Practicum: The C-S method in computation of the logistic normal integral

by

© *Yi Zhao*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Applied Statistics*

Department of *Mathematics and Statistic*

Memorial University of Newfoundland

May 2014

St. John's Newfoundland

Abstract

In logistic regression models with measurement error involved, the likelihood function is often taking the form as the logistic normal integral. Among several methods that have been proposed to compute this kind of integral, the method by E. Crouch and D. Spiegelman (1990) (C-S), programmed in FORTRAN, is believed to be a good candidate from the computational perspective. We investigate this method by calling the FORTRAN code into R, and compare its performance with the classic Gaussian Quadrature method. The simulation results show that the C-S approach is much faster than the classic Gaussian Quadrature algorithm without losing any precision of the estimates, especially when the sample size is large.

Acknowledgements

I sincerely appreciate E. Crouch and D. Spiegelman, who provided the FORTRAN code, as well as those who have helped me on achieving this report, especially my supervisor, Dr. Fan and Dr. Ye, who dedicate themselves on guiding and teaching me all the time.

Contents

Abstract ii

Acknowledgements iii

List of Tables vi

List of Figures vii

1 Introduction 1

1.1 Logistic Regression 2

1.2 Measurement Error Model 3

1.2.1 Systematic and Random Error 4

1.2.2 Additive and Multiplicative Error 4

1.2.3 Differential and Nondifferential Error 5

1.2.4 Classical, Berkson and Mixed Error 5

1.3 Model Building 7

1.3.1 Constructing the Likelihood 8

1.3.2 Measurement Error Model Selection 9

1.3.3 Our Model 9

2 Computing logistic-normal integral **11**

2.1 Probit Approximation 11

2.1.1 One-probit approximation 12

2.1.2 Two-probit approximation 13

2.2 Laplace Approximation 13

2.3 Gaussian Quadrature 15

2.4 Quasi-Monte Carlo Method 16

2.5 The C-S Method 17

3 Maximization of the Likelihood **20**

4 Calling J(x,y;n) in R **25**

5 Simulation Study and Comparison with Gaussian Quadrature **30**

5.1 Effect of Error and Variance 31

5.2 Integral Computing and Comparison 32

5.3 Parameter Estimation 34

6 Conclusions **37**

Bibliography **38**

A The original Fortran code of C.S method **41**

B Abscissas and weight factors for Hermite Integration **45**

C Iteration R code example **47**

List of Tables

5.1 Comparison of two methods on approximating integral 33

5.2 Result comparison of two methods n=500 35

5.3 Result comparison of two methods n=1000 35

List of Figures

Chapter 1

Introduction

In the application of regression models, a reliable inference procedure about the regression coefficients is always expected. Extreme care needs to be taken of the estimation methods for the regression parameters, especially when the covariates are prone to the measurement errors. The estimates of unknown parameters could be biased and hence lead to misleading conclusions if the measurement errors are not appropriately treated.

Measurement error in variables also exist in logistic regression models, which is a special case of the generalized linear models. If we assume that the measurement error is normally distributed, we could model the effect of the measurement error by using the joint distribution of the errors with the random response and then obtain the likelihood function through the logistic-normal integral. The regression parameters can then be consistently estimated by using the likelihood method.

It is difficult to compute the exact value of likelihood function, which is known as the logistic-normal integral. Several methods were proposed in literature to approximate this kind of integral, see Goodwin (1949), Crouch and Spiegelman (1990), E.

Demidenko (2004), among others.

In this thesis, we compare the performance of some of the methods from the computational perspective. Focus has been put on the comparison between the C-S approximation and the Gaussian quadrature algorithm. Some background knowledge is reviewed as the foundation of our study.

1.1 Logistic Regression

Logistic regression models belong to the family of generalized linear models. They are developed to model the relationship between a binary response and a set of covariates. Logistic regression has been widely applied in social, medical, economical and many other fields. As opposed to the linear regression, logistic regression has a discrete response. In linear regression, the conditional mean response is given by $\beta_0 + \beta X$; in logistic regression, the conditional mean response function, which also equals to the probability of success, is nonlinear and defined as

$$E(Y|x) = \frac{e^x}{1+e^x}, \tag{1.1}$$

to guarantee it is always positive and less than 1. In the previous equation, x is the covariate that we observed and the mean response takes the value between 0 and 1. If we consider the regression part, the model becomes

$$\pi = \frac{e^{\beta_0+\beta' X}}{1+e^{\beta_0+\beta' X}}, \tag{1.2}$$

where $(\beta_0, \beta)'$ is the parameter vector and X is the covariate vector. Meanwhile, π takes values from the interval $[0, 1]$. The key part in logistic regression is the logit transformation denoted by L, which takes the form of

$$L = \ln \frac{\pi}{1-\pi} = \beta_0 + \beta' X \tag{1.3}$$

The advantage of logit transformation is that L has many similar properties of linear regression model. The challange about logistic regression is that the response follows a binary distribution.

1.2 Measurement Error Model

Measurement errors occur when we are not able to collect the accurate data, due to some financial and instrumental limitations. The measurement errors can occur on both responses and covariates. However, we only consider the situation where the covariate measurement error occurs. For instance, if we consider a regression with the response Y observed, the covariates can be divided into two groups: one is those we are able to measure accurately (Z), which means that Z does not have error involved; the other part is those we do not observe accurately (X). Instead of observing X, we observe a variable W related to X. Then, we fit Y to (Z, W) to get the estimates of the regression coefficients. We usually assume that W is the unbiased estimator of X such that the error ε has the mean of zero and the standard deviation σ. We also assume the homoscedasticity in this case, which means that the variance is always constant. With the existence of measurement errors, the relationship between Y and X can not be directly modeled by simply fitting Y to W, and certain further adjustment is required. Measurement errors can be classified by different criteria and the combination of them tend to be applied in most of the situations. Several classifications have been made by Zemel'man (1985), Fridman (2012) and Stefanski et al.(2006), and we introduce some of them in the following subsection.

1.2.1 Systematic and Random Error

The systematic and random errors are not caused by observers, but generated by the environment that measurements are involved and the measurment device itself. If the error is random, the values from each measurement are unpredictable, and they can be larger or smaller than the true values due to certain circumstances. For example, random error can be caused by the precision limitation of the measurement device or the changing environment such as temperature, which may not well controlled. If the error is systematic, it is mainly caused by the measuring instrument and it could repeatly happen during the measurement process. For example, the number showing on the body weight measurement is always 1g less than the true weight because of the original setup is not accurate. Systematic error can be a time effected event that we are familiar with. John.R.Tyler (1997) provided some good examples to explain the difference between these two errors, and the comparison of them is also made in his research. In most of the cases, the measurements are influenced by both random and systematic errors. In our assumption, we are only focusing on random error and ignoring the systematic error effect.

1.2.2 Additive and Multiplicative Error

Additive and multiplicative forms can been found in many fields in application. The additive error means that the error is used in terms of additive form, and it has the basic form as $X + \varepsilon$; multiplicative error is existing in the form of multiplication, and it takes the form as $X\varepsilon$. Both additive and multiplicative error models can be applied in practice. The choice can be made based on the nature of the underlying problem.

1.2.3 Differential and Nondifferential Error

Sometimes the difference between these two errors is ignored; however, it is indeed necessary and important to distinguish them. Whether the error is differential or not is determined by the relationship between the observed data W and the response Y conditionally on the true variable X that we can not observe. Suppose that we observed W instead of the true covariate X, and if W does not contain any information about the response Y given X, we say that nondifferential error occurs in this situation. Statistically, the nondifferential error occurs when the distribution of response Y given (X, W) only depends on X; otherwise, the differential error occurs. If the error is nondifferential, W is called a surrogate, which means that W is conditionally independent of Y given X. Stefanski (2006) states two situations when differential error occurs: One is that the response is observed before observing some of the covariates X or W especially in nutrition field; the other case is that W is observed as another different variable performing as a proxy to X. The differential measurement error can be well explained in examples provided by Satton and Kupper (1993), and Kosuke and Teppei (2010). In the parameter estimation process, the nondifferential error is considered to be our priority because we estimate the parameters based on the true covariates, even though the true covariates cannot be observed.

