance components will be proven.

Finally, the last chapter will present the study's conclusions, and suggestions for future research.

Chapter 2

Two-Way CCC Model and Optimal Parameter Estimation

Credibility theory is an experience rating technique used to determine insurance premiums for contracts in a more or less heterogeneous portfolio.

Bühlmann's classical model introduced in 1967 is the first and simplest credibility model. It is simple because it requires that the past claims experience of a policyholder comprise independent and identically distributed components with respect to each past year. However, Bühlmann's model does not allow for variations in exposure or size. In 1970, the Bühlmann-Straub model [4] was generated from the classical credibility model by adding weights to observations. Since then, the model has been widely used in reinsurance or auto insurance. Indeed, it became the cornerstone of greatest accuracy credibility theory. After that, many models were developed based on Bühlmann's model, including Hachemeister's [26] regression model, Jewell's [27] linear hierarchical model, Norberg's [34] nested classification model, etc.

In 1995, the crossed classification credibility model was introduced. Consider automobile insurance, where an insurance contract is affected by many risk factors. These risk factors can not be modeled as nested or hierarchical relationships. One risk factor has no prior level over the others. Furthermore, risk factors interact with each other. For example, the risks of an automobile portfolio can be classified by the sex and by the age of the insured: a young man may share driving characteristics with a young woman, and a young woman with an older woman. In order to deal with this kind of classification structure, Dannenburg [7] introduced the variance components approach into credibility theory.

Variance components models come from the statistical theory of linear models. In a variance components model, a random variable is written as a sum of uncorrelated random variables, each representing the contribution of a risk factor or of an interaction between risk factors to the variance of this variable [7]. Therefore, every component of the structure is called a variance component. See Searle et al. [37] for an excellent treatment of the subject.

2.1 A Two-Way Crossed Classification Credibility Model

We introduce in this section the two way CCC model that will be used in our simulation study.

Credibility theory lays emphasis on predicting future claims. Assuming a two-way crossed classification structure, Dannenburg wrote the claim amount X_{ijt} as:

$$X_{ijt} = m + \Xi_i^{(1)} + \Xi_j^{(2)} + \Xi_{ij}^{(12)} + \Xi_{ijt}^{(123)} .$$

Here, $\Xi_i^{(1)}$ represents the first risk factor; $\Xi_j^{(2)}$ represents the second risk factor and $\Xi_{ij}^{(12)}$ is the risk characteristic specific to the interaction between the two risk factors. Random variable $\Xi_{ijt}^{(123)}$ represents time variability. The expected value of every random variable Ξ is zero and the variances are:

$$\mathrm{Var}[\Xi_i^{(1)}] = b^{(1)},$$

$$\mathrm{Var}[\Xi_j^{(2)}] = b^{(2)},$$

$$\mathrm{Var}[\Xi_{ij}^{(12)}] = b^{(12)},$$

$$\mathrm{Var}[\Xi_{ijt}^{(123)}] = \frac{s^2}{w_{ijt}}.$$

Under the proceeding assumptions, the credibility estimator for $X_{ij,T_{ij}+1}$ in the unbalanced two-way crossed classification model is expressed as:

$$\hat{X}_{ij,T_{ij}+1} = m + z_{ij}^{(12)}(X_{ijw} - m) + (1 - z_{ij}^{(12)})(\hat{\Xi}_i^{(1)} + \hat{\Xi}_j^{(2)}) . \tag{2.1}$$

$\hat{\Xi}_i^{(1)}$ and $\hat{\Xi}_j^{(2)}$ are called credibility estimators for $\Xi_i^{(1)}$ and $\Xi_j^{(2)}$ respectively. They are obtained from the following equations:

$$\hat{\Xi}_i^{(1)} = z_i^{(1)}(X_{izw} - m) - z_i^{(1)} \sum_{l=1}^{J} \frac{z_{il}^{(12)}}{z_{i\Sigma}^{(12)}} \hat{\Xi}_l^{(2)} ;$$

$$\hat{\Xi}_j^{(2)} = z_j^{(2)}(X_{zjw} - m) - z_j^{(2)} \sum_{k=1}^{I} \frac{z_{kj}^{(12)}}{z_{\Sigma j}^{(12)}} \hat{\Xi}_k^{(1)} .$$

These equations are based on the credibility factors

$$z_{ij}^{(12)} = \frac{b^{(12)}}{b^{(12)} + s^2/w_{ij\Sigma}}$$
$$z_i^{(1)} = \frac{b^{(1)}}{b^{(1)} + b^{(12)}/z_{i\Sigma}^{(12)}}$$
$$z_j^{(2)} = \frac{b^{(2)}}{b^{(2)} + b^{(12)}/z_{\Sigma j}^{(12)}}$$

and the weighted averages

$$X_{ijw} = \sum_{t=1}^{T_{ij}} \frac{w_{ijt}}{w_{ij\Sigma}} X_{ijt}$$
$$X_{izw} = \sum_{l=1}^{J} \frac{z_{il}^{(12)}}{z_{i\Sigma}^{(12)}} X_{ilw}$$
$$X_{zjw} = \sum_{k=1}^{I} \frac{z_{kj}^{(12)}}{z_{\Sigma j}^{(12)}} X_{kjw} \ ,$$

where

$$z_{i\Sigma}^{(12)} = \sum_{j=1}^{J} z_{ij}^{(12)}$$
$$z_{\Sigma j}^{(12)} = \sum_{i=1}^{I} z_{ij}^{(12)}$$
$$w_{ij\Sigma} = \sum_{t=1}^{T_{ij}} w_{ijt} \ .$$

The proof of this result can be found in [7]. In practical application, computation of these credibility estimators requires estimations of the structure parameters. In a two-way CCC model, these parameters are the collective mean m, the time variance s^2 and the variance components $b^{(1)}, b^{(2)}$ and $b^{(12)}$.

2.2 Estimation of the Structure Parameters in the CCC Model

Estimation of structure parameters is a crucial step in the application of a credibility model. In this section, structure parameters estimators introduced by Dannenburg will be presented.

An unbiased estimator of m is the weighted mean of the observations:

$$\hat{m} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{w_{ij\Sigma}}{w_{\Sigma\Sigma\Sigma}} X_{ijw} \ . \tag{2.2}$$

An unbiased estimator for s^2 is as follows:

$$\hat{s}^2 = \frac{1}{\sum_{i=1}^{I}\sum_{j=1}^{J}(T_{ij}-1)} \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{t=1}^{T_{ij}} w_{ijt}(X_{ijt}-X_{ijw})^2 \ . \tag{2.3}$$

Dannenburg proposed estimators of parameters $b^{(1)}$, $b^{(2)}$ and $b^{(12)}$ based on the following expectations:

$$E\Big[\sum_{i=1}^{I} \frac{g_i^{(1)}}{g_\Sigma^{(1)}} \Big[\sum_{j=1}^{J} \frac{w_{ij\Sigma}}{w_{i\Sigma\Sigma}}(X_{ijw}-X_{iww})^2 - \hat{s}^2(J-1)/w_{i\Sigma\Sigma}\Big]\Big]$$
$$= (b^{(2)}+b^{(12)})\Big[1-\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{g_i^{(1)}}{g_\Sigma^{(1)}}\Big[\frac{w_{ij\Sigma}}{w_{i\Sigma\Sigma}}\Big]^2\Big] \ ,$$

$$E\Big[\sum_{j=1}^{J} \frac{g_j^{(2)}}{g_\Sigma^{(2)}} \Big[\sum_{i=1}^{I} \frac{w_{ij\Sigma}}{w_{\Sigma j\Sigma}}(X_{ijw}-X_{wjw})^2 - \hat{s}^2(I-1)/w_{\Sigma j\Sigma}\Big]\Big]$$
$$= (b^{(1)}+b^{(12)})\Big[1-\sum_{j=1}^{J}\sum_{i=1}^{I}\frac{g_j^{(2)}}{g_\Sigma^{(2)}}\Big[\frac{w_{ij\Sigma}}{w_{\Sigma j\Sigma}}\Big]^2\Big] \ ,$$

and

$$E\Big[\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{w_{ij\Sigma}}{w_{\Sigma\Sigma\Sigma}}(X_{ijw}-X_{www})^2-\hat{s}^2(IJ-1)/w_{\Sigma\Sigma\Sigma}\Big]=$$

$$b^{(1)}\Big[1-\sum_{i=1}^{I}\Big[\frac{w_{i\Sigma\Sigma}}{w_{\Sigma\Sigma\Sigma}}\Big]^2\Big]+b^{(2)}\Big[1-\sum_{j=1}^{J}\Big[\frac{w_{\Sigma j\Sigma}}{w_{\Sigma\Sigma\Sigma}}\Big]^2\Big]+b^{(12)}\Big[1-\sum_{i=1}^{I}\sum_{j=1}^{J}\Big[\frac{w_{ij\Sigma}}{w_{\Sigma\Sigma\Sigma}}\Big]^2\Big].$$

Defining

$$\begin{aligned}
X_1 &= \frac{1}{I}\sum_{i=1}^{I}(\sum_{j=1}^{J}\frac{w_{ij\Sigma}}{w_{i\Sigma\Sigma}}(X_{ijw}-X_{iww})^2-\frac{(J-1)}{w_{i\Sigma\Sigma}}\hat{s}^2)\\
X_2 &= \frac{1}{J}\sum_{j=1}^{J}(\sum_{i=1}^{I}\frac{w_{ij\Sigma}}{w_{\Sigma j\Sigma}}(X_{ijw}-X_{wjw})^2-\frac{(I-1)}{w_{\Sigma j\Sigma}}\hat{s}^2)\\
X_3 &= \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{w_{ij\Sigma}}{w_{\Sigma\Sigma\Sigma}}(X_{ijw}-X_{www})^2-\frac{(IJ-1)}{w_{\Sigma\Sigma\Sigma}}\hat{s}^2
\end{aligned}$$

and

$$\begin{aligned}
a_1 &= 1-\frac{1}{I}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(\frac{w_{ij\Sigma}}{w_{i\Sigma\Sigma}}\right)^2\\
a_2 &= 1-\frac{1}{J}\sum_{j=1}^{J}\sum_{i=1}^{I}\left(\frac{w_{ij\Sigma}}{w_{\Sigma j\Sigma}}\right)^2\\
a_3 &= 1-\sum_{i=1}^{I}\left(\frac{w_{i\Sigma\Sigma}}{w_{\Sigma\Sigma\Sigma}}\right)^2\\
a_4 &= 1-\sum_{j=1}^{J}\left(\frac{w_{\Sigma j\Sigma}}{w_{\Sigma\Sigma\Sigma}}\right)^2\\
a_5 &= 1-\sum_{i=1}^{I}\sum_{j=1}^{J}\left(\frac{w_{ij\Sigma}}{w_{\Sigma\Sigma\Sigma}}\right)^2,
\end{aligned}$$

then estimators for $b^{(1)}, b^{(2)}$ and $b^{(12)}$ are obtained by solving the following linear system of equations:

$$\begin{pmatrix} X_1\\ X_2\\ X_3\end{pmatrix}=\begin{pmatrix} 0 & a_1 & a_1\\ a_2 & 0 & a_2\\ a_3 & a_4 & a_5\end{pmatrix}\begin{pmatrix} b^{(1)}\\ b^{(2)}\\ b^{(12)}\end{pmatrix}.$$

These estimators have no known optimality properties other than unbiasedness. Furthermore, these variance components estimators have a strong tendency to yield negative estimates.

2.3 Optimal Estimation of the Variance Components

The most frequently desirable properties for estimators are unbiasedness and minimum variance. Since Dannenburg's estimators already have the unbiasedness property, minimum variance becomes the first consideration in the optimization of the estimation.

In 1992, De Vylder and Goovaerts [13] obtained the minimum variance estimators of the structure parameters in the Bühlmann credibility model under an assumption of zero excess for some of the involved random variables. However, a supplementary restrictive assumption regarding the third central moment is needed in order to prove optimality of the estimators. In 1998, Goulet [23] used the variance components approach to circumvent the supplementary condition.

Goulet's optimal estimation of the variance components operates under the assumption that all random effects have no excess. The coefficient of excess of any non-

degenerated random variable with a finite fourth-order moment is defined as

$$\frac{E[(X-EX)^4]}{E^2[(X-EX)^2]} - 3 \, .$$

We say that X has zero excess if its coefficient of excess equals zero. Normally distributed random variables have zero excess.

The following covariance relations are needed in the sequel. Proof of the theorem can be found in Goulet [3].

Theorem 1. In a two-way CCC model,

$$\begin{aligned}
\mathrm{Cov}(X_{ijw}, X_{klw}) &= \sigma^{(12)}_{ijkl} = \delta_{ik} b^{(1)} + \delta_{jl} b^{(2)} + \delta_{ij,kl} \frac{b^{(12)}}{z_{ij}^{(12)}} \, , \\
\mathrm{Cov}(X_{izw}, X_{kzw}) &= \sigma^{(1)}_{ik} = \delta_{ik} \frac{b^{(1)}}{z_i^{(1)}} + b^{(2)} S^{(1)}_{ik} \, , \\
\mathrm{Cov}(X_{zjw}, X_{zlw}) &= \sigma^{(2)}_{jl} = \delta_{jl} \frac{b^{(2)}}{z_j^{(2)}} + b^{(1)} S^{(2)}_{jl} \, ,
\end{aligned}$$

with

$$\begin{aligned}
S^{(1)}_{ik} &= \sum_{j=1}^{J} \frac{z^{(12)}_{ij}}{z^{(12)}_{i\Sigma}} \frac{z^{(12)}_{kj}}{z^{(12)}_{k\Sigma}} \, , \\
S^{(2)}_{jl} &= \sum_{i=1}^{I} \frac{z^{(12)}_{ij}}{z^{(12)}_{\Sigma j}} \frac{z^{(12)}_{il}}{z^{(12)}_{\Sigma l}} \, .
\end{aligned}$$

The Kronecker symbol δ_{ij} equals 1 when $i = j$ and 0 otherwise.

The following theorems present optimal estimators of parameters $b^{(12)}, b^{(1)}, b^{(2)}$. Proofs are found in Goulet [24].

Theorem 2. Estimation of $b^{(12)}$

If all random variables Ξ have zero excess in a two-way CCC model, then the minimum variance estimator of structure parameter $b^{(12)}$ among all estimators of the form

$$\sum_{ij} \sum_{kl>ij} \alpha_{ijkl}(X_{ijw} - X_{klw})^2 \tag{2.4}$$

with expected value $b^{(12)}$ is

$$b_*^{(12)} = X' \mathrm{diag}(A) X \ , \tag{2.5}$$

where

$$A = \frac{b^{(12)}}{B' C^{-1} B} C^{-1} B \ , \tag{2.6}$$

and

$$B = [\sigma_{ijij}^{(12)} - 2\sigma_{ijkl}^{(12)} + \sigma_{klkl}^{(12)}]_{\frac{IJ(IJ-1)}{2} \times 1} \ , \tag{2.7}$$

$$C = [(\sigma_{ijgh}^{(12)} - \sigma_{ijpq}^{(12)} - \sigma_{klgh}^{(12)} + \sigma_{klpq}^{(12)})^2]_{\frac{IJ(IJ-1)}{2} \times \frac{IJ(IJ-1)}{2}} \ , \tag{2.8}$$

$$X = [X_{ijw} - X_{klw}]_{\frac{IJ(IJ-1)}{2} \times 1} \ . \tag{2.9}$$

In the expression for C, subscripts i, j, k and l refer to the lines of the matrix and subscripts g, h, p and q, the columns. Notation $kl > ij$ is defined as follows, for $k \geq i = 1, ..., I$ and $j = 1, ..., J$:

$$kl > ij \Leftrightarrow \begin{cases} l = 1, ..., J, & if k > i \ , \\ l > j, & if k = i \ . \end{cases}$$

Theorem 3. Estimation of $b^{(1)}$

If all random variables Ξ have no excess in a two-way CCC model, then the minimum variance estimator of structure parameter $b^{(1)}$ among all estimators of the form

$$\sum_i \sum_{k>i} \alpha_{ik}(X_{izw} - X_{kzw})^2 \tag{2.10}$$

with expected value $b^{(1)}$ is

$$b_*^{(1)} = X' \mathrm{diag}(A) X \ , \tag{2.11}$$

where

$$A = \frac{b^{(1)}}{B' C^{-1} B} C^{-1} B \ , \tag{2.12}$$

and

$$\begin{aligned} B &= [\sigma_{ii}^{(1)} - 2\sigma_{ik}^{(1)} + \sigma_{kk}^{(1)}]_{\frac{I(I-1)}{2} \times 1} \ , \\ C &= [(\sigma_{ig}^{(1)} - \sigma_{ip}^{(1)} - \sigma_{kg}^{(1)} + \sigma_{kp}^{(1)})^2]_{\frac{I(I-1)}{2} \times \frac{I(I-1)}{2}} \ , \\ X &= [X_{izw} - X_{kzw}]_{\frac{I(I-1)}{2} \times 1} \ . \end{aligned}$$

In the expression for C, subscripts i and k refer to the lines of the matrix and subscripts g and p, the columns.

Theorem 4. Estimation of $b^{(2)}$

If all random variables Ξ have no excess in a two-way CCC model, then the minimum variance estimator of the structure parameter $b^{(2)}$ among all estimators of the form

$$\sum_j \sum_{l>j} \alpha_{jl}(X_{zjw} - X_{zlw})^2 \tag{2.13}$$

with expected value $b^{(2)}$ is

$$b_*^{(2)} = X^{'}\text{diag}(A)X \ , \tag{2.14}$$

where

$$A = \frac{b^{(2)}}{B'C^{-1}B}C^{-1}B \ , \tag{2.15}$$

and

$$\begin{aligned} B &= [\sigma_{jj}^{(2)} - 2\sigma_{jl}^{(2)} + \sigma_{ll}^{(2)}]_{\frac{J(J-1)}{2}\times 1} \ , \\ C &= [(\sigma_{jh}^{(2)} - \sigma_{jq}^{(2)} - \sigma_{lh}^{(2)} + \sigma_{lq}^{(2)})^2]_{\frac{J(J-1)}{2}\times\frac{J(J-1)}{2}} \ , \\ X &= [X_{zjw} - X_{zlw}]_{\frac{J(J-1)}{2}\times 1} \ . \end{aligned}$$

In the expression for C, subscripts j and l refer to the lines of the matrix and subscripts h and q, the columns.

Chapter 3

Simulation Study

3.1 Previous Work and Existing Problems

Goulet [24] has implemented the optimal estimation of variance components for small portfolios of data in which every simulation consists of 300 runs on portfolios with $I = 4, J = 4$, and $T = 5$. As seen previously, the estimators of variance components are defined as solutions to a fixed-point problem. On this simulation, each iteration involves the construction of a large matrix and its inversion. Therefore, calculation of the optimal variance components estimators is very computer intensive, especially for large portfolios.

On the other hand, numerical tests of mean and standard deviation on $b^{(1)}, b^{(2)}, b^{(12)}$ require a large number of repetitions. For the repetitive number 5,000, for example, much computing resources will be required both in terms of storage and time. It is due to the inexpedient time delay in yielding solutions that it is difficult to perform

the simulation on large portfolios.

The main purpose of this thesis is to implement the estimators of variance components in such a way as to make computation for larger porfolios, of at least for $I = 10, J = 10$, and $T = 5$, feasible. This implementation will then be used to conduct a simulation study. It is therefore important to use an effective programming language to implement the calculation, to apply optimal algorithms and to look for tools to increase computation speed.

3.2 Structure of Matrix C and Attributes

The variance-covariance matrix C in estimators of $b^{(1)}, b^{(2)}, b^{(12)}$ and their inversion hinder the speed of execution. From the optimal estimation of $b^{(12)}, b^{(1)}$ and $b^{(2)}$ in a two-way crossed classification credibility model, we know that the estimation of $b^{(12)}$ is obtained by using equations (2.5)—(2.9) and Theorem 1.

Equation (2.8) shows that the size of matrix C increases by the order of $(IJ)^2$. For small numbers of categories, say $I = 3$ and $J = 3$, matrix C in $b_*^{(12)}$ already contains $[IJ(IJ-1)/2] \times [IJ(IJ-1)/2] = 36 \times 36 = 1,296$ elements. If $I = 10$ and $J = 10$, the size of matrix C increases to $4,950 \times 4,950 = 24,502,500$ elements.

Matrix C is a variance-covariance matrix. Since the variance-covariance of column i with column j is the same as the variance-covariance of column j with column i, the

variance-covariance matrix C is symmetrical.

$$
\begin{aligned}
C^{(12)}_{(ijkl)(ghpq)} &= \left(\sigma^{(12)}_{ijgh} - \sigma^{(12)}_{ijpq} - \sigma^{(12)}_{klgh} + \sigma^{(12)}_{klpq}\right)^2 \\
&= \left[\mathrm{Cov}(X_{ijw}, X_{ghw}) - \mathrm{Cov}(X_{ijw}, X_{pqw}) - \mathrm{Cov}(X_{klw}, X_{ghw}) + \mathrm{Cov}(X_{klw}, X_{pqw})\right]^2 \\
&= \left[\mathrm{Cov}(X_{ghw}, X_{ijw}) - \mathrm{Cov}(X_{pqw}, X_{ijw}) - \mathrm{Cov}(X_{ghw}, X_{klw}) + \mathrm{Cov}(X_{pqw}, X_{klw})\right]^2 \\
&= \left(\sigma^{(12)}_{ghij} - \sigma^{(12)}_{ghkl} - \sigma^{(12)}_{pqij} + \sigma^{(12)}_{pqkl}\right)^2 \\
&= C^{(12)}_{(ghpq)(ijkl)}
\end{aligned}
$$

Furthermore, we should also note that matrix C has none of the following features: sparse, scalar, singular, diagonal, lower triangular, upper triangular, permutation, indefinite symmetric, banded symmetric. Many algorithms which work on these matrices can not be applied in this simulation.

Matrix C is, however, positive definite [24]. The positive definite character of matrix C can also be checked by the Matlab "chol" function.

Therefore, the variance-covariance matrix C is a symmetric positive definite matrix.

3.3 Simulation of Variance Components

The simulation process is composed of four parts.

3.3.1 Structure of the Simulation on Variance Components

The following structure describes the whole procedure of calculating means and standard deviations for N estimators of three variance components $b^{(1)}, b^{(2)}, b^{(12)}$. Solutions are obtained by using a fixed-point algorithm. I is the first risk factor; J is the second factor; T is time when the insured's claims occur; m is the collective mean; s^2 is the time variance. Before computing every group of solutions, loss ratios X_{ijt} need to be simulated; w_{ijt}, which is the weight assigned to X_{ijt}, needs to be simulated as well; s^2 represents the estimation of the time variance. Tolerance is set at 10^{-6}.

```
Repeat N times
{
    Input I, J, T, m, s^2 ,b^(1) ,b^(2) ,b^(12) ;
    Simulate data---weight wijt;
    Simulate data---loss ratios Xijt;
    Compute structural parameter s2;
    Repeat
    {
        B^(12)= g12(b^(12),b^(1),b^(2));
        B^(1)= g1(B^(12),b^(1),b^(2));
        B^(2)= g2(B^(12),B^(1),b^(2));
```

```
    }
    until |(B^(12)-b^(12))/b^(12)|<=1e-6
      and |(B^(1)-b^(1))/b^(1)|<=1e-6
      and |(B^(2)-b^(2))/b^(2)|<=1e-6
    Output and Record a new set of solutions on b^(1) ,b^(2) ,b^(12).
}
Calculate mean of N times b^(1) ;
Calculate standard deviation of N times b^(1);
Calculate mean of N times b^(2);
Calculate standard deviation of N times b^(2);
Calculate mean of N times b^(12);
Calculate standard deviation of N times b^(12).
```

3.3.2 Model of $b^{(12)}$

Figure 3.1 describes the structure of the variance component $b^{(12)}$. It represents the relationship between input and output from bottom to top for different variables and matrices. It shows $b^{(1)}, b^{(2)}, b^{(12)}$, s^2, $wijt$ and $Xijt$ working as input parameters for function $b^{(12)}$. All the equations can be found in section 2.2, estimation of $b^{(12)}$. This hierarchical diagram provides a clear representation of the relationship between these

equations.

3.3.3 Model of $b^{(1)}$

Figure 3.2 describes the structure of variance component $b^{(1)}$ and the relationship between different variables and matrice. $b^{(1)}, b^{(2)}, b^{(12)}$, s^2, $wijt$ and $Xijt$ work as input parameters for function $b^{(1)}$. These equations can be found in section 2.2, estimation of $b^{(1)}$.

3.3.4 Model of $b^{(2)}$

Figure 3.3 shows the structure of variance component $b^{(2)}$ and the relationship between different variables and matrice. $b^{(1)}, b^{(2)}, b^{(12)}$, s^2, $wijt$ and $Xijt$ work as input parameters for function $b^{(2)}$. These equations can be found in section 2.2, estimation of $b^{(2)}$.

3.4 Properties of Fixed-Point Algorithm

Iteration is a fundamental technique in numerical computing and other algorithms in computer science. Iteration techniques are used to find the roots of equations, solutions to linear and nonlinear systems of equations, and solutions to differential equations.

The fixed point iteration method is applied in estimation of optimal variance compo-