1.2.4 Classical, Berkson and Mixed Error

Classical and Berkson models are considered as the most general and important forms of measurement errors. Stefanski (2006) has classified the measurement error models into two general types: classical measurement error models and regression calibration models, including Berkson error models. These two types of measurement error models are also introduced in Stefanski et al.(2005)

1. Classic measurement error model: This model is well accepted and widely used when the observed values vary depending on the true covariate X. Let us consider additive model with the response Y. Assume that X is the true covariate vector, W is the error-prone observation of X, and Z is the observed data without measurement error. Meanwhile, define ε as the measurement error vector with each element having mean zero and constant variance and assume further that the measurement error is independent of X and Z. Then W can be modeled in the additive form as

$$W = \rho_0 + \rho_x^{'} X + \rho_z^{'} Z + \varepsilon \tag{1.4}$$

where ρ_0, ρ_x, ρ_z are the coefficient parameters. A special case of this model is $W = X + \varepsilon$. In this classical model, the observed covariant equals the true covariant plus the error term. Based on the assumption of the error term, we have that $E(\varepsilon|X, Z) = 0$. It is clear that the variation of the true covariate X is smaller than that of the observed W, which causes the bias estimation when naively using the observed W.

2. Other than the classical model, Berkson (1950) introduced another model for measurement errors in the experimental environment where the observed variable W is controlled. With the same assumptions in the classical model, it has the form

$$X = \rho_0 + \rho_1^{'} W + \rho_2^{'} Z + \varepsilon \tag{1.5}$$

In contrast with the classic model, the Berkson model defines that the conditional expectation of X is based on W and Z, rather than the conditional expectation of W given X and Z in classical model.

3. We can also combine classic and Berkson to get a more complicated error model. An example provided by Stefanski (2006) is given as

$$W = \rho_0 + \rho_z^{'} Z + log(1 + \rho_x^{'} X) + \varepsilon \tag{1.6}$$

This mixed model includes the regular classic error, and a Berkson error in the form of logarithm.

When we consider whether the error model is classic or Berkson, we can make our decision from two aspects. One is that whether the covairates are fixed. For example, if we provide the same number of years as the covariates of two different areas, we want to detect the relationship between the precipitation corresponding to the time. In this case, the only difference between the two areas is determined by the error. Therefore, we use the Berkson error model. The other aspect is about the assumption that we make on the variance of covariate. Stefanski (2006) has stated the criteria for choosing classic or Berkson error model. The variance of the observed data W is larger than the variance of the true covariate (X) in classic error model, and smaller than the variance of X in Berkson error model. This fact simply tells us that assuming the variation of covariates before choosing the model can provide us a hint on deciding which model is more reasonable.

1.3 Model Building

We are interested in estimating the regression parameters, β, in a logistic regression model. The likelihood method is usually adopted to solve this problem. A well-established parametric likelihood can help us finding the maximum likelihood estimator (MLE) of β, which is asymptotically efficient under some regularity conditions. Accordingly, the main procedure for the likelihood analysis with measurement error can be simplified as

1. Specify the form of likelihood;
2. Choose the measurement error model used in the likelihood analysis;
3. Computing the likelihood function;
4. Maximize the likelihood and find the MLE of parameters.

Following the above steps, we can develop our study and try to optimize the result, especially in step 3. Steps 1 and 2 can be easily conducted.

1.3.1 Constructing the Likelihood

Likelihood function is a parametric function based on the sample points we observed and it can be computed based on the distribution of the responses. There are many likelihood-based approaches proposed in literature such as conditional and unconditional likelihood, quasi-likelihood and proportional likelihood. Suppose that the sample size is finite such that we are able to compute the complete likelihood and estimate the parameters numerically. Every likelihood-based analysis starts with specifying the likelihood function. Since the logistic regression is under consideration, the response random variable Y has binary outcomes taking the value 0 and 1. Following the notations given above, $(Y, X, Z, W, \varepsilon)$ are the variables that we need to consider. Suppose that we are able to observe all of the predictors, which means that X and Z are all known to us, the likelihood function can be expressed as

$$L = \prod p_i^{y_i}(1-p_i)^{1-y_i} \tag{1.7}$$

where $i = 1, \cdots, n$, and p_i is the success probability of Y_i given X_i and Z_i , which takes the form

$$p_i = \frac{e^{\beta_0 + \beta_x' X_i + \beta_z' Z_i}}{1 + e^{\beta_0 + \beta_x' X_i + \beta_z' Z_i}} \tag{1.8}$$

1.3.2 Measurement Error Model Selection

Next we need to choose an appropriate error model for our study. Measurement error models are sometimes complicated. We are focusing on a reasonable and simple model to work with. Here we ignore the systematic errors and assume nondifferential error occurs in our case. Additive and multiplicative models are similar. So, we choose an additive model in our study. In regression problems, we are always expecting that the covariate can be able to observed with or without errors.Thus, we assume that the covariate are given so that the Berkson model is preferred. Finally, we decide to use additive Berkson error model to develop our study and the other error models can be detected in the future work. Without the accurate measured variable Z, the model is further simplified as

$$X = W + \varepsilon. \tag{1.9}$$

1.3.3 Our Model

Based on the likelihood form and measurement error model that we have selected, we are ready to build our model. If a random error variable is involved, the conditional density function of the response is given by

$$f(Y|W, \varepsilon) = \frac{e^{\beta_0 + \beta_1'(W+\varepsilon)}}{1 + e^{\beta_0 + \beta_1'(W+\varepsilon)}} \tag{1.10}$$

We assume that the error ε follows the multivariate normal distribution given by

$$f(\varepsilon) = (\frac{1}{2\pi})^{\frac{p}{2}} \mid V \mid^{\frac{1}{2}} e^{\frac{1}{2}\varepsilon' V^{-1}\varepsilon} \tag{1.11}$$

where V is the variance covariance matrix. Then we can integrate the error part out and obtain the mean response as

$$p_i = p(Y=1|W_i) = \int_{-\infty}^{+\infty} \frac{e^{\beta_0+\beta_1'(W_i+\varepsilon)}}{1+e^{\beta_0+\beta_1'(W_i+\varepsilon)}} (\frac{1}{2\pi})^{\frac{p}{2}} \mid V \mid^{\frac{1}{2}} e^{\frac{1}{2}(X-W_i)'V^{-1}(X-W_i)} dx \tag{1.12}$$

Now, the likelihood function can be expressed as the equation (1.7), where p_i takes the form as given in equation (1.12).

To reach the goal of finding the MLE of the parameters, we will conduct our study step by step and take a special care on computing the value of p_i. Since we have already decided the form of our likelihood, which takes the form as a logistic-normal integral, several typical methods for computing logistic-normal integral will be introduced first. One of the methods we are interested is proposed by Crouch and Spiegelman (1990)(C-S method), which applies mathematical technique to obtain a more accurate value of p_i. Then, the maximization method of likelihood will be discussed based on C-S method. In order to efficiently estimate parameters for C-S method, we use the technique of calling the FORTRAN code in R to obtain p_i. Then, the simulation will reveal the result of how C-S methond performs in terms of precision and accuration.

The whole project will reach the objective of finding a more efficient way of computing logistic-normal integral and estimating the likelihood parameters. The core process is to use R instead of FORTRAN to improve the time efficiency based on the C-S method. The better performance of the estimation results will definitely benefit on the improvement of measurement error problems and computational users.

Chapter 2

Computing logistic-normal integral

The likelihood in the logistic-normal model involves the logistic-normal integral. The logistic-normal integral is the essential part in the logistic regression with a normal-distributed covariate measurement error. In order to compute the logistic-normal integral in the form of (1.12), various methods have been proposed in literature and the attempt of achieving a perfect result of approximating this kind of integral has never stopped. Comparisons among different methods have been developed by statisticians. Our interest lies in the numerical methods which are easy to develop and efficient computationally.

2.1 Probit Approximation

Probit is known as a quantile function related to the inverse cumulative distribution function of the normal random variable. In the logistic-normal model, logit function can be well approximated by probit based on the fact that the probit is immune from taking expectation, which is free of integration. Without considering regression part, the one dimensional logistic normal integral Λ can be written as

$$\Lambda = \int_{-\infty}^{\infty} \frac{e^x}{1+e^x} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-z)^2}{2\sigma^2}} dx. \tag{2.1}$$