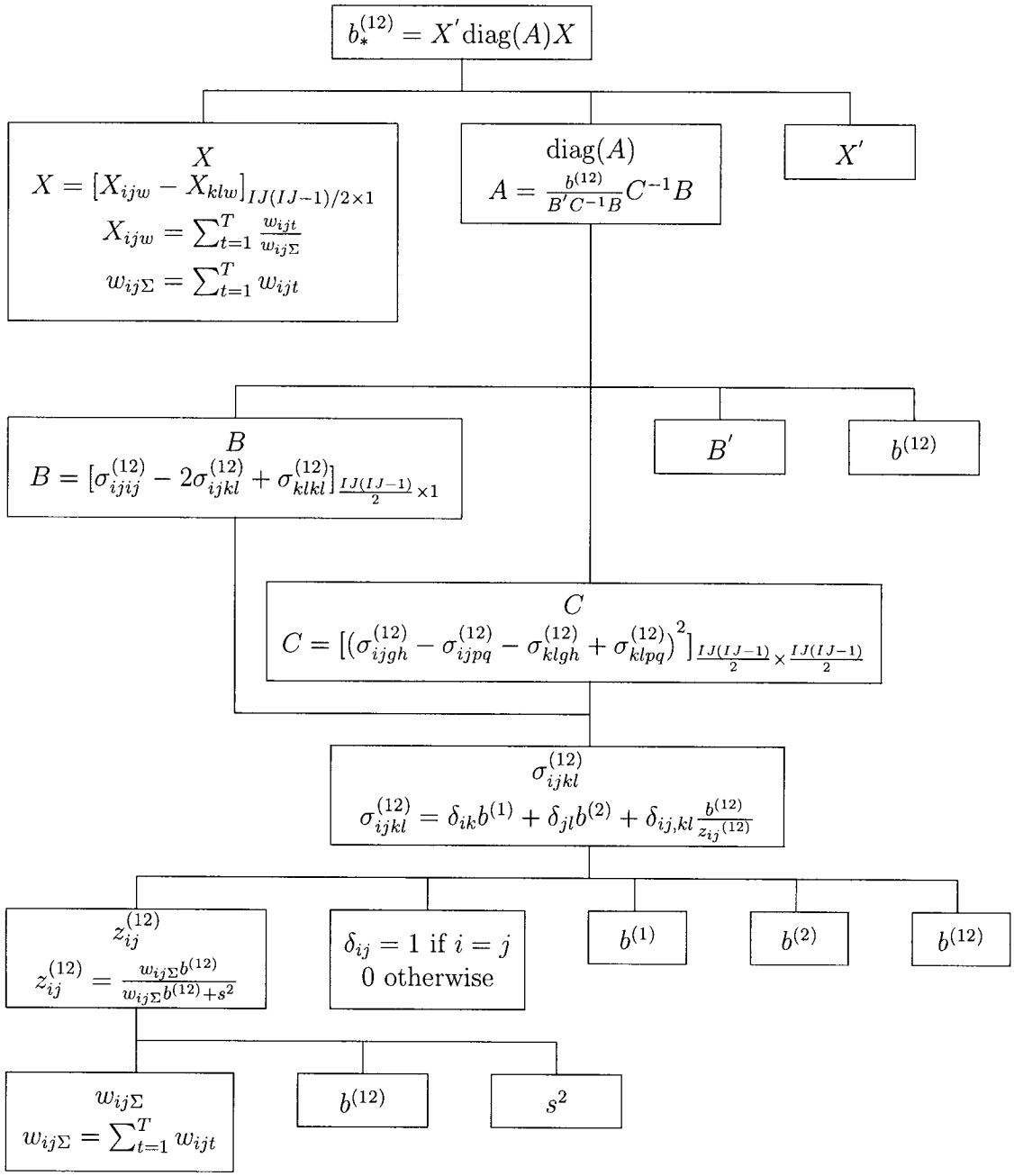

Figure 3.1: Model of Structural Parameter $b^{(12)}$

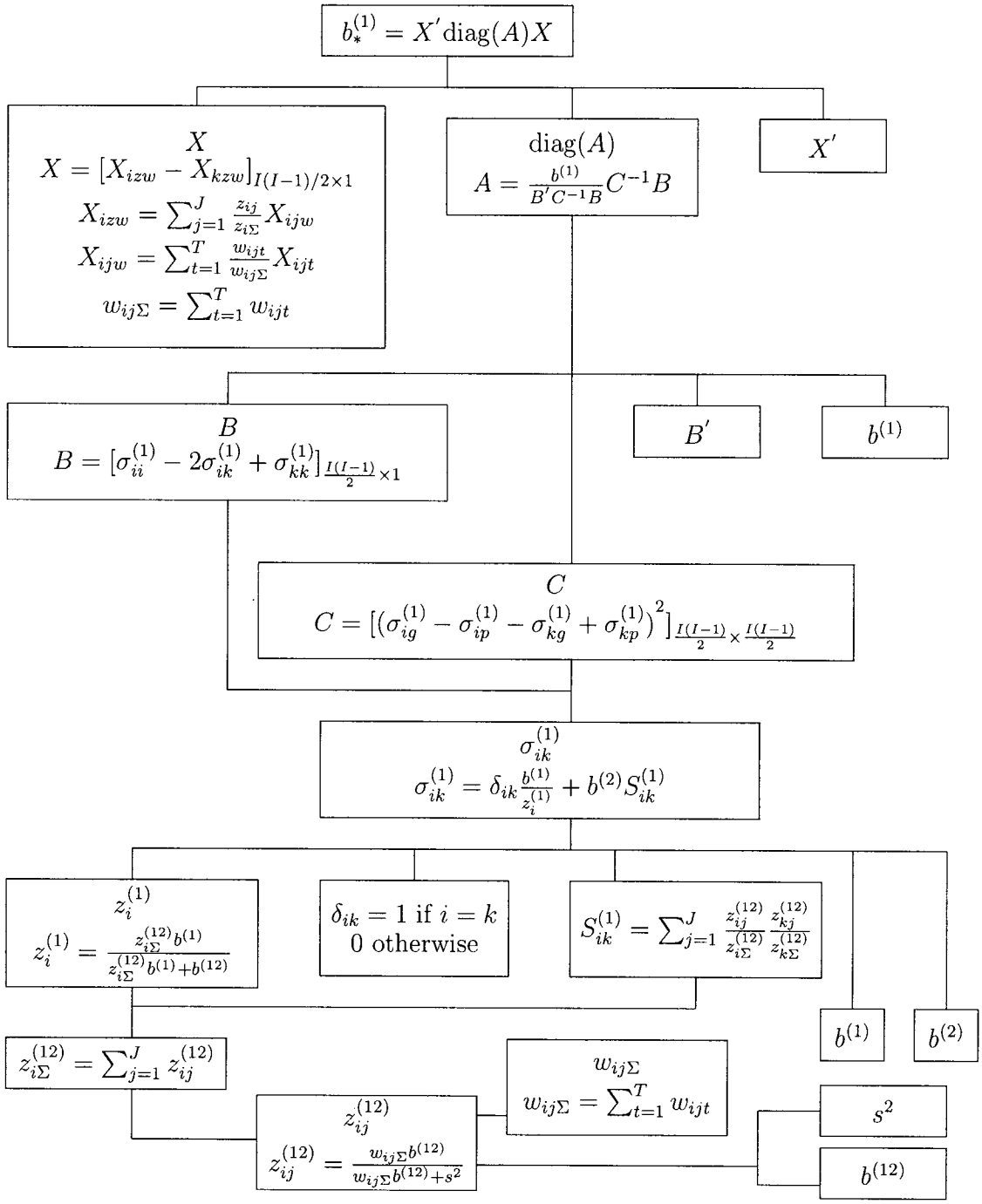

Figure 3.2: Model of Structural Parameter $b^{(1)}$

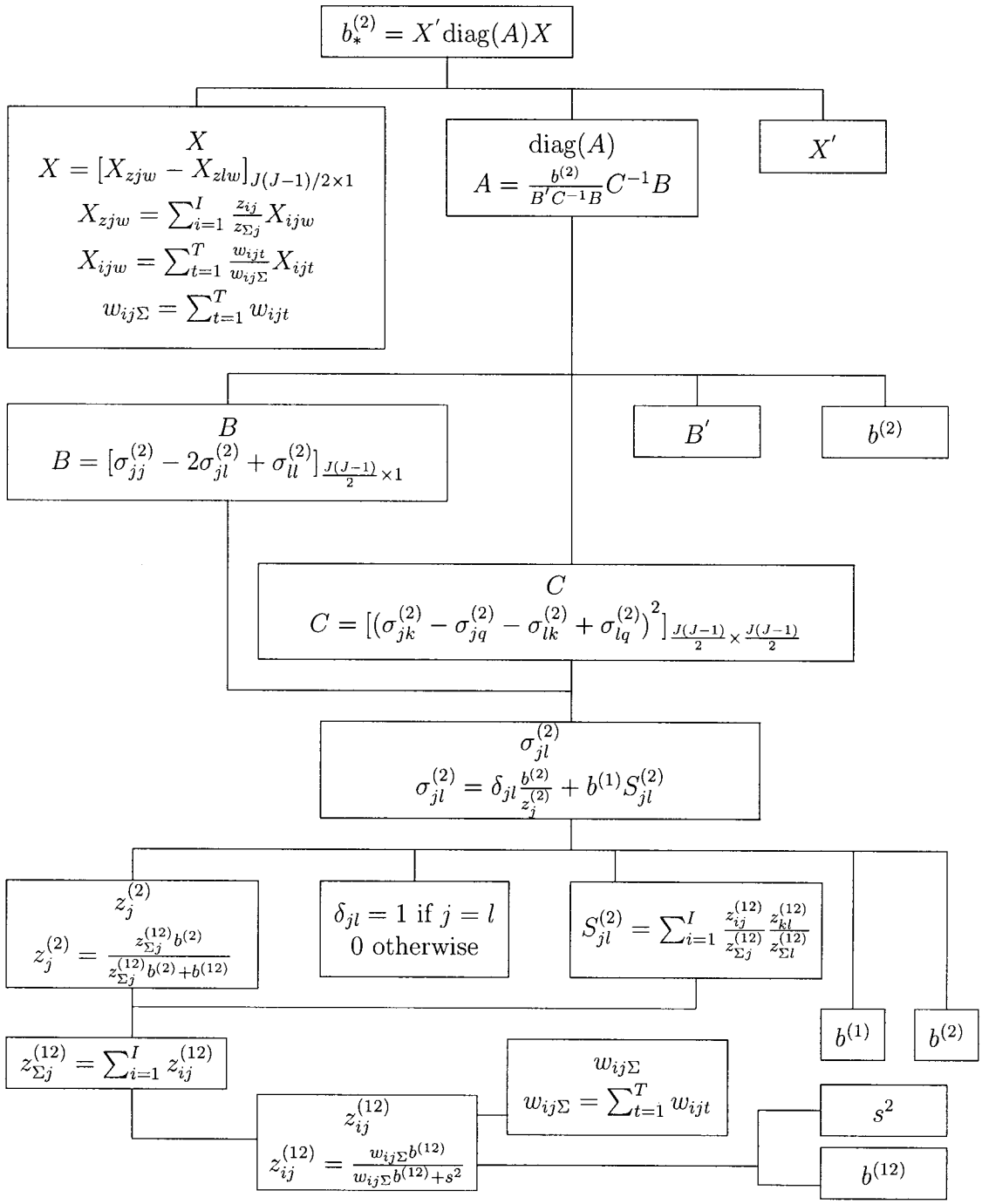

Figure 3.3: Model of Structural Parameter $b^{(2)}$

nents which includes many iterations. In every iteration, the generation of $b^{(12)}, b^{(1)}$ and $b^{(2)}$ involves the calculation of matrix C.

Before introducing properties of fixed point algorithm, it is useful to define fixed point and relative theorems [6].

Definition 1. Fixed Point

The fixed point of a function $g(x)$ is a number P such that $P = g(P)$.

Definition 2. Fixed Point Iteration

The iteration $P_n = g(P_{n-1})$ for $n = 0, 1, ...$ is called a fixed point iteration.

Theorem 5. Converging sequence

Assume that $g(x)$ is a continuous function and that $[P_n]_{n=0}^{\infty}$ is a sequence generated by fixed point iteration. If

$$\lim_{n\to\infty} P_n = P,$$

then P is a fixed point of $g(x)$.

The following two theorems establish conditions for the existence of a fixed point and the convergence of the fixed-point iteration process to a fixed point.

Theorem 6. The First Fixed Point Theorem

If we assume that $g \in C[a, b]$, i.e. $g(x)$ is continuous on $[a, b]$, then we have the following results:

(i) If the range of the mapping $Y = g(x)$ satisfies $Y \in [a, b]$ for all $x \in [a, b]$, then g

has a fixed point in $[a, b]$;

(ii) Furthermore, suppose that $g'(x)$ is defined over (a, b) and that a positive constant $K < 1$ exists with $|g'(x)| \leq K$ for all $x \in (a, b)$, then g has a unique fixed point P in $[a, b]$.

Theorem 7. The Second Fixed Point Theorem

Assuming that the following hypotheses hold true,

(a) P is a fixed point of a function g;

(b) $g, g' \in C[a, b]$;

(c) K is a positive constant;

(d) $P_0 \in (a, b)$; and

(e) $g(x) \in [a, b]$ for all $x \in [a, b]$,

then we have the following conclusions.

(i). If $|g'(x)| \leq K < 1$ for all $x \in [a, b]$, then the iteration $P_n = g(P_{n-1})$ will converge to the unique fixed point $P \in (a, b)$. In this case, P is said to be an attractive fixed point.

(ii). If $|g'(x)| > 1$ for all $x \in [a, b]$, then the iteration $P_n = g(P_{n-1})$ will not converge to P. In this case, P is said to be a repelling fixed point and the iteration exhibits

local divergence.

In the estimation of the variance components, the three variance components optimal pseudo-estimators form the multidimensional fixed point of a system of equations of the type

$$\begin{aligned} b_*^{(12)} &= g_{12}(b^{(1)}, b^{(2)}, b^{(12)}) \\ b_*^{(1)} &= g_1(b^{(1)}, b^{(2)}, b^{(12)}) \\ b_*^{(2)} &= g_2(b^{(1)}, b^{(2)}, b^{(12)}) \, . \end{aligned}$$

The convergence of the iterative procedure can be accelerated by Gauss Siedel's method [43]. The Gauss Siedel method always uses the most up to date iterate for any particular variable. In order to calculate $b_*^{(1)}$, we use $b^{(12)}$. However, since iterate $b_*^{(12)}$ has already been calculated and is more accurate, we use $b_*^{(12)}$ instead of $b^{(12)}$. $b_*^{(2)}$ would, thus, be calculated by using $b_*^{(12)}$ and $b_*^{(1)}$. Using the Gauss Siedel method, the iterative scheme will be

$$\begin{aligned} b_*^{(12)} &= g_{12}(b^{(1)}, b^{(2)}, b^{(12)}) \\ b_*^{(1)} &= g_1(b^{(1)}, b^{(2)}, b_*^{(12)}) \\ b_*^{(2)} &= g_2(b_*^{(1)}, b^{(2)}, b_*^{(12)}) \, . \end{aligned}$$

This kind of process will be repeated until a preestablished stopping criterion is satisfied.

Functions g_{12}, g_1, and g_2 are not simple. Figure 3.2, 3.3 and 3.4 describe the steps needed to obtain the values of estimators $b_*^{(12)}$, $b_*^{(1)}$ and $b_*^{(2)}$. Here, g_{12} is given as an example to describe the procedure.

As we know from chapter 2,

$$b_*^{(12)} = X' \operatorname{diag}\left(\frac{b^{(12)}}{B'C^{-1}B} C^{-1}B\right) X \tag{3.1}$$

$$B = [\sigma_{ijij}^{(12)} - 2\sigma_{ijkl}^{(12)} + \sigma_{klkl}^{(12)}]_{\frac{IJ(IJ-1)}{2}\times 1} \tag{3.2}$$

$$C = [(\sigma_{ijgh}^{(12)} - \sigma_{ijpq}^{(12)} - \sigma_{klgh}^{(12)} + \sigma_{klpq}^{(12)})^2]_{\frac{IJ(IJ-1)}{2}\times\frac{IJ(IJ-1)}{2}} \tag{3.3}$$

$$X = [X_{ijw} - X_{klw}]_{\frac{IJ(IJ-1)}{2}\times 1} . \tag{3.4}$$

If $I = 2, J = 3, K = 2, L = 3$; δ_{ij} equals 1 when $i = j$ and 0 otherwise; $\delta_{ij,kl}$ equals 1 when $i = k, j = l$ and 0 otherwise, B_{0101} will be obtained by

$$\begin{aligned}
B_{0101} &= \sigma_{1111}^{(12)} - 2\sigma_{1112}^{(12)} + \sigma_{1212}^{(12)} \\
&= \delta_{11}b^{(1)} + \delta_{11}b^{(2)} + \delta_{11,11}\frac{b^{(12)}}{z_{11}^{(12)}} - 2(\delta_{11}b^{(1)} + \delta_{12}b^{(2)} + \delta_{11,12}\frac{b^{(12)}}{z_{11}^{(12)}}) \\
&\quad +\delta_{11}b^{(1)} + \delta_{22}b^{(2)} + \delta_{12,12}\frac{b^{(12)}}{z_{12}^{(12)}} \\
&= b^{(1)} + b^{(2)} + \frac{b^{(12)}}{z_{11}^{(12)}} - 2b^{(1)} + b^{(1)} + b^{(2)} + \frac{b^{(12)}}{z_{12}^{(12)}} \\
&= 2b^{(2)} + \frac{b^{(12)}}{\frac{w_{11\Sigma}b^{(12)}}{w_{11\Sigma}b^{(12)}+s^2}} + \frac{b^{(12)}}{\frac{w_{12\Sigma}b^{(12)}}{w_{12\Sigma}b^{(12)}+s^2}} .
\end{aligned}$$

This procedure is repeated until we get all elements of matrix B:

$$B = \begin{pmatrix} B_{0101} \\ \cdots \\ B_{1501} \end{pmatrix}$$

C_{0101} will be obtained by

$$\begin{aligned}
C_{0101} &= (\sigma^{(12)}_{1111} - \sigma^{(12)}_{1112} - \sigma^{(12)}_{1211} + \sigma^{(12)}_{1212})^2 \\
&= [\delta_{11}b^{(1)} + \delta_{11}b^{(2)} + \delta_{11,11}\frac{b^{(12)}}{z^{(12)}_{11}} - (\delta_{11}b^{(1)} + \delta_{12}b^{(2)} + \delta_{11,12}\frac{b^{(12)}}{z^{(12)}_{11}}) \\
&\quad -(\delta_{11}b^{(1)} + \delta_{21}b^{(2)} + \delta_{12,11}\frac{b^{(12)}}{z^{(12)}_{12}}) + (\delta_{11}b^{(1)} + \delta_{22}b^{(2)} + \delta_{12,12}\frac{b^{(12)}}{z^{(12)}_{12}})]^2 \\
&= (b^{(1)} + b^{(2)} + \frac{b^{(12)}}{z^{(12)}_{11}} - b^{(1)} - b^{(1)} + b^{(1)} + b^{(2)} + \frac{b^{(12)}}{z^{(12)}_{12}})^2 \\
&= \left(2b^{(2)} + \frac{b^{(12)}}{\frac{w_{11\Sigma}b^{(12)}}{w_{11\Sigma}b^{(12)}+s^2}} + \frac{b^{(12)}}{\frac{w_{12\Sigma}b^{(12)}}{w_{12\Sigma}b^{(12)}+s^2}}\right)^2 .
\end{aligned}$$

This procedure is repeated until we get all elements of matrix C:

$$C = \begin{pmatrix} C_{0101} & \dots & C_{0115} \\ \dots & \dots & \dots \\ C_{1501} & \dots & C_{1515} \end{pmatrix}$$

From theorem 7, proof of uniqueness of a fixed point requires the calculation of the partial derivatives of equation (3.1). This function has a very complex structure and is thus very difficult to work with. As a result, we were not able to prove existence and uniqueness of the fixed point analytically.

In numerical tests, the iterative procedure always converges to a unique fixed-point independent of the starting values. Furthermore, coordinates of the fixed-point are always non negative. In this experiment, optimal estimators are present and positive.

According to Gauss Siedel's method, every consecutive function has to wait for the solution of the previous function as the input parameter. Hence, the three functions for $b^{(12)}_*, b^{(1)}_*, b^{(2)}_*$ can not be implemented by using parallel computing.

3.5 Problem Analysis and Resolving Method

This simulation is implemented by using C language, Matlab C Math library, and R math library on Linux platform in order to allocate and free memory dynamically, and increase execution speed.

3.5.1 Solving a System of Linear Equations

Generation of matrix A involves the inversion of matrix C. Calculation of the matrix C inversion, which is $O(N^3)$, increases dramatically with the size of the matrix. It thus requires a lot of arithmetic which results in slower simulation. Inverting a matrix quickly and stably is usually not simple. Many optimal algorithms (including Gauss-Jordan Elimination algorithm) used to calculate C^{-1}, have been tried and applied in this program. However, there is no obvious improvement in running speed. Furthermore, in the vast majority of practical computational problems, it is unnecessary and inadvisable to actually compute the inversion of a matrix. Therefore, it is for these reasons that we have taken structure of matrix A into consideration.

From the expression of matrix A, we see

$$A = \frac{b^{(12)}}{B'C^{-1}B} C^{-1}B \ .$$

Instead of computing the C^{-1}, we can combine the C^{-1} and B together. If we consider

$C^{-1}B$ as an object, and assign vector M as their solution, we have:

$$M = C^{-1}B$$

$$CM = B$$

Compared with Gauss-Jordan and LU decomposition algorithms, Gauss-Jordan is three times slower than LU decomposition, which is the best alternative technique for solving a single linear set. Therefore, LU decomposition has been applied in this simulation.

Suppose we are able to write matrix C as a product of two matrices.

$$LU = C \,, \tag{3.5}$$

where L is the lower triangular and U is the upper triangular. For the case of 3×3 matrix C, equation (3.5) would be written:

$$\begin{bmatrix} \alpha_{11} & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ 0 & \beta_{22} & \beta_{23} \\ 0 & 0 & \beta_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} .$$

We can use decomposition to solve this linear set

$$CM = (LU)M = L(UM) = B \,. \tag{3.6}$$

First by solving the vector Y using

$$LY = B \,, \tag{3.7}$$

and then solving

$$UM = Y \ . \tag{3.8}$$

Therefore, we can use forward substitution to solve the equation (3.7) as follows:

$$y_1 = \frac{b_1}{\alpha_{11}} \tag{3.9}$$

$$y_i = \frac{1}{\alpha_{ii}}[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j], \quad i = 2, 3, ..., N \tag{3.10}$$

while we can use backsubstitution to solve (3.8):

$$m_N = \frac{y_N}{\beta_{NN}} \tag{3.11}$$

$$m_i = \frac{1}{\beta_{ii}}[y_i - \sum_{j=i+1}^{N} \beta_{ij} m_j], \quad i = N-1, N-2, ..., 1 \tag{3.12}$$

LU Decomposition's application simplifies computing of matrix A. Calculation of the inversion of matrix C is replaced by solving linear equations (see equation 3.6).

3.5.2 Cholesky Decomposition Method

We saw earlier that matrix C is symmetrical and positive definite. Cholesky decomposition is the best way to solve the system of linear equations proper to these matrices.

Cholesky decomposition is approximately 2 times faster than alternative methods for solving linear equations [36]. Instead of seeking arbitrary lower and upper triangular factors L and U, Cholesky decomposition constructs a lower triangular matrix L

whose transpose L^T can itself serve as the upper triangular. C can be then replaced by

$$L \cdot L^T = C \,, \tag{3.13}$$

where

$$L_{ij}^T = L_{ji} \,,$$

from which we obtain

$$L_{ii} = \left(c_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{\frac{1}{2}} \,, \tag{3.14}$$

and

$$L_{ji} = \frac{1}{L_{ii}} \left(c_{ij} - \sum_{k=1}^{i-1} L_{ik} L_{jk} \right), \quad j = i+1, i+2, ..., N \,. \tag{3.15}$$

By applying the two equations above in the order $i = 1, 2, ..., N$, we will see that $L's$ which occur on the right-hand side are already determined by the time they are needed. Also, only components c_{ij} with $j \geq i$ are referenced. It is practicable to employ L to overwrite the subdiagonal part of C, preserving the input upper triangular values of C. Only one extra vector of length N is needed to store the diagonal part of L. The operations count is $N^3/6$ executions of the inner loop with N square roots. It is approximately 2 times better than the LU decomposition of C. Therefore, executation speed of this simulation is improved by using Cholesky algorithm.

3.5.3 Application of Standard Subroutine Packages

Normally, for a set of linear algebraic equations, when N is too large, the accumulated roundoff errors in the solution process can affect the true solution, and the set of m's are inaccurate. Much of the sophistication of complicated "linear equation-solving packages" is devoted to the correction of such pathologies. Basically, linear sets with N as large as 20 or 50 can be routinely solved with single precision without using sophisticated methods. With double precision, this number can be extended to N as large as several hundred, after which point the limiting factor is generally machine time, not accuracy [36]. Because N of equation (3.6) will increase the closer it comes to 5,000, we have no alternative but to use sophisticated program packages to resolve the linear equation problem.

Sophisticated packages have a number of advantages. They are designed for very large systems of equations. Therefore, they go to great effort to minimize not only the number of operations, but also the required storage. Routines for various tasks are usually provided in several versions, conrresponding to several possible simplifications in the form of the input coefficient matrices: symmetric, triangular, banded, positive definite, etc. Provided with these different routines, executions of a large matrix with these forms can be accelerated efficiently.

Many standard packages are available for large linear systems including the following:

Matlab, Linpack, Lapack, ScaLAPACK, PLapack, Blas, Atlas, NAG, GSL, ESSL, IMSL, CMTM, etc.

Through a comparison of a few libraries, we find that Lapack is the new replacement for Eispack and Linpack, the latter of which are based on the vector operation kernels of the level 1 Blas. Lapack is a large, multi-author, Fortran library for numerical linear algebra and is designed to exploit the level 3 Blas at most. It provides matrix factorizations, such as, LU, Cholesky and QR, etc. Lapack was originally intended for use on supercomputers and other high-end machines. It uses block algorithms, which operate on several columns of a matrix at a time in the innermost loops. These block operations can be optimized for each architecture to account for the memory hierarchy, and so provide a transportable way to achieve high efficiency on diverse modern machines. Lapack also provides a more extensive set of capabilities than its predecessors [44].

Therefore, Lapack Cholesky routines have become the first choice to accelerate execution. Indeed, any library which integrates with the Lapack Cholesky routines can make execution of this simulation faster.

3.5.4 Algorithm of Lapack Cholesky Routines

In this section, we introduce a basic version and a block version of Cholesky algorithm [33] implemented by Lapack routines spotf2 and spotrf so as to demonstrate the

advantages of block Cholesky algorithm. As the name suggests, the block version of Cholesky algorithm operates on blocks, or submatrices of the original matrix and is therefore rich in matrix-matrix operations. Since these operations (level 3 Blas) are faster, block algorithm are usually more efficient.

Algorithm 3.1 (Gaxpy version of Cholesky algorithm). This algorithm computes a lower triangular $L \in R^{n \times n}$ such that $A = LL^T$, and L overwrites the lower triangular of A:

for $j = 1 : n$

// Compute $A(j, j)$

1. $A_{j,j} \leftarrow A_{j,j} - A_{1:j-1,j} A_{1,j-1,j}^T$

// Test for non-positive-definiteness and compute $L(j, j)$

if $A_{j,j} <= 0$

then stop

else

2. $A_{j,j} \leftarrow \sqrt{A_{j,j}}$

endif

// Compute elements $j + 1 : n$ of column j

if $(j < n)$ then

3. $A_{j+1:n,j} \leftarrow A_{j+1:n,j} - A_{j+1:n,1:j-1} A_{j,1:j-1}^T$

4. $A_{j+1:n,j} \leftarrow A_{j+1:n,j} / A_{j,j}$

endif

end

Algorithm 3.2 (Block version of Cholesky algorithm).

// matrix A is partitioned in blocks of size nb.

for $j = 1 : (n/nb + 1)$

$s = (j-1).nb + 1$ // start of block to factorize

$e = \min(j.nb, n)$ // end of block to factorize

$u = e + 1$ // start position for update

// Update the current diagonal block

if $j > 1$ then

1. $A_{s:e,s:e} \leftarrow A_{s:e,s:e} - A_{s:e,1:s-1} A^T_{s:e,1:s-1}$

endif

// Factorize the current diagonal block

2. $A_{s:e,s:e} \leftarrow G$, with $GG^T = A_{s:e,s:e}$

if $1 < j <= n/nb$ then

// Update the current block column

3. $A_{u:n,s:e} \leftarrow A_{u:n,s:e} - A_{u:n,1:s-1} A^T_{s:e,1:s-1}$

endif

if $j <= n/nb$ then

4. $A_{u:n,s:e} \leftarrow A_{u:n,s:e} / A^T_{s:e,s:e}$

endif

end

This block version algorithm is also called the block left-looking version of Cholesky algorithm. In this version, A is factorized one block column at a time. For each j, step 3 is a rank $s-1$ update that accesses the previously factorized block columns.

3.5.5 Integration with Matlab C Math Library

With respect to the structure of matrix C, Lapack Cholesky algorithm is applied to resolve problems of calcualtion on linear equations in order to speed up execution.

Matlab C Math Library [40] is based on the Matlab language and constitutes the mathematical core of Matlab. It is a collection of more than 400 mathematical optimal routines written in C. The mathematical routines in the Matlab C Math Library are C callable versions of features of the Matlab language. Programs written in any language capable of calling C functions can, thus, call these routines to perform mathematical computations.

In the simulation established in this study, we integrated Matlab C Math Library with a C program. To employ Matlab C Math Library in a C program, it is necessary to include `matlab.h` in the code.

However, before running a stand-alone application, we have to locate shared libraries,

and tell the system where the API and C shared libraries reside. The following statements set this path.

```
% setenv LD_LIBRARY_PATH [matlab]/extern/lib/[arch]: $ LD_LIBRARY_PATH
```

where, `[matlab]` is the Matlab root directory, and `[arch]` is the architecture.

Matlab C Math Library requires the following command to compile a C program on Linux:

```
% mbuild -setup filename.c
```

3.5.6 Application of Back Slash

In Matlab, *mldivide*(A, B) and the equivalent $A \setminus B$ perform matrix left division (back slash). A and B must be matrices that have the same number of rows. If A is a square matrix, then $A \setminus B$ is roughly the same as $A^{-1}B$, but it is computed in a different way. If A is an $n \times n$ matrix and B is a column vector with n elements, or a matrix with several such columns, then $X = A \setminus B$ is the solution to the equation $AX = B$ computed by Gaussian elimination with partial pivoting. A warning message is displayed if A is badly scaled or close to singular.

Algorithm Description:

The specific algorithm used for solving the simultaneous linear equations denoted by

$X = A \setminus B$ depends upon the structure of the coefficient matrix A. To determine the structure of A and in order to select the appropriate algorithm, Matlab follows this procedure:

1. If A is sparse and diagonal, X is computed by dividing by the diagonal elements of A.

2. If A is sparse, square, and banded, then a banded solver is used.

3. If A is an upper or lower triangular matrix, then X is quickly computed with a backsubstitution algorithm for upper triangualar matrices, or a forward substitution algorithm for lower triangular matrices.

4. If A is a permutation of a triangular matrix, then X is computed with a permuted backsubstitution algorithm.

5. If A is symmetric, or Hermitian, and has real positive diagonal elements, then a Cholesky factorization is attempted. If A is found to be positive definite, the Cholesky factorization attempt is successful and requires less than half the time of a general factorization. Nonpositive definite matrices are usually detected almost immediately, so this check also requires little time.

   If successful, the Cholesky factorization for full A is

$$A = R' \times R\,.$$

where R is upper triangular. The solution X is computed by solving two triangular systems:

$$X = R \setminus (R' \setminus B)$$

Computations are performed using the Lapack routines given the following table.

	Real	Complex
A and B double	DLANGE, DPOTRF, DPOTRS, DPOCON	ZLANGE,ZPOTRF, ZPOTRS,ZPOCON
A or B single	SLANGE, SPOTRF, SPOTRS, SDPOCON	CLANGE,CPOTRF, CPOTRS, CPOCON

6. If A is Hessenberg, but not sparse, it is reduced to an upper triangular matrix and that system is solved via substitution.

7. If A is square and does not satisfy criteria 1 through 6, then a general triangular factorization is computed by Gaussian elimination with partial pivoting.

8. If A is not square, then Householder reflections are used to compute an orthogonal-triangular factorization.

According to this algorithm, in this simulation, because matrix C is a symmetric positive definite, a Cholesky factorization algorithm is applied to solve these linear equations, which requires less time to execute. Solving this problem depends on Lapack routines.

3.5.7 Memory Allocation

Application of fixed-point algorithm and a great deal of matrix operations require us to consider how to use limited memory to carry out calculations for unknown large porfolios of input data.

C language is a very popular language for computing. Application of pointers instead of arrays used in R programming provides dynamic memory allocation. The use of arrays in which memory is allocated at a definite time and can not be recycled, is the reason why simulation in R can not be used for the calculation of large portfolios. Memory for a dynamic matrix or vector in C is allocated when needed, and is freed when not needed. Therefore, C is the language selected to implement the simulation in this study.

3.5.8 Application of R Math Library

From the structure of the program, we can see that data weights w_{ijt} and loss ratios X_{ijt} must be generated randomly before continuing to calculate structural parameters. The statistical environment R features all needed random number generators. The R Math Library [18] has entry points for C code and can be used in C program easily. The R Math library is used to generate random uniform and normal variables.

The interface to R's internal random number generation routines is:

```
double unif_rand();

double norm_rand();
```

or `double rnorm(double mu, double sigma);`

Uniform or normal pseudo-random variables are generated on these given routines.

The C code behind R's functions can be accessed by including the header file `Rmath.h` in a C file and linked against '`-lRmath`'.

```
#define MATHLIB_STANDALONE

#include <Rmath.h>
```

3.6 Accuracy of Solution

We check our results against those obtained by Goulet [24] for the same portfolios in R. We obtained the same solutions. We also tested every function in this simulation and have made sure that the codes themselves are correct. We are confident that the solutions are accurate.