Equation (2.1) can be interpreted as the expectation of a function of a normal random variable

$$\Lambda = E_\varepsilon \left(\frac{e^{z+\sigma\varepsilon}}{1+e^{z+\sigma\varepsilon}}\right), \tag{2.2}$$

where ε follows a normal distribution with mean 0 and variance 1. An important formula to approximate Λ using Φ provided by E. Demidenko (2004) is given as

$$E_\varepsilon \Phi(x+\varepsilon) = \Phi\left(\frac{x}{\sqrt{1+\sigma^2}}\right), \tag{2.3}$$

where ε follows the normal distribution with mean 0 and variance σ^2. After some simple transformation, Λ can be rewritten as

$$\Lambda \simeq \Phi\left(\frac{x}{\sqrt{c^2+\sigma^2}}\right). \tag{2.4}$$

2.1.1 One-probit approximation

The one-probit approximation is given as

$$F = \frac{e^x}{1+e^x} \simeq \Phi\left(\frac{x}{c}\right), \tag{2.5}$$

where $c > 1$. According to Johnson and Kotz (1970) and Caroll et al.(1995,p.64), it is reasonable to choose $c \simeq 1.7$. Based on the equation (2.4), the one-probit approximation is

$$\Lambda \simeq \Phi\left(\frac{x}{\sqrt{1.7^2+\sigma^2}}\right). \tag{2.6}$$

2.1.2 Two-probit approximation

The two-probit approximation has a similar form as (2.5). It is the linear combination of two probits where the sum of coefficients is equal to 1. To approximate F, we have

$$F = \frac{e^x}{1+e^x} \simeq p\Phi(\frac{x}{c_1}) + (1-p)\Phi(\frac{x}{c_2}), \tag{2.7}$$

where the value of p is between 0 and 1, and c_1 and c_2 can be chosen according to the criteria provided by Eugene(2004,p.336) given as

$$\int_{-\infty}^{\infty} F - p\Phi(\frac{x}{c_1}) + (1-p)\Phi(\frac{x}{c_2})dx. \tag{2.8}$$

The optimization technique such as Newton-Raphson Method can help us minimize the above integral such that p, c_1, c_2 can be obtained. Then, we have

$$F \simeq 0.4353\Phi(\frac{x}{2.2967}) + 0.5647\Phi(\frac{x}{1.3017}). \tag{2.9}$$

Now consider our logistic-normal integral, based on (2.4) and (2.9), we have

$$\Lambda \simeq 0.4353\Phi(\frac{x}{2.2967^2+\sigma^2}) + 0.5647\Phi(\frac{x}{1.3017^2+\sigma^2}). \tag{2.10}$$

2.2 Laplace Approximation

Laplace approximation is a very important method for nonlinear models. The idea of Laplace approximation is to use quadratic approximation at the points where the integrand has the maximum value. First, let us rewrite the integrand $\frac{e^{x+\sigma\varepsilon}}{1+e^{x+\sigma\varepsilon}}$ as $e^{l(\varepsilon)}$, where $l(\varepsilon) = x + \sigma\varepsilon - \ln(1+e^{x+\sigma\varepsilon})$. Define $h(\varepsilon) = l(\varepsilon) - \frac{1}{2}\varepsilon^2$. Suppose that we need to approximate the integral $\int_{-\infty}^{\infty} e^{h(\varepsilon)}d\varepsilon$, we need first to find $max[h(\varepsilon)]$. Since $h(\varepsilon)$ would disappear when x reaches the maximum value, we come to the second-order approximation

$$h(\varepsilon) \simeq h_{max} + \frac{1}{2}(\varepsilon - \varepsilon_{max})^2(\frac{d^2h}{d\varepsilon^2}|_{\varepsilon=\varepsilon_{max}}). \tag{2.11}$$

Based on the identity

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{ax-bx^2}dx = \frac{1}{2b}e^{\frac{a^2}{4b}}, \tag{2.12}$$

we have

$$\int_{-\infty}^{\infty} e^{h(\varepsilon)}d\varepsilon \simeq \sqrt{2\pi}e^{h_{max}}(-\frac{d^2h}{d\varepsilon^2})^{-1/2}. \tag{2.13}$$

Now we apply Laplace method in our logistic normal model and rewrite (2.1) as

$$\Lambda = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{h(x)}dx. \tag{2.14}$$

It is easy to prove that $h \to -\infty$ when $\varepsilon \to \pm\infty$, and the h_{max} exists. The first and second derivatives of h are given as

$$\begin{aligned} \frac{dh}{d\varepsilon} &= \frac{\sigma}{1+e^{x+\sigma\varepsilon}} - \varepsilon \\ \frac{d^2h}{d\varepsilon^2} &= -[\frac{\sigma^2 e^{x+\sigma\varepsilon}}{(1+e^{x+\sigma\varepsilon})^2} + 1]. \end{aligned} \tag{2.15}$$

Since the negative second derivative leads to the concave function h, we can use the Newton algorithm

$$\varepsilon_{x+1} = \varepsilon_x + \frac{[\sigma - \varepsilon(1+e^{x+\sigma\varepsilon_x})](1+e^{x+\sigma\varepsilon_x})}{\sigma^2 e^{x+\sigma\varepsilon_x} + (1+e^{x+\sigma\varepsilon_x})^2} \tag{2.16}$$

to obtain ε_{max}, which is the limit point of iteration. Then based on (1.12), we have

$$\Lambda \simeq \frac{e^{x+\sigma\varepsilon_{max}-\frac{1}{2}\varepsilon_{max}^2}}{\sqrt{\sigma^2 e^{x+\sigma\varepsilon_{max}} + (1+e^{x+\sigma\varepsilon_{max}})^2}}. \tag{2.17}$$

In practice, we can replace ε_{max} using the first iteration ε_1, which also provides a good approximation to ε_{max}. We notice that in Laplace approximation procedure, the iteration process is involved in most cases to obtain the maximum values. Because of this fact, the problem of time consuming may appear to be one of our concerns.

2.3 Gaussian Quadrature

The Gaussian Quadrature is believed to be a convincing numerical method of computing logistic-normal integral effectively. Many generalizations have been developed to well explain the application of using this numerical way: Geert Verbeke (1995) has shown that one can apply adaptive Gaussian Quadrature method instead of Penalized quasi-likelihood (PQL) or Marginal quasi-likelihood (MQL) in logistic regression model because both of them have some disadvantages. For example, MQL always provids biased estimation and PQL obtains good results only when the sample size is large. Besides, Jorge Gonzalez (2006) compared three different numerical methods of computing this integral: Gauss-Hermit Quadrature (GH), Monte Carlo Method (M-C) and Quasi-Monte Carlo Method (QMC). However, the comparison only focused on the number of points used as well as the estimation precision. One of the useful quadratures based on Gaussian Quadrature is Gauss-Hermite Quadrature. As an extension of Gaussian Quadrature, it aims to approximate the value of the integral

$$\int_{-\infty}^{\infty} f(t)e^{-t^2}dt. \tag{2.18}$$

Similar to normal Gaussian Quadrature, we can also use the trapezoidal rule to approximate this kind of integral such that

$$\int_{-\infty}^{\infty} f(t)e^{-t^2}dt = \sum_{i=1}^{n} w_i f(x_i). \tag{2.19}$$

Here, w_i and x_i are known as the weights and the nodes(abscissas). The values of w_i and x_i are provided in the table when $n \leq 20$ (Abramowitz and Stegun, 1967, p.924,)[see the Appendix B]. The value of K is determined by the error of the approximation, and several options to compute K have been proposed by statisticians. The effect of choosing different number of points for approximation has been discussed (Emmanuel and Bart,2000). It is believed that Gaussian-Hermite Quadrature with $K = 11$ or 13 are reasonable to approximate logistic normal integral (Eugene, 2004). Also, Lesaffre and Spiessens (2000) has discussed the number of quadrature points in logistic random-effect model and stated that $K = 10$ is usually adequate for approximation. The well accepted situation is that $K = 20$ is adequate for approximating this integral (Pierce and Sands 1975). We will try different numbers of K in our study to compare the accuracies, and detect if there is any particular trend of the number of K.

2.4 Quasi-Monte Carlo Method

Compared with Gaussian Quadrature, Monte Carlo method is considered to be applied better in high dimensional cases. Quasi-Monte Carlo integral is an optimal generalization of Monte Carlo method in terms of choosing quadrature points. The approximation equation for both cases is given as

$$\int_{\Omega} f(x_1, \cdots, x_p) d_{x_1}, \cdots, d_{x_p} \approx \frac{V_{\Omega}}{N} \sum_{i=1}^{N} f(x_{i1}, \cdots, x_{ip}). \tag{2.20}$$

V_{Ω} is known as the volume of Ω with the form $\int_{\Omega} d_{x_1}, \cdots, d_{x_p}$. In the regular Monte Carlo method, $(x_{i1}, \cdots, x_{ip})$ are chosen randomly from a uniform distribution in Ω. In the Quasi-Monte Carlo method, the points in this vector are not a random sample, but a set of points that can be determined through a uniformly distributed deterministic sequence, called low discrepancy sequence (LDS) (Niederreiter, 1992). There are many discrepancy sequences existing for us to choose the points. In order to compute the logistic normal integral in the form of (1.12), we can apply Halton and Sobol's sequences (Kocis and Whiten, 1997) to determine the points because they are the most frequently used .

Both Gaussian and Monte-Carlo approximation are widely used and accepted in the calculation process. The focus of our study in this thesis is to investigate whether other methods can perform them out in both precision of the calculation and the computational efficiency.