3.7 Speed Test

If we assume I=10, J=10, T=5, $b^{(1)}$=1.0, $b^{(2)}$=1.0, $b^{(12)}$=1.0, s^2=6.450058, then we obtained the following solutions in 9 minutes on Linux.

```
[credsim@confsys /src]./3_opt_B2
```

```
RepeatTime=7, Final b_12= 8.86160772448163669424

RepeatTime=7, Final b_1= 5.87119735197586045672

RepeatTime=7, Final b_2= 4.05814014651073495799
```

The same solution is obtained in R in 22 minutes on Linux with Pentium 2 at 200 MHz. Because the R code uses function "solve()" which integrates Linpack to calculate $C^{-1}B$, execution is slower than using Matlab backslash which uses Lapack library. Therefore, execution speed in C is thus less than half of the speed in the R program.

3.8 Integration with R

R [46] is a language and environment that is used for statistical computing and graphics. R provides a wide variety of statistical and graphic techniques, and is highly extensible. S language [38] is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

R is designed around a true computer language, and it allows users to add additional functionality by defining new functions. For computationally intensive tasks, C, C++, and Fortran code can be linked and called at run time. Estimation of the variance components has been implemented in R. Because of the restriction of R language itself, it is difficult to obtain simulation solutions for large porfolios. At the outset of this research, it was assumed that estimation of the variance components which

includes intensive computational tasks could be implemented in C, and integrated C codes with R code in order to carry out statistical analysis.

Research on integrating C with R was processed during this simulation study. R provides a standard interface ".C( )" to compiled C code that has been linked into R. Under Linux, in order to load a C function into R, for example `convolve.c`, a shared library must be created by

```
R CMD SHLIB convolve.c [...]
```

optionally using the `-o` flag to set the name of the library, which here will default to `convolve.so` or `convolve.sl` as appropriate. Then, it can be loaded by

```
dyn.load("PATHTOCONVOLVE/convolve")
```

Where `PATHTOCONVOLVE` is the path name to the `convolve` library file. Shared libraries can be generated automatically as part of the installation of an R package.

Below is an example [18] of a C function `convolve.c`:

```
void convolve(double *a, int *na, double *b, int *nb, double *ab)
{
  int i, j, nab=*na+*nb-1;
```

```
  for (i=0;i<nab;i++)
    ab[i]=0.0;
  for (i=0;i<*na;i++)
    for (j=0;j<*nb;j++)
     ab[i+j]+=a[i]*b[j];
}
```

The detailed operation of loading `convolve.c` under Linux is as follows:

```
[credsim@confsys ~/src]$ R CMD SHLIB convolve.c
make: 'convolve.so' is up to date.
[credsim@confsys ~/src]$ locate convolve
/home/credsim/src/convolve.o
/home/credsim/src/convolve.c
/home/credsim/src/convolve.so
[credsim@confsys ~/src]$ R
[Previously saved workspace restored]
> dyn.load("/b2/home/credsim/src/convolve.so")
> testa <- c(1:10)
> testb <- c(1:10)
> conv <- function(a,b) {
```

```
+ .C("convolve",
+ as.double(a),
+ as.integer(length(a)),
+ as.double(b),
+ as.integer(length(b)),
+ ab=double(length(a)+length(b)-1)) $ab
+ }
> conv(testa, testb)
 [1]   1   4  10  20  35  56  84 120 165 220 264 296 315 320 310
      284 241 180 100
```

The experiment demonstrates that a C function without libraries included could be loaded into R following the above steps. However, although a C function with libraries can be created as a shared library, it can not be loaded dynamically into R by using `dyn.load( )`. Therefore, the operations are as follows:

```
[credsim@confsys ~/src]$ R CMD SHLIB main.c
gcc -I/usr/lib/R/include  -I/usr/local/include -D__NO_MATH_INLINES
-mieee-fp -fPIC  -O2 -g -march=i386 -mcpu=i686 -c main.c -o main.o
gcc -shared -L/usr/local/lib -o main.so main.o
[credsim@confsys ~/src]$ R
```

```
[Previously saved workspace restored]
> dyn.load("main.so")
Error in dyn.load(x, as.logical(local), as.logical(now)) :
        unable to load shared library "/home/credsim/src/main.so":
  /home/credsim/src/main.so: undefined symbol: rnorm
```

Here, `rnorm` is the normal derivate generation function in `libRmath`. This means `libRmath` is not loaded in R dynamically. Alternatively, if the R math library is dropped from this simulation, and only Matlab C library is included in the C code, the operation is as follows:

```
[credsim@confsys ~/src]$ R CMD SHLIB main.c
gcc -I/usr/lib/R/include  -I/usr/local/include -D__NO_MATH_INLINES
-mieee-fp  -fPIC  -O2 -g -march=i386 -mcpu=i686 -c main.c -o main.o
gcc -shared -L/usr/local/lib -o main.so main.o
[credsim@confsys ~/src]$ R
[Previously saved workspace restored]
> dyn.load("main.so")
Error in dyn.load(x, as.logical(local), as.logical(now)) :
        unable to load shared library "/home/credsim/src/main.so":
  /home/credsim/src/main.so: undefined symbol: mclCleanupProtectedItems
```

The routine `mclCleanupProtectedItems` is found in `libmatlb.so`. This means Matlab C library isn't loaded in R.

In this simulation, a C program includes static `libRmath` and Matlab C library. However, R integrates with C codes dynamically. Experiments have shown that R is difficult to connect with C codes which include libraries. Even if the Matlab C library is replaced by Lapack, Atlas or other linear algebra libraries, R still can not connect with a C code which includes `libRmath`.

In R manual released in 2003 [41], it is stated:

> The linear algebra routines in R can make use of enhanced Blas routines. Optimized versions of Lapack are available, but no provision is made for using them with R as the likely performance gains are thought to be small.

This simulation study applies Lapack Cholesky routines for a symmetric positive definite coefficient matrix to accelerate execution, and does not make use of Blas. By the same token, R itself contains a large number of Blas, Linpack and Eispack linear algebra functions which are included in the header file `R_ext/Linpack.h`, but it does not contain Lapack routines. Therefore, it is difficult to realize the C implementation of both speed acceleration and R integration.

However, in the R manual released in 2004 November [42], provision is made for using an external Lapack library. This Lapack is written in Fortran, rather than in C. R provides a set of macros to cover up the platform-specific differences. In theory, it is possible to apply `R_ext/Lapack.h` routines in a C code which, however, can not be integrated with R directly. We have tried to create a R package which includes a R code and a C code with `R_ext/Lapack.h`. Problems that arise still can not be resolved. Although efforts were made in this study to integrate a C code (libraries included) with R, we could not implement it finally.

Therefore, we would recommend that further research on integrating R and C with libraries will be done in the future. In order to bypass this problem, in this simulation, part of the R code is translated into C code. The solutions obtained from this simulation study are presented in Chapter 5.

Chapter 4

Software Implementation

The main focus in the previous chapters has been on the theory and methods for the implementation of the simulation study on optimal variance components estimators in the CCC model. This chapter includes problem description, input and output analysis, the process of implementing the software for this numerical evaluation, and implementation strategies.

4.1 Problem Description

The application is used to implement a simulation study on optimal variance components estimators $b^{(12)}, b^{(1)}, b^{(2)}$ in a two-way crossed classification credibility model, which are finally applied to credibility estimator of the future insurance premium. Simulation of data requires inputs of I, J, T, m, s^2, $b^{(12)}$, $b^{(1)}$, and $b^{(2)}$. Variance components estimation requires starting values of $b^{(12)}$, $b^{(1)}$, $b^{(2)}$. The simulation study requires the number of runs nr. First, the implementation simulates random

observations (loss ratios) `X_ijt` and natural weight `w_ijt`, which are used for computing estimation values of `b_12, b_1, b_2`. After `nr` times iteratives, `nr` groups of estimation results of `b_12, b_1, b_2` will be stored in three vectors `v12, v1, v2`. Finally, outputs of mean and standard deviation of `nr` groups of `b_12, b_1, b_2` are calculated and stored in a solution file. Tolerance of fixed point iteration is $|(b_*^{(12)} - b^{(12)})/b^{(12)}| < 10^{-6}$ and $|(b_*^{(1)} - b^{(1)})/b^{(1)}| < 10^{-6}$ and $|(b_*^{(2)} - b^{(2)})/b^{(2)}| < 10^{-6}$. The process diagram is presented in Figure 4.1.

All the values of input parameters are set at the beginning of this application and will be modified as needed depending on the sizes of future portfolios of data.

4.2 Input Parameters

Input parameters of this application include:

1. Ip — first risk factor I
2. Jp — second risk factor J
3. Tp — time period T
4. m — collective mean m
5. s_2 — time variance s^2
6. b_12 — assumed starting value $b^{(12)}$
7. b_1 — assumed starting value $b^{(1)}$

8. b_2 — assumed starting value $b^{(2)}$

4.3 Output Requirement

The output of the application includes the file output and screen output. All the solutions are saved in the file `stdout.txt` or any output file. The solution file includes:

1. mb1 — mean of `nr` times `b_1` ;
2. sb1 — standard deviation of `nr` times `b_1`;
3. mb2 — mean of `nr` times `b_2` ;
4. sb2 — standard deviation of `nr` times `b_2`;
5. mb12 — mean of `nr` times `b_12` ;
6. sb12 — standard deviation of `nr` times `b_12`;

4.4 Simulation of Portfolios

Simulation of natural weights is composed of two steps [24]. In order to avoid large fluctuations in the size of contracts from one year to the next, an average weight $\bar{w}_{ij}$ is first created from a uniform distribution on [2, 10] for every contract; then simulation of annual contract weights is taken from a uniform distribution on $[0.5\bar{w}_{ij}, 1.5\bar{w}_{ij}]$.

Estimators $b_*^{(12)}, b_*^{(2)}, b_*^{(2)}$ are optimal under a zero-excess assumption for random effects. The simplest way to simulate loss ratios is to simulate random effects from

normal distributions:

$$\Xi_i^{(1)} \sim N(0, b^{(1)}),$$

$$\Xi_j^{(2)} \sim N(0, b^{(2)}),$$

$$\Xi_{ij}^{(12)} \sim N(0, b^{(12)}),$$

$$\Xi_{ijt}^{(123)} \sim N(0, s^2/w_{ijt}).$$

Then, loss ratios X_{ijt} are obtained by summing up the random effects and the collective mean.

4.5 Flow Chart of the Simulation Study on Variance Components Estimators

Figure 4.1 shows the application's flow chart. It begins with inputing data `I_p`, `J_p`, `T_p`, `m`, `s_2`, `b_12`, `b_1`, `b_2`, `nr`. If iterations are more than `nr`, the loop stops and calculates the mean and standard deviation of `b_12,b_1,b_2`. Otherwise, repeat the following process: simulate a new group of loss ratios `X_ijt` (see section 4.4); calculate a new group of weights w_ijt (see section 4.4); simulate structural parameter `s2` (see equation 2.3); calculate new estimation of `b_12,b_1,b_2` by using fixed point algorithm and store new results.

The process of calculation on estimation of `b_12,b_1,b_2` by using fixed point algorithm is as follows:

```
while (|(d12-b_12)/b_12|>=1e-6 or |(d1-b_1)/b_1|>=1e-6
       or |(d2-b_2)/b_2|>=1e-6)
{
  b_12=d12;
  b_1=d1;
  b_2=d2;

  B_12(b_1, b_2, b_12);
  B_1(b_1, b_2, d12);
  B_2(d1, b_2, d12);
}
```

4.6 Implementation Strategies

4.6.1 Application of Multiple Level Pointer

Based on the problem description of section 4.1, the simulation study requires a number of runs nr, and an estimation of variance components based on data X_{ijt} and w_{ijt}. In order to obtain accurate and stable results, we must select the value of nr at least 1000 for large data portfolio $I = 10$ and $J = 10$. Estimation of variance components requires not only large portfolios of data but also a fixed point iteration. Due to the complicated loop, the simulation from Goulet was only implemented on a

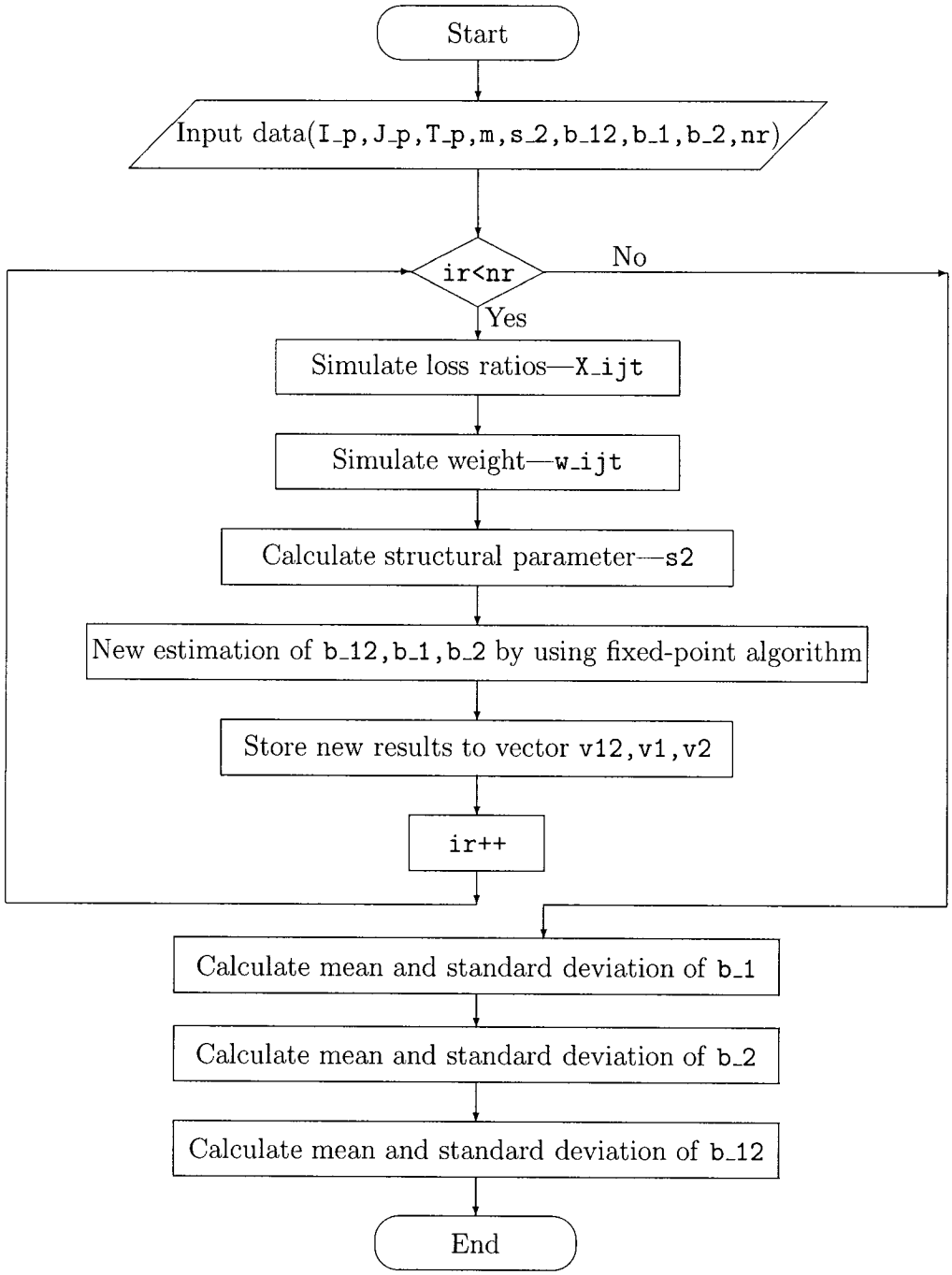

Figure 4.1: Simulation Flow Chart

small number of runs of 300 by using APL, S, and R language.

The main difficulty of this implementation is that it is restricted by memory. In C, pointers are one of the most versatile features and make it easier to allocate memory at run time using `malloc`. The application of pointers postpones the decision on size and the allocation time of the memory block, and, then, when that memory is no longer needed, it can be freed up for other uses. When memory is allocated, the allocating function returns a pointer. On an ANSI compliant compiler, `malloc` returns a void pointer. Because this application includes not only a one dimensional array but also multi-dimensional arrays, for example matrix C and matrix σ_{ijkl}, and even three dimensional arrays X_{ijt} and w_{ijt}, it increases the difficulty of the implementation. Therefore, we have to apply multiple level pointers in this application.

The method of dynamically allocating two-dimension array pointers is to create an array of pointers of type double and then allocate space for each row and point these pointers at each row. For example:

```
int nrows=200; ncols=200; row;
double **rowptr;


rowptr=malloc(nrows*sizeof (double *));
if (rowptr==NULL)
```

```
{
    printf("Failure to allocate room for row pointers.");
    exit(0);
}
for (row=0; row<nrows; row++)
{
    rowptr[row]=malloc(ncols * sizeof(double));
    if (rowptr[row]==NULL)
    {
      printf("Failure to allocate memory for row[%d]\n", row);
      exit(0);
    }
}
```

In the above code, `rowptr` is a pointer to a row of pointers of type double. It points to the first element of a vector of pointers of type double.

Pointers should be set and freed at appropriate points. In the implementation, the structure of setting and freeing pointers is as follows:

```
If (n < number of runs)
{
```

```
allocate memory for a 3D array X_ijt;

allocate memory for a 3D array w_ijt;

allocate memory for a 2D array X_ij.;

allocate memory for a 2D array w_ij.;

calculate X_ij.;

calculate w_ij.;

allocate memory for a vector X_12;

calculate X_12;

Free 3D memory X_ijt;

Free 3D memory w_ijt;


While( condition is satisfied )

{

  B_12( );

  { allocate memory for a 2D array A_12;

    Matrix_A_12( );

    { allocate memory for a 2D array C_12;

      allocate memory for a vector B_12;

      allocate memory for a 2D array Sigma_ijkl_12;

      Calculate Sigma_ijkl_12
```

```
    {allocate memory for a 2D array z_ij_12;
     Free 2D memory z_ij_12;
    }
    Matrix_B_12( );
    Matrix_C_12( );
    calculate A_12;
    Free 2D memory Sigma_ijkl_12;
  }
  calculate B_12;
  Free 2D memory C_12;
  Free vector B_12;
}
Free 2D memory A_12;
B_1( );
{ allocate memory for a vector X_1;
  alocate memory for a 2D array z_ij_12;
  calculate X_1;
  allocate memory for a 2D array A_1;
  Matrix_A_1( );
  { allocate memory for a 2D array C_1;
```

```
    allocate memory for a vector B_1;
    allocate memory for a 2D array Sigma_ik_1;
    Matrix_B_1( );
    Matrix_C_1( );
    calculate A_1;
    Free 2D memory Sigma_ik_1;
  }
  calculate B_1;
  Free 2D memory C_1;
  Free vector B_1;
}
Free 2D memory A_1;
Free vector X_1;
B_2( );
{ allocate memory for a vector X_2;
  allocate memory for a 2D array A_2;
  Matrix_A_2( );
  { allocate memory for a 2D array C_2;
    allocate memory for a vector B_2;
    allocate memory for a 2D array Sigma_jl_2;
```

```
        Matrix_B_2( );
        Matrix_C_2( );
        calculate A_2;
        Free 2D memory Sigma_jl_2;
      }
      calculate B_2;
      Free 2D memory C_2;
      Free vector B_2;
    }
    Free a vector X_2;
    Free 2D memory A_2;
    Free 2D memory z_ij_12;
  }
  Free 2D memory X_ij.;
  Free 2D memory w_ij.;
  Free a vector X_12;
}
```

4.6.2 Transformation of Data Structures

In this application, Matlab C library is used. The data structure used in Matlab is different from the C data structure. All the matrices used in Matlab C library must be defined by `mxArray`. Matrices constructed in C can not be used in Matlab C library. Therefore, before applying Matlab functions, all the data has to be transformed into a data structure which can be used by Matlab C library. If given a $n \times n$ pointer variable A, the transformation process is as follows:

```
double **A;
double *X;
mxArray *mat0=mxCreateDoubleMatrix(n,n,mxREAL);
X=vector(n*n);
TransMatrix(A,n,X);
memcpy(mxGetPr(mat0),X,n*n*sizeof (double));
```

4.6.3 Elimination of Redundant Calculations

In the computation of $b^{(1)}, b^{(2)}$, and $b^{(12)}$, many functions are repeated many times. Eliminating redundant calculation will increase the program running speed.

In the calculation of $b_*^{(12)}$, X is expressed as:

$$X = [X_{ijw} - X_{klw}]_{\frac{IJ(IJ-1)}{2} \times 1} ,$$

where,

$$X_{ijw} = \sum_{t=1}^{T} \frac{w_{ijt}}{w_{ij\Sigma}} X_{ijt}$$
$$w_{ij\Sigma} = \sum_{t=1}^{T} w_{ijt} .$$

If the numbers of risk categories I and J are fixed, simulation data X_{ijt} and w_{ijt} will not be changed during the iterative process of the fixed-point estimation of $b^{(12)}$. X will not change either. Therefore, the function of X in Figure 3.1 is calculated once before iterative circulation.

Another case is the computation of matrices B and C. They are expressed as:

$$B = [\sigma^{(12)}_{ijij} - 2\sigma^{(12)}_{ijkl} + \sigma^{(12)}_{klkl}]_{\frac{IJ(IJ-1)}{2}\times 1} ,$$
$$C = [(\sigma^{(12)}_{ijgh} - \sigma^{(12)}_{ijpq} - \sigma^{(12)}_{klgh} + \sigma^{(12)}_{klpq})^2]_{\frac{IJ(IJ-1)}{2}\times\frac{IJ(IJ-1)}{2}} ,$$

and $\sigma^{(12)}_{ijkl}$ is

$$\sigma^{(12)}_{ijkl} = \delta_{ik}b^{(1)} + \delta_{jl}b^{(2)} + \delta_{ij,kl}\frac{b^{(12)}}{z^{(12)}_{ij}} .$$

The value of $z^{(12)}_{ij}$ will be changed in every new iterative process with new $b^{(12)}$ input value. If the value of matrices B and C are obtained by calling function $\sigma^{(12)}_{ijkl}$ in every σ calculation, the total calling times in one iterative process are $(IJ)^2(IJ-1)^2 + 3/2IJ(IJ-1)$.

In order to reduce redundant calculations of $\sigma^{(12)}_{ijkl}$, we calculate and store all values of the $\sigma^{(12)}_{ijkl}$ in a $IJ \times IJ$ matrix. The only thing left to do in the calculation of

matrices B and C is to withdraw the value of $\sigma_{ijkl}^{(12)}$ directly from the matrix. This will reduce computing times to $(IJ)^2$, which is obviously smaller than $(IJ)^2(IJ-1)^2 + 3/2IJ(IJ-1)$.

This method is similarly applied to calculation of matrices $B^{(1)}, C^{(1)}$ and $B^{(2)}, C^{(2)}$.

Chapter 5

Results of the Simulation Study

5.1 Results and Comparison with Dannenburg's Estimators

The main goal of the simulation study consists in comparing the performance of the optimal variance components estimators with those proposed by Dannenburg, both in terms of bias and variance. The following subsections present the results of a number of simulations with different input parameters.

Note that all simulations were carried out with $m = 5$, $s^2 = 5$, and $T = 5$. Column Parameter represents the input value of $b^{(1)}$, $b^{(2)}$, and $b^{(12)}$; column Estimation shows there are two groups of results, Dannenburg's results and Optimal estimation results; column Average represents the mean of N runs of $b^{(1)}$, $b^{(2)}$, and $b^{(12)}$; column Std. Dev. presents the standard deviations of N runs of $b^{(1)}$, $b^{(2)}$, and $b^{(12)}$; column Coef.of Var. represents coefficient of variation (which is obtained by dividing standard deviation by mean); the last column Negative represents the number of negative estimates of

$b^{(1)}$, $b^{(2)}$, and $b^{(12)}$. Simulation of Dannenburg's estimators is provided by Goulet.

5.1.1 Simulation Results for $I = J = 4$

Results of two simulations — Dannenburg's solutions and optimal estimation's solutions for small portfolio $I = 4, J = 4$, using 1000 iterations are summarized in Tables 5.1—5.3. Results from $I = 4, J = 4$, using 5000 iterations are summarized in Tables 5.4—5.6.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.5227	0.6509	1.2452	186
	Optimal	0.5702	0.6459	1.1328	0
$b^{(2)} = 0.2$	Dannenburg's	0.2073	0.3793	1.8300	321
	Optimal	0.2345	0.3429	1.4623	0
$b^{(12)} = 0.7$	Dannenburg's	0.7102	0.4383	0.6171	9
	Optimal	0.6087	0.3886	0.6384	0

Table 5.1: Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 4$, 1000 *runs*.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2$	Dannenburg's	2.0226	2.4613	1.2169	184
	Optimal	2.2100	2.4514	1.1092	0
$b^{(2)} = 1.5$	Dannenburg's	1.4703	1.8620	1.2664	212
	Optimal	1.4870	1.8164	1.2215	0
$b^{(12)} = 3$	Dannenburg's	2.9299	1.5276	0.5214	4
	Optimal	2.7547	1.3788	0.5005	0

Table 5.2: Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 4$, 1000 *runs*.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	10.7268	13.7443	1.2813	192
	Optimal	11.2197	13.0408	1.1623	0
$b^{(2)} = 15$	Dannenburg's	16.1401	17.9104	1.1097	145
	Optimal	14.4622	15.7705	1.0905	0
$b^{(12)} = 20$	Dannenburg's	20.0371	10.9447	0.5462	3
	Optimal	18.5996	8.5992	0.4623	0

Table 5.3: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 4$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.4938	0.5991	1.2132	945
	Optimal	0.5327	0.5955	1.1179	0
$b^{(2)} = 0.2$	Dannenburg's	0.2025	0.3769	1.8612	1648
	Optimal	0.2489	0.3276	1.3162	0
$b^{(12)} = 0.7$	Dannenburg's	0.7021	0.4309	0.6137	76
	Optimal	0.6192	0.3776	0.6098	0

Table 5.4: Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 4$, 5000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2$	Dannenburg's	1.9952	2.3413	1.1735	857
	Optimal	2.0714	2.2864	1.1038	0
$b^{(2)} = 1.5$	Dannenburg's	1.5339	1.9837	1.2932	1089
	Optimal	1.5795	1.7793	1.1265	0
$b^{(12)} = 3$	Dannenburg's	2.9888	1.5591	0.5216	16
	Optimal	2.8132	1.3649	0.4852	0

Table 5.5: Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 4$, 5000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	10.1854	13.3027	1.3061	1090
	Optimal	10.5528	12.1769	1.1539	0
$b^{(2)} = 15$	Dannenburg's	14.8404	16.9111	1.1395	800
	Optimal	15.2494	15.7148	1.0305	0
$b^{(12)} = 20$	Dannenburg's	19.9136	9.8576	0.4950	18
	Optimal	19.0231	8.5927	0.4517	0

Table 5.6: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 4$, 5000 *runs*.

5.1.2 Simulation Results for $I = J = 6$

Results of the two simulations for larger portfolios ($I = 6, J = 6$), 1000 runs and 5000 runs are summarized in Tables 5.7—5.9 and 5.10—5.12, respectively.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.5152	0.4227	0.8205	50
	Optimal	0.5009	0.4364	0.8712	0
$b^{(2)} = 0.2$	Dannenburg's	0.1914	0.2327	1.2158	210
	Optimal	0.2126	0.2316	1.0894	0
$b^{(12)} = 0.7$	Dannenburg's	0.7030	0.2572	0.3659	0
	Optimal	0.6732	0.2509	0.3727	0

Table 5.7: Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 6$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2$	Dannenburg's	2.0903	1.8021	0.8621	54
	Optimal	1.9910	1.6743	0.8409	0
$b^{(2)} = 1.5$	Dannenburg's	1.4793	1.3412	0.9067	85
	Optimal	1.5179	1.3245	0.8726	0
$b^{(12)} = 3$	Dannenburg's	3.0331	0.9434	0.3110	0
	Optimal	2.9521	0.8747	0.2963	0

Table 5.8: Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 6$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	9.9695	8.8406	0.8868	77
	Optimal	10.0032	8.7394	0.8737	0
$b^{(2)} = 15$	Dannenburg's	15.2329	12.4135	0.8149	32
	Optimal	15.1179	11.8842	0.7861	0
$b^{(12)} = 20$	Dannenburg's	19.9885	6.1145	0.3059	0
	Optimal	19.7804	5.5162	0.2789	0

Table 5.9: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 6$, 1000 *runs*.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.4992	0.4236	0.8485	272
	Optimal	0.5125	0.4302	0.8394	0
$b^{(2)} = 0.2$	Dannenburg's	0.2021	0.2303	1.1395	881
	Optimal	0.2159	0.2227	1.0315	0
$b^{(12)} = 0.7$	Dannenburg's	0.6986	0.2554	0.3656	0
	Optimal	0.6696	0.2493	0.3723	0

Table 5.10: Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 6$, 5000 *runs*.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2$	Dannenburg's	1.9911	1.6308	0.8191	268
	Optimal	2.0248	1.6571	0.8184	0
$b^{(2)} = 1.5$	Dannenburg's	1.5053	1.3373	0.8884	415
	Optimal	1.5296	1.2886	0.8424	0
$b^{(12)} = 3$	Dannenburg's	2.9809	0.9424	0.3162	0
	Optimal	2.9567	0.8844	0.2991	0

Table 5.11: Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 6$, 5000 *runs*.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	9.8907	8.6326	0.8728	413
	Optimal	10.1469	8.6795	0.8554	0
$b^{(2)} = 15$	Dannenburg's	15.0191	11.9356	0.7947	205
	Optimal	15.1839	11.6267	0.7657	0
$b^{(12)} = 20$	Dannenburg's	20.1642	6.0224	0.2987	0
	Optimal	19.8836	5.5951	0.2814	0

Table 5.12: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 6$, 5000 *runs*.