2.5 The C-S Method

One of the numerical methods was proposed by Goodwin(1949), which approximates the integral of the form of (2.18) using a simple trapezoidal rule

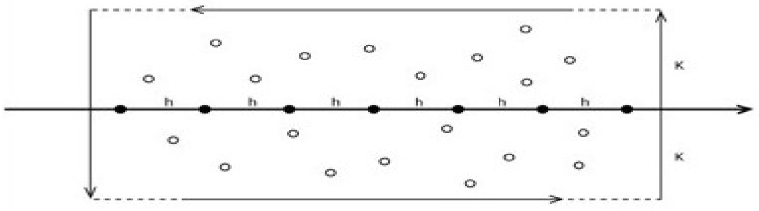
$$\int_{-\infty}^{\infty} f(t)e^{-t^2}dt = h\sum_{n=-\infty}^{\infty} f(nh)exp(-n^2h^2) + \varepsilon(h). \tag{2.21}$$

This method allows a large step size h for a given error, but the assumptions of $f(t)$ restricts the application of using this method.

The so called C-S method was proposed by Crouch and Spiegelman (1990), which is basically a generalization and optimization of Goodwin's approximation. This method aims to help removing some restrictive assumptions of $f(t)$ and try to obtain a better result. The idea of this method is to do the calculation by expending the real number field to the complex field. Suppose that our $f(t)$ has some simple poles within the strip $-K < f(t) < K$. Based on the general form of our likelihood, we multiply the integrand $f(t)exp(-t^2)$ by a function

$$1 - C(t)^{-1}, \tag{2.22}$$

where $C(t) = e^{-2\pi i(t-t_0)/h}$. This function has simple poles $(t_0 + nh)$ along the real axis and t_0 can be chosen for convenience. Then the modified integrand can be integrated around a rectangular contour. Let us make it a distance K from the real axis such that it contains all the residues on or depart from the real axis inside the rectangular.



The limit theory can be applied such that when the rectangular contours approches to infinity in real direction, it leaves the integral parts only over the edges above and below the real axis approximately. According to residual theory and the definition of contour integration, we have

$$\int_{-\infty-K}^{+\infty-K} f(t)[e^{-t^2}][1-C(t)]^{-1}dt + \int_{-\infty+K}^{+\infty+K} f(t)[e^{-t^2}][1-C(t)]^{-1}dt = \\ h\sum_{n=-\infty}^{\infty} f(t+nh)e^{-(t_0+nh)^2} + 2\pi i \sum_j resf(t_j)[e^{-t_j^2}][1-C(t_j)]^{-1}. \tag{2.23}$$

Note that the integral below the real axis is divided into two parts and one of them is exactly the one we are looking for:

$$\int_{-\infty-K}^{+\infty-K} f(t)[e^{-t^2}][1-C(t)]^{-1}dt = \\ \int_{-\infty-K}^{+\infty-K} f(t)[e^{-t^2}]dt + \int_{-\infty-K}^{+\infty-K} f(t)[e^{-t^2}]C(t)[1-C(t)]^{-1}dt \tag{2.24}$$

By choosing h, the remainder can be negligibly small. At last, the left hand side integral of the equation is a trapezoidal-rule-like sum for the desired integral

$$\int_{-\infty}^{+\infty} f(t)[e^{-t^2}]dt = h\sum_{n=-\infty}^{\infty} f(t_0+nh)e^{-(t_0+nh)^2} + \varepsilon(h) + Res, \tag{2.25}$$

where $\varepsilon(h)$ is the error term bounded in a negligible small interval for us to specify, 'Res' is the summarized residual coming from the holes within the rectangular contour. Then we can choose h and k to make the error term as small as possible. This method seems complicated to understand and needs tremendous amount of mathematics calculation, but it uses mathematical techniques to solve the statistical problem. This can sometimes be really beneficial. Using this numerical method, we can specify and control the maximum acceptable error easily and the restriction on $f(t)$ only depends on the analytic structure of itself. In the application for real data set, this method becomes much easier.

Among all of the methods listed above, Gaussian Quadrature method has been considered to be the standard method and widely applied in practical computation. Next, we focus on the simple lower dimensional case and mainly compare Gaussian Quadrature with C-S method in our study with respect to the precision and computational efficiency, when estimating the unknown coefficients in logistic regression.

Chapter 3

Maximization of the Likelihood

In this chapter, we apply the C-S method to the likelihood calculation of our error-in-variable logistic regression model. In our model setting, $f(t)$ is a probability distribution function–it is always hermitian analytic (Su, 1990, p.468). One application of this generalized numerical method is to specify the form of $f(t)$ to $1/(1+e^{x+yt})$, which can generally represent the form of logistic regression model. Based on the method we discussed above, we compute

$$\int_{-\infty}^{+\infty} \frac{1}{1+e^{x+yt}} e^{-t^2} dt. \tag{3.1}$$

In order to satisfy $|f(t+ik)| \leq 1$, the only restriction is given by $2j\pi - \pi/2 \leq yk \leq 2j\pi + \pi/2$, and this leads to $k \leq \pi/2y$. Here, there are no residuals of poles from $f(t)$ involved in our calculation.

There are two different situations in the selection of h .

(1). Set $k = \pi/h$, which allows

$$|\varepsilon(h)| \leq 2\pi^{1/2} e^{-\pi^2/h^2}. \tag{3.2}$$

If the maximum acceptable error is η, choose $h = \pi[\ln(2\pi^{1/2}/\eta)]^{-1/2}$. In this case, the condition is $k = \pi/h \leq \pi/2y$, which implies $y \leq h/2$.

(2). The other situation happens when $y \geq h/2$. Let us set $k = \pi/2y$, which leads to $h = 4y\pi^2/(\pi^2 + 4y^2 \ln(2\pi^{1/2}/\eta))$ to reach the desired accuracy. To simplify our calculation, define

$$I(x, y; n) = \pi^{-1/2} \int_{-\infty}^{+\infty} \frac{1}{(1 + e^{x+yt})^n} e^{-t^2} dt, and \tag{3.3}$$

$$J(x, y; n) = exp(nx + n^2y^2/4)I(x + \frac{1}{2}ny^2, y; n + 1) \tag{3.4}$$

And the partial derivative of $I(x, y; n)$ and $J(x, y; n)$ are given by Crouch and Donna (1990) as

$$I_x(x, y; n) = -nexp(x + \frac{1}{4}y^2)I(x + \frac{1}{2}y^2, y; n + 1) \tag{3.5}$$

$$I_y(x, y; n) = \frac{1}{2}yI_{xx}(x, y; n) \tag{3.6}$$

$$J_x(x, y; n) = nJ(x, y; n) - (n - 1)J(x, y; n + 1) \tag{3.7}$$

Based on all of these quantities, we are now ready to calculate our likelihood for maximization process.

Based on the likelihood function given in (1.7), the log-likelihood function can be written as

$$\ell(\beta) = \sum y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i). \tag{3.8}$$

Note that in the formula (1.12), if β_1 is one dimensional, p_i has the similar form of $J(x, y; 0)$. After transformation, we have

$$p_i = \pi^{-1/2} \int_{-\infty}^{+\infty} \frac{1}{e^{x_i+yt}} e^{-t^2} dt \tag{3.9}$$

where $x_i = -(\beta_0 + \beta_1 z_i)$, $y = -\sqrt{2}\sigma\beta_1$. After some simple calculation, we have

$$p_i = J(x, y; 0) = I(x, y; 1) = J_x \tag{3.10}$$

Now we can calculate $\ell(\beta)$ using computer software and maximize it to obtain the MLE of β.