5.1.3 Simulation Results for $I = J = 8$

Results of the two simulations for portfolios with $I = 8, J = 8$, 1000 runs are presented in Tables 5.13—5.15.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.5247	0.3461	0.6597	12
	Optimal	0.5149	0.3477	0.6753	0
$b^{(2)} = 0.2$	Dannenburg's	0.2014	0.1685	0.8369	80
	Optimal	0.1989	0.1617	0.8130	0
$b^{(12)} = 0.7$	Dannenburg's	0.6992	0.1840	0.2632	0
	Optimal	0.6842	0.1850	0.2704	0

Table 5.13: Comparison of Dannenburg & Optimal variance components estimators for small values parameters, $I = J = 8$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2$	Dannenburg's	2.0717	1.3093	0.6320	7
	Optimal	2.0468	1.3461	0.6577	0
$b^{(2)} = 1.5$	Dannenburg's	1.5564	1.0660	0.6849	25
	Optimal	1.4800	0.9956	0.6727	0
$b^{(12)} = 3$	Dannenburg's	2.9876	0.6741	0.2256	0
	Optimal	2.9891	0.6480	0.2168	0

Table 5.14: Comparison of Dannenburg & Optimal variance components estimators for moderate values parameters, $I = J = 8$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	9.9008	6.7497	0.6817	15
	Optimal	10.2293	6.9703	0.6814	0
$b^{(2)} = 15$	Dannenburg's	14.9519	9.5413	0.6381	11
	Optimal	14.8991	9.2592	0.6215	0
$b^{(12)} = 20$	Dannenburg's	19.8063	4.2700	0.2156	0
	Optimal	20.0765	4.0773	0.2031	0

Table 5.15: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 8$, 1000 *runs.*

5.1.4 Simulation Results for $I = J = 10$

The results of portfolios with $I = J = 10$, 1000 runs are presented in Tables 5.16—5.18.

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 0.5$	Dannenburg's	0.5069	0.2895	0.5711	2
	Optimal	0.4888	0.2743	0.5612	0
$b^{(2)} = 0.2$	Dannenburg's	0.2076	0.1421	0.6846	27
	Optimal	0.1967	0.1413	0.7184	0
$b^{(12)} = 0.7$	Dannenburg's	0.6952	0.1404	0.2019	0
	Optimal	0.6887	0.1540	0.2236	0

Table 5.16: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.*

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 2.0$	Dannenburg's	2.0287	1.0866	0.5356	1
	Optimal	1.9446	1.0753	0.5530	0
$b^{(2)} = 1.5$	Dannenburg's	1.5062	0.9075	0.6025	2
	Optimal	1.4804	0.8823	0.5960	0
$b^{(12)} = 3.0$	Dannenburg's	3.0142	0.5202	0.1726	0
	Optimal	2.9950	0.5263	0.1757	0

Table 5.17: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.*

5.2 Graphical Comparison of Standard Deviations

Graphs are used to compare two kinds of results (Optimal and Dannenburg's). The following graphs are based on the data: $T = 5$, $m = 5$, $s^2 = 5$, 1000 runs. The X axis

Parameter	Estimator	Average	Std. Dev.	Coef. of Var.	Negative
$b^{(1)} = 10$	Dannenburg's	10.1898	5.7772	0.5670	5
	Optimal	9.6724	5.5526	0.5741	0
$b^{(2)} = 15$	Dannenburg's	14.8041	8.2300	0.5559	0
	Optimal	14.8712	8.2359	0.5538	0
$b^{(12)} = 20$	Dannenburg's	19.9223	3.3213	0.1667	0
	Optimal	20.0901	3.2366	0.1611	0

Table 5.18: Comparison of Dannenburg & Optimal variance components estimators for large values parameters, $I = J = 10$, 1000 *runs.*

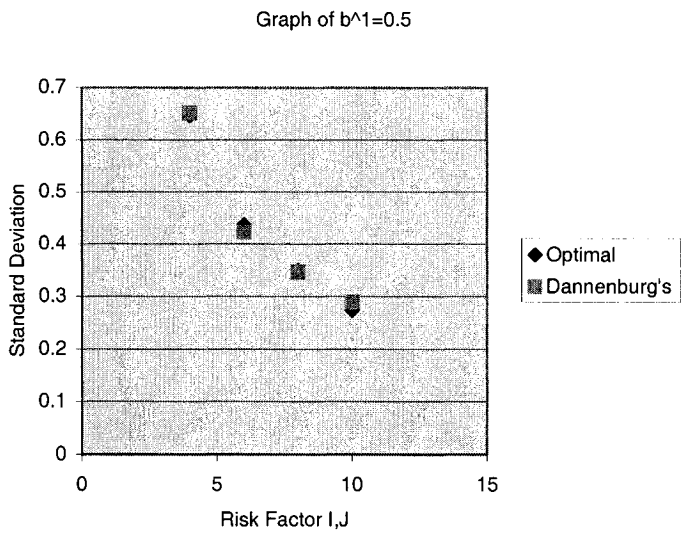

Figure 5.1: Graphs of Small Values Parameters_ $b^{(1)}$

represents I and J. I, J are 4, 6, 8, 10 respectively. The Y axis represents standard deviation. Due to similar results between the two simulations, some points overlap a little bit. The detailed results can be found in the above section.

The first group of graphs present comparisons of small values parameters, $b^{(1)} = 0.5, b^{(2)} = 0.2, b^{(12)} = 0.7$.

See Figure 5.1—5.3: Graphs of Small Values Parameters

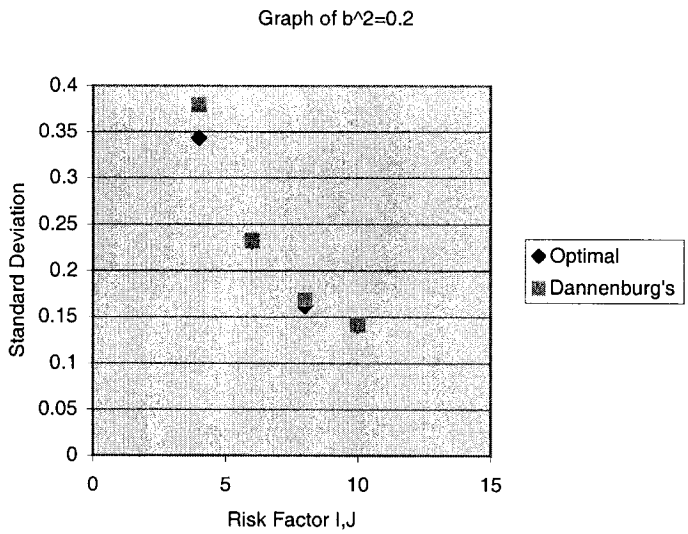

Figure 5.2: Graphs of Small Values Parameters_ $b^{(2)}$

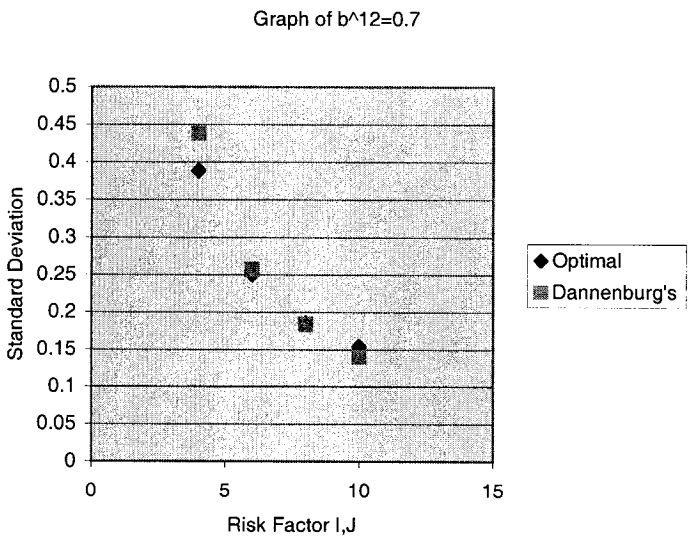

Figure 5.3: Graphs of Small Values Parameters_ $b^{(12)}$

The second group of graphs present comparisons of moderate values parameters, $b^{(1)} = 2, b^{(2)} = 1.5, b^{(12)} = 3$.

See Figure 5.4—5.6: Graphs of Moderate Values Parameters

The third group of graphs present comparisons about large values parameters, $b^{(1)} = 10, b^{(2)} = 15, b^{(12)} = 20$.

See Figure 5.7—5.9: Graphs of Large Values Parameters

5.3 Conclusions

From the results of the numerical test, optimal estimators are reasonably unbiased.

For $I = J = 4$, $T = 5$, runs 1000 or runs 5000, optimal standard deviations are obviously smaller than Dannenburg's standard deviations. Coefficients of Variation in the optimal model are apparently smaller than those in Dannenburg's model.

For $I = J = 6$, $T = 5$, runs are 1000, standard deviation of $b^{(1)}$=0.5 is a little larger than Dannenburg's. When runs are 5000, standard deviations of $b^{(2)}, b^{(12)}$ in the optimal model are smaller than those in Dannenburg's model; optimal standard deviation of $b^{(1)}$ is a little bit larger than that in Dannenburg's model. One optimal coefficient of variation is a little larger than Dannenburg's.

With I, J increasing to 8, runs 1000, optimal standard deviation of $b^{(1)}$ is a little bit

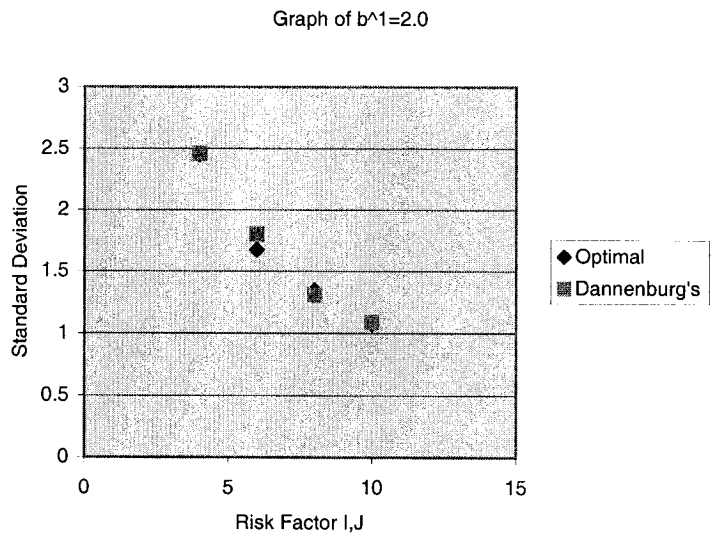

Figure 5.4: Graphs of Moderate Values Parameters_ $b^{(1)}$

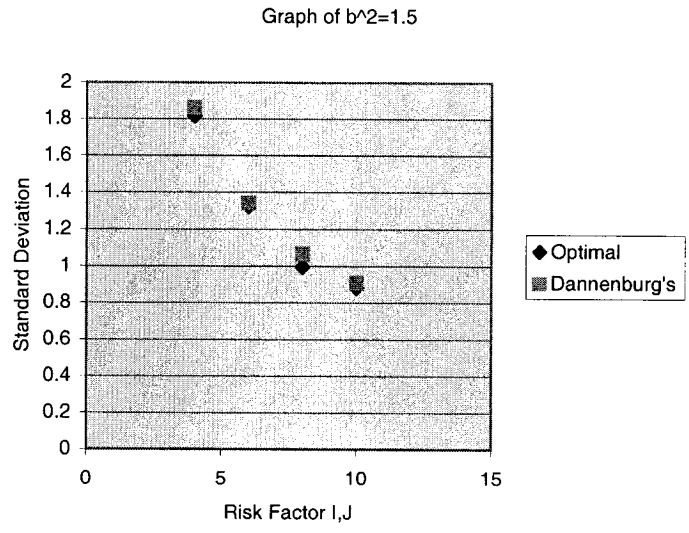

Figure 5.5: Graphs of Moderate Values Parameters_ $b^{(2)}$

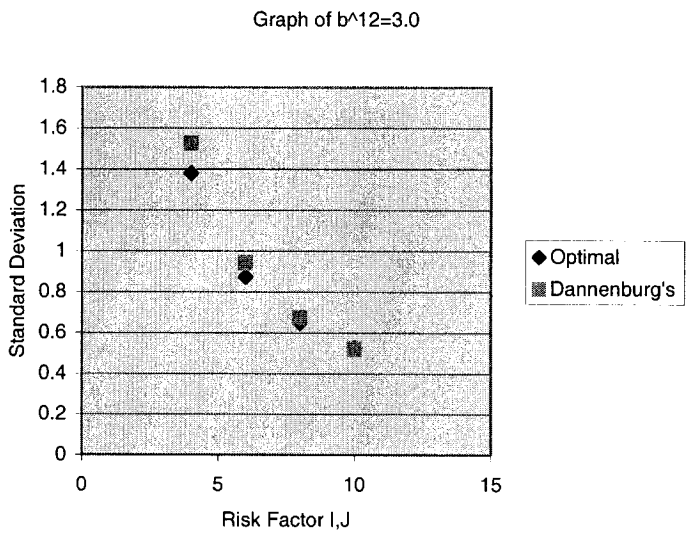

Figure 5.6: Graphs of Moderate Values Parameters_ $b^{(12)}$

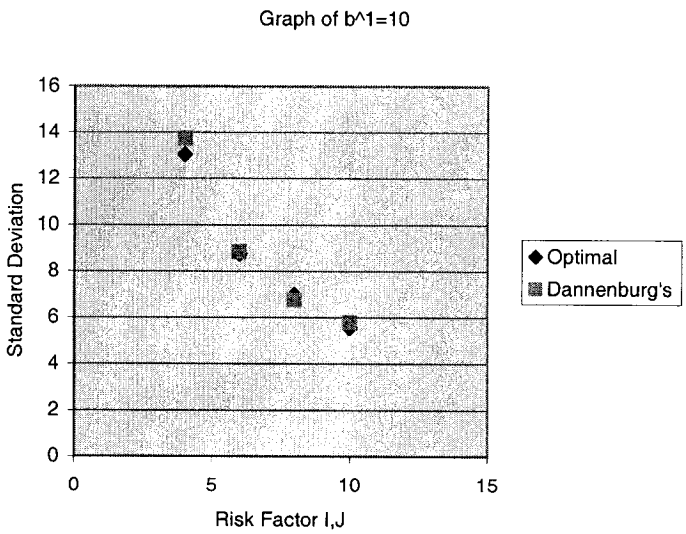

Figure 5.7: Graphs of Large Values Parameters_ $b^{(1)}$

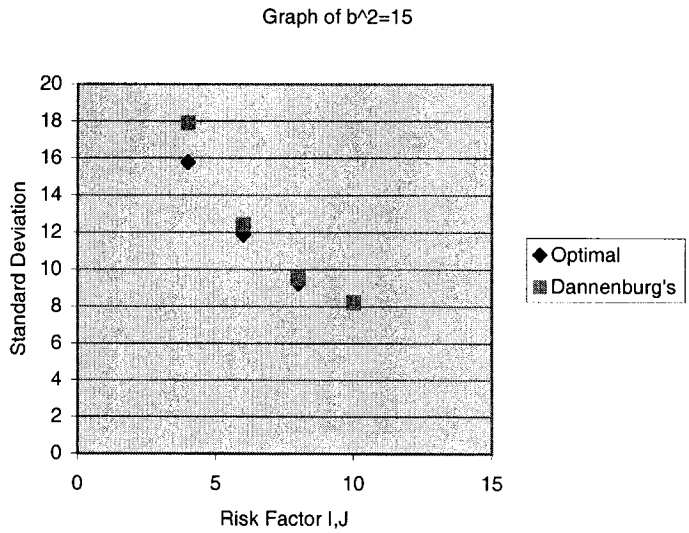

Figure 5.8: Graphs of Large Values Parameters_ $b^{(2)}$

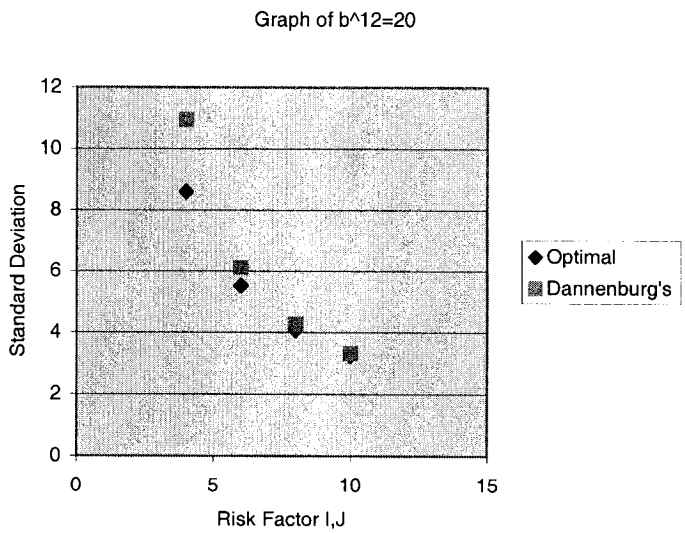

Figure 5.9: Graphs of Large Values Parameters_ $b^{(12)}$

larger than that in Dannenburg's model. Some optimal coefficients of variation are larger than Dannenburg's.