The maximization methods are numerous and widely applicable. We can use either of them to obtain $\hat{\beta}$, which are equally stable. For simplicity of the illustration, we consider the maximization of the likelihood with only 2 parameters, $\beta = (\beta_0, \beta_1)$. The first and the second derivative of $\ell(\beta)$ are involved in the maximization process, which are calculated as

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_0} &= \sum_{i=1}^{n} [-\frac{y_i}{J_i} J_x + \frac{1-y_i}{1-J_i} J_x], \\
\frac{\partial \ell}{\partial \beta_1} &= \sum_{i=1}^{\infty} [-\frac{y_i}{J_i} (z_i J_x + \sqrt{2}\sigma J_y) + \frac{1-y_i}{1-J_i} (z_i J_x + \sqrt{2}\sigma J_y)], \\
\frac{\partial \ell^2}{\partial \beta_0 \partial \beta_1} &= \sum_{i=1}^{\infty} y_i [\frac{(J_x z_i + J_y \sqrt{2}\sigma) J_x}{J_i^2} + \frac{J_{xx} z_i + J_{xy} \sqrt{2}\sigma}{J_i}] \\
&\quad + (1-y_i)[\frac{J_x z_i + J_y \sqrt{2}\sigma) J_x}{(1-J_i)^2} - \frac{J_{xx} z_i + J_{xy}\sqrt{2}\sigma}{1-J_i}], \\
\frac{\partial \ell^2}{\partial \beta_0^2} &= \sum_{i=1}^{\infty} [y_i (\frac{J_x^2}{J_i^2} + \frac{J_{xx}}{J_i}) + (1-y_i)(\frac{J_x^2}{(1-J_i)^2} - \frac{J_{xx}}{1-J_i})], \\
\frac{\partial \ell^2}{\partial \beta_1 \partial \beta_0} &= \sum_{i=1}^{\infty} y_i z_i [\frac{J_x^2 + J_i J_{xx}}{J_i^2}] + \sqrt{2} y_i \sigma [\frac{J_x^2 + J_i J_{yx}}{J_i^2}] \qquad (3.11) \\
&\quad + (1-y_i) z_i [\frac{J_x^2}{(1-J_x)^2} - \frac{J_{xx}}{1-J_i}] + (1-y_i)\sqrt{2}\sigma [\frac{J_x^2}{(1-J_x)^2} - \frac{J_{yx}}{1-J_i}], \\
\frac{\partial \ell^2}{\partial \beta_1^2} &= \sum_{i=1}^{\infty} y_i z_i [\frac{(J_x z_i + J_y \sqrt{2}\sigma) J_x}{J_i^2} + \frac{J_{xx} z_i + J_{xy}\sqrt{2}\sigma}{J_i}] \\
&\quad + \sqrt{2}\sigma y_i [\frac{(J_x z_i + J_y \sqrt{2}\sigma) J_y}{J_i^2} + \frac{J_{yx} z_i + J_{yy}\sqrt{2}\sigma}{J_i}] \\
&\quad + (1-y_i) z_i [\frac{(J_x z_i + J_y \sqrt{2}\sigma) J_x}{J_i^2} + \frac{J_{xx} z_i + J_{xy}\sqrt{2}\sigma}{J_i}] \\
&\quad + (1-y_i)\sqrt{2}\sigma [\frac{(J_x z_i + J_y \sqrt{2}\sigma) J_y}{J_i^2} + \frac{J_{yx} z_i + J_{yy}\sqrt{2}\sigma}{J_i}],
\end{aligned}$$

where

$$\begin{aligned}
&J_x = -J(x,y;1)\\
&J_y = (-1/2)yJ(x,y;1) - 2J(x,y;2)\\
&J_{xx} = -J(x,y;1) - 2J(x,y;2\\
&J_{yx} = -(1/2)y[J(x,y;1) - 2J(x,y;2)] + y[2J(x,y;2) - 3J(x,y;3)]\\
&J_{xy} = -(y/2)J(x,y;1) - exp(x+\frac{y^2}{4})[(1/2)yI_{xx}(x+\frac{y^2}{2},y;2)]\\
&J_{yy} = -(1/2)[yJ_y(x,y;1) + J(x,y;1)] + [yJ_y(x,y;2) + J(x,y;2)]\\
&J_y(x,y;1) = \frac{y}{2}J(x,y;1) - y[J(x,y;2) - 3J(x,y;3)]\\
&J_y(x,y;2) = 2yJ(x,y;2) - \frac{3y}{2}[J(x,y;3) - 4J(x,y;4)]\\
&I_{xx}(x+\frac{y^2}{2},y;2) = -2exp(-x-\frac{y^2}{4})[J(x,y;2) - 3J(x,y;3)]\\
&I_{xx}(x+\frac{y^2}{2},y;3) = -3exp(-2x-y^2)[J(x,y;3) - 4J(x,y;4)]
\end{aligned} \tag{3.12}$$

Using these quantities, we can construct our maximization structures directly.

The commonly used maximization method to find MLE is to solve the equation system

$$\begin{cases} \frac{\partial \ell}{\partial \beta_0} = 0, \\ \frac{\partial \ell}{\partial \beta_1} = 0. \end{cases} \tag{3.13}$$

This system contains two nonlinear equations and we can solve it using certain computer software. Then, we can obtain our estimators $\hat{\beta}$. We can also consider the Newton-Raphson method to solve the iteration equation given as

$$\beta^{k+1} = \beta^k + F^{-1}\varepsilon, \tag{3.14}$$

where F is the Hessian matrix given as

$$\begin{pmatrix} \frac{\partial \ell^2}{\partial \beta_0^2} & \frac{\partial \ell^2}{\partial \beta_0 \partial \beta_1} \\ \frac{\partial \ell^2}{\partial \beta_1 \partial \beta_0} & \frac{\partial \ell^2}{\partial \beta_1^2} \end{pmatrix}$$

. The elements in the Hessian matrix are provided in the above equations. This iteration process will lead to the MLE of β when it converges.

Chapter 4

Calling J(x,y;n) in R

The likelihood calculation process has involved the function J(x,y;n). Efficient calculation of its values will be essential for the maximization of the corresponding likelihood function. FORTRAN and R are both considered as good options by statisticians. FORTRAN is a traditional and powerful scientific computing software with applications in many fields. Many other software packages were written originally in FORTRAN. Also, it is much faster in execution. However, we have to accept the fact that FORTRAN is becoming an old tool even though some modern features are added to it. R is a free public shared statistical environment and widely used among modern academic communities. There are many online resources with detailed documentation available nearly everywhere. R is also famous for its compatibility since it can run in Windows, Linux, Mac OS and etc. You can even call functions, routines written in other languages into R for application. Crouch and Spiegelman (1990) has provided the FORTRAN subroutine for the calculation of $J(x, y; n)$. We are hence motivated to explore the potential of their method in statistical applications. In the remaining part of this chapter, we provide the details of the procedure of calling the C-S subroutine into R for the calculation of the likelihood of our logistic-normal model.

First, it is important to know that R can only call files from the dynamic link library(DLL), so if we do not have the ”*.dll*” file available, we have to compile our existing file into the ”*.dll*” type. The second point is that R can only call FORTRAN subroutines rather than functions. There are slight differences between the definitions of subroutine and function in FORTRAN. Subroutine is the routine that can be called several times and executed with multiple outputs. However, a function has only single output and can be called once at a time. After some modification on the given subroutine, we are ready to use it in compilation phase. The detailed steps are summarized as following and these steps can help us realize the calling process in most situations.

1. Download the compiler. There are two compilers commonly used for compiling, $'MinGW'$ and $'Cygwin'$. In our case, we download $'MinGW'$ from its official web site as well as the package $'g77'$. We then install the compiler in $'C:\backslash MinGW'$. A good news is that the latest version of $'MinGW'$ can be used in either 32 or 64 bit computer system. The package $'g77'$ is such a good tool, especially for compiling FORTRAN codes. Actually, $'g77'$ is an old compiling package. We can also use $'gfortran'$ or $'gcc'$ instead of $'g77'$ to compile DLL files.

2. Go to $'MyComputer'$ and create a dictionary called $'Programs'$ inside $'C:\backslash MinGW'$ directory.

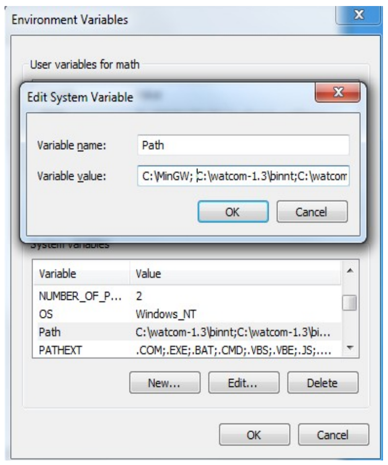
3. Open Notepad (or WordPad) and enter the FORTRAN code that we have already modified. Then we have to choose the file type $'All\ type'$ and save the file named $'goodwin.f'$. Be careful: we need to make sure that the format of the FORTRAN code is correctly used such as keeping 8 bytes space ahead in each row. If

the format is not correct, the error will occur and the compiling process can not be completed.

4. Change the environment variable. Go to

$$'MyComputer' \Rightarrow 'SystemProperty' \Rightarrow 'Advancedsystemsetup' \Rightarrow 'EnvironmentVariable'$$

and add the directory $'MinGW'$ to the path. Do not overlay the directory on any existing one since that would change or delete some other system paths. Just add it in front of the existing path and that will be effective.



Variable name: Path

Variable value: C:\MinGW; C:\watcom-1.3\binnt;C:\watcom

Variable	Value
NUMBER_OF_P...	2
OS	Windows_NT
Path	C:\watcom-1.3\binnt;C:\watcom-1.3\bi...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

5. Now we start to use DOS system to create what we need. We can also use the compiler to compile directly but DOS is easier to use. Choose $'Run'$ under start menu and enter $'cmd'$.

6. Enter the DOS command to correct directory

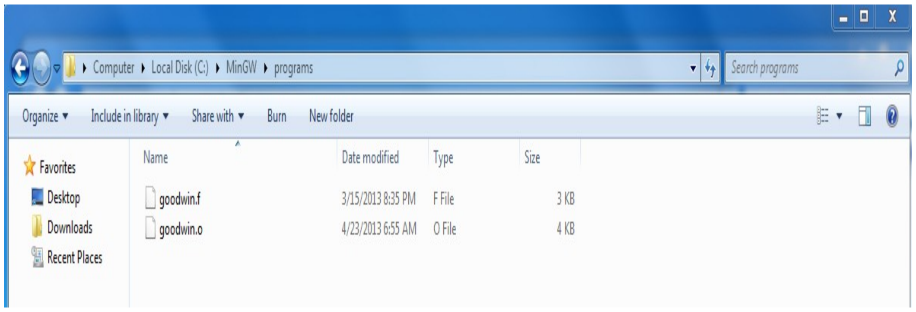
$$cd \quad C:\backslash mingw \backslash programs$$

7. Use $'g77'$ to compile the code. Note that the space between the letters in the

command must not be ignored

$$g77 \quad -c \quad goodwin.f$$

After this step, another file of the $'.o'$ type has been created in the same directory.

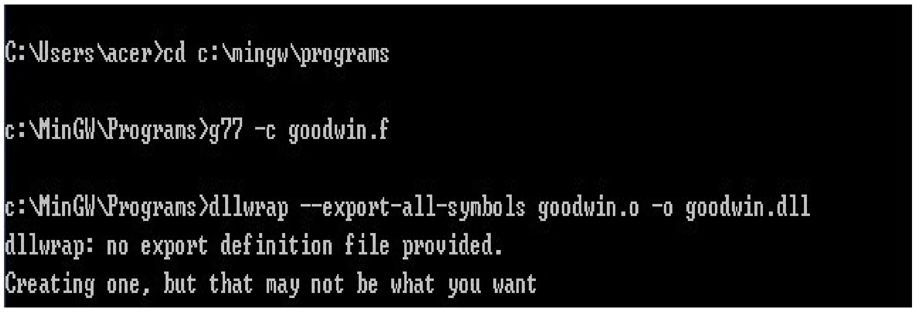


Name	Date modified	Type	Size
goodwin.f	3/15/2013 8:35 PM	F File	3 KB
goodwin.o	4/23/2013 6:55 AM	O File	4 KB

8. Create a $'dll.'$ file:

$$dllwrap \quad --export-all-symbolsgoodwin.o \quad -o \quad goodwin.dll$$

The result showing in $'cmd'$ is



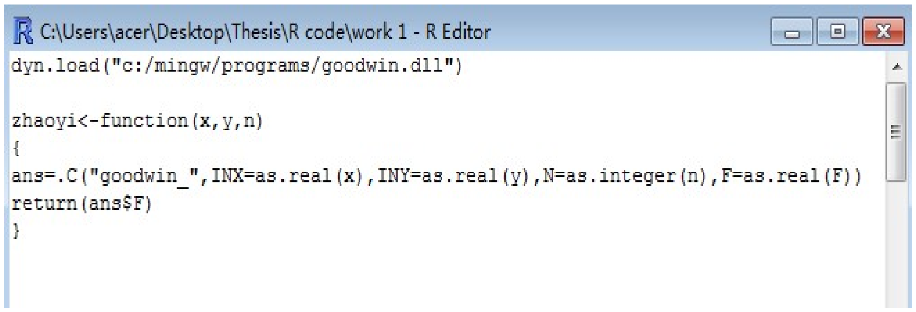
```
C:\Users\acer>cd c:\mingw\programs

c:\MinGW\Programs>g77 -c goodwin.f

c:\MinGW\Programs>dllwrap --export-all-symbols goodwin.o -o goodwin.dll
dllwrap: no export definition file provided.
Creating one, but that may not be what you want
```

We notice that there are always a few lines of explanations right after the last row of the command. We can just ignore it automatically since it doesn't affect the generation of the desired file. Then we have successfully created the $'.dll'$ file, called $'goodwin.dll'$.

9. Write a wrapper function in R to generate a function in order to simply call $J(x, y; n)$.



```
dyn.load("c:/mingw/programs/goodwin.dll")

zhaoyi<-function(x,y,n)
{
ans=.C("goodwin_",INX=as.real(x),INY=as.real(y),N=as.integer(n),F=as.real(F))
return(ans$F)
}
```

Note that the space between the letters in the command must not be ignored; $'.Fortran'$ and $'.C'$ command both load and execute the $'goodwin.f'$ in R. When using $'.Fortran'$, the underscore that appears in $'.C'$ is optional. When using$'.C'$ or $'.Fortran'$, one important thing is to identify the variable format correctly corresponding to the format given in FORTRAN subroutine such as the $'INT'$ in FORTRAN is corresponding to the integer in R. After running the above R code, we can call the function $J(x, y; n)$ to get the value of it if the values of x, y, n are given. This kind of technique is very useful and efficient in application since calling subroutines from FORTRAN can save much more time than using R code in the computing phase. When FORTRAN code and software R are both available, we can follow the previous steps to achieve fast computation.

Chapter 5

Simulation Study and Comparison with Gaussian Quadrature

In this chapter, we examine the performance of C-S method and compare it with the classic Gaussian Quadrature on our logistic normal model. We mainly focus on the precision of the estimation and the CPU time for both cases. We choose $K = 10, 16, 20$ for the Gaussian Quadrature method and report the precision of the estimates and CPU time. Then we compare them with those based on the C-S method.

Without loss of generality, we consider the simple Berkson model in order to make the comparison apparent, which allows β=(β_0,β_1), X=(1,x), $x = w + \epsilon$, where $\epsilon \sim Normal(0, \sigma^2)$. We do the simulation with sample sizes of 500 and 1000.

Suppose we use n=1000. First, we generate 1000 sample points for ϵ, which follows a normal distribution with mean zero and constant variance σ^2, then we generate 1000 data for w, which follows a uniform distribution between 0 and 4. Now we can obtain 1000 data points for x, where $x_i = w_i + \epsilon_i$. Depending on the true covariate X, set

β=(0,1) and we have the success probability

$$P(i) = P(Y = 1|x_i) = \frac{exp(\beta_0 + \beta_1 x_i)}{1 + exp(\beta_0 + \beta_1 x_i)} \tag{5.1}$$

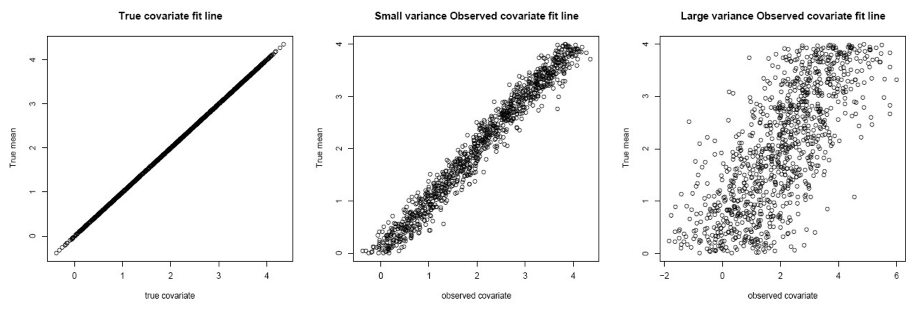
Depending on the observed covariate W, we have the success probability defined as

$$P_i = p(Y = 1|w_i) = \frac{1}{\sqrt{2\pi}\sigma} \int \frac{exp(\beta_0 + \beta_1 x)}{1 + exp(\beta_0 + \beta_1 x)} e^{\frac{(x-w_i)^2}{2\sigma^2}} dx \tag{5.2}$$

which is a special form of (1.12). Since the response Y has the binary outcome which follows a binomial distribution, so we generate y_i based on $P(i)$. Now we have obtained the observed data (y_i,w_i),where i=1,2,$\cdots$,1000.