When I, J equal 10, runs 1000, most optimal standard deviations are smaller than Dannenburg's standard deviations. Some optimal coefficients of variation are smaller than Dannenburg's.

Solutions show that in some cases, optimal means are closer to real parameters than Dannenburg's means. The numbers of negative value from Dannenburg's estimators decreases as I, J increases.

The results of the numerical test demonstrate that most optimal estimators have smaller variances so that the premium calculation from the CCC model yields a more accurate value than Dannenburg's estimators, although, in some cases, a small number of optimal solutions are a little larger than those in Dannenburg's model.

Conclusions and Future Work

The purpose of this thesis was to design and implement a numerical evaluation of optimal variance components estimators $b^{(1)}, b^{(2)}, b^{(12)}$ in a two-way crossed classification credibility model by using an appropriate programming language, and effective tools and libraries so as to implement a simulation software package capable of performing computation faster and more efficiently. This goal was achieved by analyzing existing problems encountered in realizing the simulation study of the variance components estimators, using computing language C in the simulation, combining the Matlab C library and the R math library, applying fast calculation algorithms, allocating limited memory spaces, increasing computing speed and implementing the simulation's requirement efficiently.

As a theoretical foundation of this simulation study, a two-way crossed classification credibility model was described in detail. From the credibility equation, it is easy to see that estimations of structural parameters are very important in order to yield the closest estimation solutions to the insureds' risk premiums. This is reason why research of credibility theory has focused on the estimation and optimality of struc-

ture parameters since the 1980s. In Dannenburg's crossed classification credibility model, he applied the variance components model in the credibility model to realize the estimation of the structure parameters $s^2, b^{(i_1,i_2,\ldots,i_p)}$ and to develop tests concerning the values of these parameters. However, Dannenburg's estimation has no known optimality property except unbiasedness. In chapter 2, under the assumptions that all random effects have no excess and projections onto translated linear subspaces of the Hilbert space, Goulet applied the minimum variance method in variance components estimators to optimize the estimations. The variance components estimators in Dannenburg have a strong tendency to yield negative estimates. Goulet uses the fixed-point method to calculate optimal pseudo-estimators. The iterative procedure always converges to a unique and non-negative fixed-point independent of the starting values. Meanwhile, the convergence of the iterative procedure is accelerated by Siedel's method.

The simulation study of optimal variance components for small portfolios of data has been implemented before. However, calculation of the optimal variance components estimators for large portfolios is slow and computationally intensive. Although many tools and several programming languages have been used to ease mathematical calculation so that the developing time is reduced to a minimum, they are still ineffective.

In chapter 3, the problems were analyzed in detail and resolved. Language C was used

in this simulation since it is a low level and hence powerful language for computing; moreover, the standard version is portable. Application of pointers resolved the problem of memory allocation. All the matrices and vectors used on pointer variables are allocated memory when required, which are freed later for reuse. The simulation was done on the Linux system so that it could be integrated with other codes running on Linux.

Fixed-point algorithm and large matrix inversion are computationally taxing, and, therefore, increase execution time. Since the inversion of matrix C is required three times in the program, each calculating $C^{-1}B$, which is equivalent to the computing value of a group of linear equations. Application of LU decomposition algorithm eases computation by resolving the value of linear equations. Lapack Chelosky algorithm was used to accelerate the speed of execution on linear equations since Matrix C is symmetric positive definite. Function backslash of Matlab C library applies the Lapack Chelosky algorithm and plays an important role in increasing software processing speed. According to our speed tests, for risk factors I=10, J=10, T=5, execution time decreased to 9 minutes from 22 minutes. Tests demonstrate that the algorithm and functions used in this simulation are optimal. Accuracy tests demonstrate that solutions obtained from the simulation are validated with the solutions from Goulet's codes.

Chapter 4 describes the software implementation of evaluating estimators of the opti-

mal variance components in a two-way crossed classification credibility model. A flow chart was given to show the entire process of this program. In this flow chart, input parameters and output requirements are clearly described. Implementation strategies analyses were introduced in order to clearly explain the implementation difficulties and problem-solving method.

Chapter 5 presents the results of means and standard deviations of different size parameters $b^{(1)}, b^{(2)}, b^{(12)}$ on large portfolios. Compared with Dannenburg's structure parameters estimation, numerical tests prove that Goulet's optimal estimators have smaller variances. Minimum variance unbiased estimation method optimizes variance components estimators which brings the expectation of future insurance premiums closer to accurate values.

Although the simulation improved the execution speed for a large portfolio dramatically, the execution time is still considerable. For example, I=10, J=10, T=5, $b^{(1)} = 1.0$, $b^{(2)} = 1.0$, $b^{(12)} = 1.0$, m=5.0, $s^2 = 5.0$, runs=5000, one iterative process takes 9 minutes. Total time for execution is $5,000 \times 9$=45,000 mins=750 hours=31 days. This means that we have to wait for one month to obtain this group of solutions. So far, numerical tests indicate that running speed is different on different systems. It is thus necessary either to continue looking for more effective algorithms and matrix calculation software, or to try to execute on a faster computer system in order to reduce computation time.

From the point of view of mathematics, realization of optimal estimation depends on fixed point computation. Although the calculation tends to converge to a unique fixed point independent of the starting values, the number of iterations can not be known until termination of the calculation. This kind of unknown iterative runs waste a lot of computer resources and time. It is thus worth doing further research on whether fixed point estimation algorithm could be replaced by other more efficient computing algorithms. Certainly, more efficient optimal parameter estimation methods in the crossed classification credibility model are waiting to be developed.

At the outset of this research, the application employed in the present simulation is required to integrate with a R code. However, this simulation study demonstrates a C program with libraries is difficult to connect with R. In theory, it is possible to apply libraries which have been already used and integrated with R in a C code. We have tried to create a package which includes C code with `R_ext/Lapack.h` and installed this package in R so as to implement the integration. However, we could not resolve the problems that arose during this experiment. Further research on integrating C with R should, therefore, be carried out in the future.

Bibliography

[1] Bailey, A.L. (1945), "A Generalized Theory of Credibility". *Proceedings of the Casualty Actuarial Society* 32: 13-20.

[2] Bailey, A.L. (1950), "Credibility Procedures, Laplace's Generalization of Bayes' Rule and the Combination of Collateral Knowledge with Observed Data". *Proceedings of the Casualty Actuarial Society* 37: 7-23.

[3] Bühlmann, H. (1967), "Experience Rating and Credibility". *ASTIN Bulletin*,4, 199-207.

[4] Bühlmann, H. and Straub, E. (1970), "Glaubgwürdigkeit für Schadensätze". *Bulletin of the Swiss Association of Actuaries* 70, 111-133. English translation by C.E.Brooks.

[5] Bühlmann, H. (1969), "Experience Rating and Credibility". *ASTIN Bulletin*,5, 157-165.

[6] Burden, R.L. and Faires, J.D. (1993), *Numerical Analysis*, fifth edition, PWS-Kent, Pub.Co. Boston.

[7] Dannenburg, D. (1995), "Crossed Classification Credibility Models". *Transactions of the 25th International Congress of Actuaries* 4, 1-35.

[8] De Vylder, F. and Goovaerts, M.J. (1991), "Estimation of the Heterogeneity Parameter in the Bühlmann-Straub Credibility Theory Model". *Insurance: Mathematics and Economics* 10:233-238.

[9] De Vylder, F. (1978), "Parameter Estimation in Credibility Theory". *ASTIN Bulletin* 10: 99-112.

[10] De Vylder, F. (1981), "Practical Credibility Theory with Emphasis on Parameter Estimation". *ASTIN Bulletin* 12:115-131.

[11] De Vylder, F. (1984), "Practical Model in Credibility Theory, Including Parameter Estimation". *Premium Calculation in Insurance.* NATO ASI Series: 133-150.

[12] De Vylder, F. and Goovaerts, M.J. (1992), "A Summary of New Results on Optimal Parameter Estimation under Zero-excess Assumptions". *Insurance: Mathematics and Economics* 11: 153-161. North-Holland.

[13] De Vylder, F. and Goovaerts, M.J. (1992), "Optimal Parameter Estimation under Zero-excess Assumptions in a Classical Model". *Insurance: Mathematics and Economics* 11: 1-6. North-Holland.

[14] De Vylder, F. and Goovaerts, M.J. (1991), "Estimation of the Heterogeneity Parameter in the Bühlmann-Straub Credibility Theory Model". *Insurance: Mathematics and Economics* 10:233-238.

[15] Dongarra, J.J., DuCroz, J., Hammarling, S., Hanson, R. (1988), "An Extended Set of Fortran Basic Linear Algebra Subprograms", *ACM Transactions on Mathematical Software*, Vol.14, No.1, 1-32.

[16] Dongarra, J.J., Duff, I., DuCroz, J., Hammarling, S. (1990), "A set of Level 3 Basic Linear Algebra Subprograms", *ACM Transactions on Mathematical Software*, Vol.16, No.1, 1-32.

[17] Dubey, A. and Gisler, A. (1981), "On Parameter Estimation in Credibility". *Bulletin of the Swiss Association of Actuaries* 81: 187-211.

[18] Friedrich Leisch. (2004), "Writing R Extension". *The R Manuals*, version 1.9.1, edited by the R Development Core Team. http://cran.r-project.org/manuals.html.

[19] Gisler, A. (1980), "Optimal Trimming of Data in the Credibility Model". *Bulletin of the Swiss Association of Actuaries* 80: 313-325.

[20] Gisler, A. and Reinhard, P. (1993), "Robust Credibility". *ASTIN Bulletin*23: 117-143.

[21] Goovaerts M.J., Kaas R., Van Heerwaarden A.E., Bauwelinckx T.(1990), *Effective Actuarial Methods*, Elsevier Science Pub.B.V. Netherlands.

[22] Goovaerts, M.J. and Hoogstad, W.J. (1987), *Credibility Theory.* Surveys of Actuarial Studies, No.4.Rotterdam, Holland: Nationale-Nederlanden N.V.

[23] Goulet Vincent. (1998), "Principles and Application of Credibility Theory". *Journal of Actuarial Practice.* Volume 6, No.1 and 2.

[24] Goulet Vincent. (2001), "Optimal Parameter Estimation in Crossed Classification Credibility Theory". Concordia University, Montreal, Canada.

[25] Goulet Vincent. (1998), "A Note on Optiaml Parameter Estimation under Zero-excess Assumptions". *Insurance: Mathematics and Economics* 23:111-117.

[26] Hachemeister, C.A. (1975), "Credibility for Regression Models with Application to Trend". *Credibility, Theory and Applications, Proceedings of the Berkeley Actuarial Research Conference on Credibility.* New York, N.Y.:Academic Press: 129-163.

[27] Jewell, W.S. (1975), "The Use of Collateral Data in Credibility Theory: A Hierarchical Model". *Giornale dell'Istituto Italiano degli Attuari* 38: 1-16.

[28] Kahn, P.M. (1974), "Credibility Theory and Application". *Proceedings of the Berkeley Actuarial Research Conference on Credibility, September*, 19-21.

[29] Klugman Stuart A., Panjer Harry H., Willmot Gordon E. (1998), *Loss Model-From Data to Decision*, John Wiley and Sons, INC.

[30] Künsch, H.R. (1992), "Robust Methods for Credibility". *ASTIN Bulletin* 22: 33-49.

[31] Lawson, C., Hanson, R., Kincaid, D., Krogh, F. (1979), "Basic Linear Algebra Subprograms for Fortran Usage". *ACM Transactions on Mathematical Software*, Vol.5, 308-325.

[32] Mowbray, A.H. (1914), "How Extensive a Payroll Exposure is Necessary to Give a Dependable Pure Premium?". *Proceedings of the Casualty Actuarial Society* 1, 25-30.

[33] Natacha Bereux. (2003), "A Cholesky Algorithm for Some Complex Symmetric".

[34] Norberg, R. (1986), "Hierarchical Credibility: Analysis of a Random Effects linear Model with Nested Classification". *Scandinavian Actuarial Journal*: 204-222.

[35] Norberg, R. (1980), "Empirical Bayes Credibility". *Scandinavian Actuarial Journal*: 177-194.

[36] Press. William H. (1992), *Numerical recipes in C: the art of scientific computing*, 2nd edition, Cambridge [England] ; New York, NY, USA : Cambridge University Press.

[37] Searle, S.R., Casella, G. and McCulloch, C.E. (1992), *Variance Components.* New York, N.Y.:Wiley.

[38] Venables, W.N. and Ripley, B.D. (2000), *S programming.* Springer.

[39] Whitney, A.W. (1918), "The Theory of Experience Rating". *Proceeding of the Casualty Actuarial Society* 4: 275-293.

[40] The MathWorks, Inc. (2000), *Matlab C Math Libray User's Guide*, version 2.1.

[41] The R Development Core Team. (2003), "R Installation and Administration". *The R Manuals*, version 1.6.2.

[42] The R Development Core Team. (2004), "R Installation and Administration". *The R Manuals*, version 2.0.1.

[43] http://www.cms.livjm.ac.uk/etchells/notes/ma200/jacobi1.pdf

[44] http://www.netlib.org/lapack

[45] http://www.mathworks.com/products/matlab

[46] http://www.r-project.org/