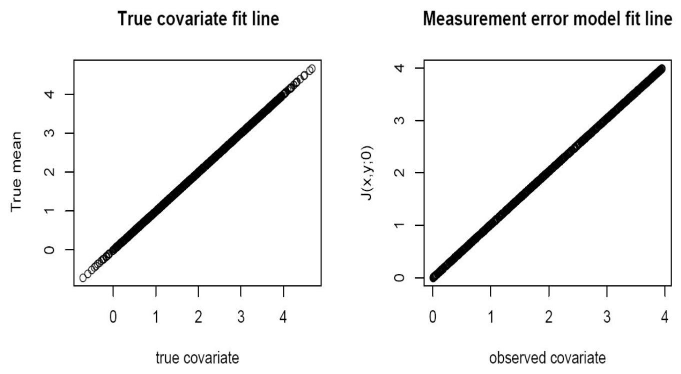
5.1 Effect of Error and Variance

We already know that W is the measurement error prone version of X. Substituting W for X without any adjustment may cause serious estimation bias. In logistic regression, if we consider L in equation (1.3), then L should have a linear relationship with the covariate X. The measurement error contained in W could ruin this linear relationship when the variance of the measurement error is large. Let us take a sample of 1000 data points of L corresponding to X and W, compare different variance values of 0.2 and 1. After plotting (L, X) and (L, W) with different variance, we have



The plots tell us clearly that the measurement error can cloud the true relationship in regression problems. If we shrink the sample size, the linear feature may be even harder to observe. The variance of the error term can affect the fitting line as well as the estimation result.

The modified logistic regression model provides a different plot. In order to compare it with the true logistic regression model, we plot $\ln(\frac{P(i)}{1-P(i)})$ corresponding with the true covariate X, and $\ln(\frac{P_i}{1-P_i})$ corresponding to the observed value W, we have



This plot shows that the two lines are overlaid almost exactly. It means that if the measurement error is involved, Berkson additive error model can be properly used. By using the logistic normal integration technique, the error effect can be removed reasonably.

5.2 Integral Computing and Comparison

The logistic normal integral can be calculated in different ways. We only investigate the C-S method and the Gaussian Quadrature method. First look at the approximated value for $J(x, y, 0)$ using both the C-S and the Gaussian Quadrature methods. For illustration, we randomly choose three different combinations of x and y in order to compute the true values of integral $J(x, y, 0)$. Then we use these two methods to approximate the desired integral and make the comparison with the true ones. Notice that x in $J(x, y; n)$ is somehow determined by the observed variable w, and y in $J(x, y; n)$ is always fixed by parameters. So the variable x is the key effect to show the path of approximation process. The factor $'Time'$ we are using is the average 'Elapsed' time in R over the three combinations of (x, y).

$method \setminus (x, y)$	$(-0.3993073, 1.131371)$	$(-1.3766027, 1.131371)$	$(-3.5777936, 1.131371)$	Time(s)
$G(k = 10)$	0.586658540925351	0.771806831635967	0.964017922924520	0.33
$G(k = 16)$	0.586658566350867	0.771806830493563	0.964017923148684	0.44
$G(k = 20)$	0.586658564500420	0.771806826731734	0.964017924744031	0.49
C.S	0.586658564469190	0.771806826660444	0.964017924721894	0.11
True	0.586658564469532	0.771806826660775	0.964017924722125	

Table 5.1: Comparison of two methods on approximating integral

Table 5.1 shows that

1. The C-S method and Gaussian Quadrature generate very similar results with difference only after several decimals, where C-S method obtained more accurate values of the integral. Due to the high precision of both methods, we cannot expect remarkable difference in the estimation of the regression parameters. This has been verified by the simulation study and the results are almost identical.

2. Although the precision of both methods are high in approximation, the difference of computing time is quite remarkable.

3. The smaller K values in Gaussian method need shorter time for the computation but lose some accuracy in approximation results.

The difference of the approximation using both methods is almost ignorable. But the computing time needed for C-S method is much shorter. Hence C-S method has great potential in practice.

5.3 Parameter Estimation

Assuming that the standard deviation takes the value of 0.3 and 0.8 and the the true value of β is $(0, 1)$, we take the sample size 500 and 1000 separately for simulation, and conduct 500 times simulation for both cases. Set the starting value of β to $(0, 0)$ for the iteration process in maximization procedure. The results of parameter estimation are listed in the tables below.

Based on the results given above, we can see that

1. The variance of random error term has certain effect on the precision of the estimators. The larger variation may cause less accuracy than the smaller variance. Besides, the variance of estimators seems to have certain linear trend with the increasing of the variance of error. This is easy to verify from the formula given as $A = \beta_0 + \beta_1(w_i + \epsilon)$, where β_1 can be treated as the slope for error random variable in this linear operation, so it is definitely affected by the variation of the error in positive direction;

2. For C-S method and the Gaussian Quadrature method with $K = 20$, the estimators are almost the same. It is basically due to the similar accuracy of the

methods	C.S		G.Q	
σ	0.3	0.8	0.3	0.8
$\widehat{\beta_0}$	-0.01768	-0.02751	-0.01768	-0.02751
$\widehat{\beta_1}$	1.02034	1.02844	1.02034	1.02844
$Bias^2(\beta_0)$	-0.01768	-0.02751	-0.01768	-0.02751
$Bias^2(\beta_1)$	0.02034	0.02844	0.02034	0.02844
$Var(\beta_0)$	0.04171	0.05398	0.04171	0.05398
$Var(\beta_1)$	0.02063	0.03447	0.02063	0.03447
Time(s)	1015.12	1074.68	7839.31	8170.52

Table 5.2: Result comparison of two methods n=500

methods	C.S		G.Q	
σ	0.3	0.8	0.3	0.8
$\widehat{\beta_0}$	0.00332	0.00937	0.00332	0.00937
$\widehat{\beta_1}$	1.00274	1.01002	1.00274	1.01002
$Bias^2(\beta_0)$	0.00332	0.00937	0.00332	0.00937
$Bias^2(\beta_1)$	0.00274	0.01002	0.00274	0.01002
$Var(\beta_0)$	0.02133	0.02655	0.02133	0.02655
$Var(\beta_1)$	0.00922	0.01049	0.00922	0.01049
Time(s)	3482.06	3749.10	22200.13	23692.42

Table 5.3: Result comparison of two methods n=1000

approximation to the logistic normal integral, in which the difference caused after several decimal can not significantly affect the estimating results.

3. The difference between the computing time of the two methods is considerably large, especially when the sample size is large. The C-S method is much faster than the traditional Gaussian Quadrature method. The comparison of the C-S method with other methods than the Gaussian Quadrature method will be conducted in the future.

Chapter 6

Conclusions

The new method and Gaussian Quadrature are both good methods to compute the logistic-normal integral. Applying these two methods to our logistic-regression model with additive Berkson measurement error model involved, the new method behaves more convinced to us to obtain a more precious value closed to the true integral. In this case, the new method is considered to present more efficiency and accuracy during the application of parameter estimation process. Besides, calling FORTRAN subroutine in R is a good technique to improve time efficiency than writing a new R code. It is faster to complete the calculation than the Gaussian quadrature.

Bibliography

[1] A. E. Fridman. (2012). "The Quality of Measurements: A Metrological Reference". *Springer*;

[2] Berkson, J. (1950), "Are there two regressions?". *Journal of American Statistical Association*, 45, 164-180;

[3] Dan Pirjol. (2012). "The logistic-normal integrals and its generalizations". *Journal of Computational and Applied Mathematics*, 237(2013), 460-469;

[4] D. A. Pierce & B. R. Sands. (1975). *Extra-Bernoulli Variation in Binary Data.* Technical Report 46, Department of Statistics, Oregon State University.

[5] E. Demidenko. (2004). *Mixed models: Theory and Application.* Hoboken, N.J: Wiley;

[6] E. Crouch and D. Spiegelman. (1990). "The evaluation of integrals of the form $\int_{-\infty}^{+\infty} f(x)e^{-x^2}dx$: Application to logistic-normal models". *JASA*, 85, 464-469;

[7] Goodwin. E. T. (1949). "The evaluation of integrals of the form $\int_{-\infty}^{+\infty} f(x)e^{-x^2}dx$", *Proceedings of Cambridge Philosophically society*, 45, 241-245;

[8] Harald Niederreiter. (1992). *Random number generation and quasi-monte carlo methods*. Vol 63 of CBMS-NSF Reginal conference series in applied mathematics;

[9] J. S. Buzas., T. D. Tosteson. and L. A. Stefanski. (2005). ”Measurement Error”, *Springer Berlin Heidelberg*, 729-765;

[10] John R. Taylor. (1997). *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. Sausalito, Calif.: University Science Books;

[11] Jorge G., Francis Tuerlinckx, Paul D. Boeck and Ronald Cools. (2006). ”Numeric integration in logistic-normal models”. *Computational Statistics and Data Analysis*, 51(3), 1535-1548;

[12] Kosuke Imai & Teppei Yamamoto. (2010). ”Casual Inference with Differential Measurement Error: Nonparametric Identification and Sensitivity Analysis”. *American Journal of Political Science*, 54(2), 543-560;

[13] Kocis. L. & Whiten. W. (1997). ”Computational investigations of low-descrepency sequences”, *ACM Transactions on Mathematical Software*, 23(2), 266-294;

[14] Lesaffre Emmanuel & Spiessens Bart. (2000). ”Marginalized Multilevel Models and Likelihood Inference-Comments & Rejoinder”. *Institute of Mathematical Statistics*, 15(1), 20-26;

[15] M. A. Zemel’man. (1985). ”Classifying Measurement Errors”. *Measurement Techniques*, 28(6), 471-475;

[16] M. Abramowitz & I. A. Stegun. (1967). *Handbook of Mathematical Functions:*

With Formulas, Graphs, and Mathematical Tables. U.S: Government Printing Office;

[17] N. L. Johnson & S. Kotz. (1970). *Distributions in StatisticsContinuous Univariate Distributions.* University of Michigan: Wiley;

[18] Qing Liu & Donald A. Pierce. (1994). "A note on Gaussian-Hermite Quadrature". *Biometrika*, 81(3), 624-629;

[19] Raymond J. Carroll, David Ruppert, Raymond J. Carroll and A. Stefanski. (2006). *Measurement Error in Nonlinear Models.* Boca Raton, FL: Chapman and hall /CRC;

[20] Satton G. A. & Kupper L. L. (1993). "Inferences about Exposure-Disease Association using Probability of Exposure Information". *Journal of the American Statistical Association*, 88, 200-208;

[21] Topping, J. (1972). *Errors of Observation and Their Treatment(4th ed.).* London: Chapman and Hall;

[22] "Calling NAG FORTRAN subroutines in R". Retrieved from http://www.nag.co.uk/numeric/RunderWindows.asp;

[23] "Calling FORTRAIN subroutines in R on a PC". Retrieved from http://www.stat.sc.edu/ grego/courses/stat740/handout.pcfortran.pdf.

Appendix A

The original Fortran code of C.S method

```
SUBROUTINE GOODWN(INX,INY,N,F)
C
C THIS ROUTINE COMPUTES INTEGRALS J(X,Y;N) AS DEFINED BY
C CROUCH & SPIEGELMAN, JASA, 1990, USING GOODWN'S METHOD
C
C INPUT PARAMETERS:   INX    PARAMETER X OF J(X,Y;N)
C                     INY    PARAMETER Y OF J(X,Y;N)
C                     N      PARAMETER N OF J(X,Y;N)
C OUTPUT              F      F=J(X,Y;N)
C
      IMPLICIT REAL*8 (A-Z)
      INTEGER N,NCOM
      PARAMETER(EPS=1.D-16)
      EXTERNAL LGFT
      COMMON /COMFOF/X,Y,C,NCOM
      ZERO=0.D0
      ONE=1.D0
      TWO=2.D0
      PI=3.14159265358979324D0
      RTPI=dSQRT(PI)
C
C DEFINE X TO GET PARAMETERIZATION FOR J(X,Y;N)
C
      Y=dABS(INY)
      C=DBLE(N)*INX+DBLE(N)**2*Y**2/4.D0
      X=INX+Y**2/TWO*DBLE(N)
      XN=N+1
      NCOM=N+1
C
C SINCE I(X,Y;N)=I(X,-Y;N), SET Y TO dABS(Y)
C
C
C FIRST, FIND H
C
      K=PI/TWO/dSQRT(dLOG(TWO/EPS))
      IF(Y.GT.K)THEN
            H=4.D0*Y*K**2/(Y**2+K**2)
      ELSE
            H=TWO*K
      ENDIF
      T0=MAXPNT(X,Y,XN)
C
C NOW, WE'RE READY TO SUM UP THE INTEGRAL
C
15      CONTINUE
      B=X+Y*T0
      A=-T0**2+C
      IF(B*XN.GT.30.D0)THEN
            F=dEXP(A-XN*B)/(ONE+dEXP(-B))**(N+1)
      ELSE
            F=dEXP(-T0**2+C)/(ONE+dEXP(X+Y*T0))**(N+1)
      ENDIF
      IF(F.EQ.ZERO)THEN
            RETURN
      ENDIF
C
C SUM FROM MIDDLE UP
C
      K=ZERO
20      CONTINUE
      K=K+ONE
C
C THIS FORM OF THE ARITHMETIC OPERATION REDUCES CHANCES OF OVERFLOW
C FOR CASES WHERE K GETS LARGE
C
      A=C-(T0+K*H)**2
      B=X+Y*(T0+K*H)
         iF(b*xn.gt.30.d0)then
      TERM=dEXP(A-(N+1)*B)/(ONE+dEXP(-B))**(N+1)
         else
           term=dexp(a)/(one+dexp(b))**(n+1)
         endif
      F=F+TERM
      IF(dABS(TERM/F).GT.EPS)GO TO 20
      K=ZERO
30      CONTINUE
C
C SUM FROM MIDDLE DOWN
C
      K=K-ONE
      A=C-(T0+K*H)**2
      B=X+Y*(T0+K*H)
      IF(B*XN.GT.30.D0)THEN
            TERM=dEXP(A-XN*B)/(ONE+dEXP(-B))**(N+1)
      ELSE
            TERM=dEXP(A)/(ONE+dEXP(B))**(N+1)
      ENDIF
      F=F+TERM
      IF(dABS(TERM/F).GT.EPS)GO TO 30
      F=H*F/RTPI
      RETURN
      END
C
C
C
C
      REAL FUNCTION MAXPNT*8(X,Y,N)
      IMPLICIT REAL*8 (A-Z)
```

```
 INTEGER COMN
      EXTERNAL LGFT
      COMMON /COMFOF/COMX,COMY,COMC,COMN
      ONE=1.D0
      TWO=2.D0
            XGUESS=-DBLE(COMN)*COMY/TWO
C
C erset is an imsl routine - delete if you find the minimum of function LGFT
C some other way
C
         CALL ERSET(5,-1,0)
      CALL ERSET(4,-1,0)
      IF(COMY.LT.0.D0)THEN
            LB=0.D0
            UB=10.D0*XGUESS
      ELSE
            LB=10.D0*XGUESS
            UB=0.D0
      ENDIF
C
C duvmif is an imsl routine which finds the minimum of a function. If imsl is
C not available to you, you must find another equivalent function, or write
C one yourself. Numerical recipes has some good choices at very low cost.
C
      CALL DUVMIF(LGFT,XGUESS,1.D0,1.d6,1.D-6,100,T0)
      CALL ERSET(5,1,2)
      CALL ERSET(4,1,2)
      MAXPNT=T0
                  RETURN
      END
C
C
      REAL FUNCTION LGFT*8(T)
      COMMON /COMFOF/X,Y,C,N
      REAL*8 T,X,Y,C,a,b
      INTEGER N
      ONE=1.D0
      A=C-T**2
      B=X+Y*T
      IF(B.GT.30.D0)THEN
            LGFT=A-DBLE(N)*B
      ELSE
            LGFT=A-DBLE(N)*dLOG(ONE+dEXP(B))
      ENDIF
      LGFT=-LGFT
      RETURN

      END
```

Appendix B

Abscissas and weight factors for Hermite Integration

Table 25.10 ABSCISSAS AND WEIGHT FACTORS FOR HERMITE INTEGRATION

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \qquad \int_{-\infty}^{\infty} g(x)dx \approx \sum_{i=1}^{n} w_i e^{x_i^2} g(x_i)$$

Abscissas $= \pm x_i$ (Zeros of Hermite Polynomials) Weight Factors $= w_i$

$\pm x_i$	w_i	$w_i e^{x_i^2}$
	$n=2$	
0.70710 67811 86548	(-1)8.86226 92545 28	1.46114 11826 611
	$n=3$	
0.00000 00000 00000	(0)1.18163 59006 04	1.18163 59006 037
1.22474 48713 91589	(-1)2.95408 97515 09	1.32393 11752 136
	$n=4$	
0.52464 76232 75290	(-1)8.04914 09000 55	1.05996 44828 950
1.65068 01238 85785	(-2)8.13128 35447 25	1.24022 58176 958
	$n=5$	
0.00000 00000 00000	(-1)9.45308 72048 29	0.94530 87204 829
0.95857 24646 13819	(-1)3.93619 32315 22	0.98658 09967 514
2.02018 28704 56086	(-2)1.99532 42059 05	1.18148 86255 360
	$n=6$	
0.43607 74119 27617	(-1)7.24629 59522 44	0.87640 13344 362
1.33584 90740 13697	(-1)1.57067 32032 29	0.93558 05576 312
2.35060 49736 74492	(-3)4.53000 99055 09	1.13690 83326 745
	$n=7$	
0.00000 00000 00000	(-1)8.10264 61755 68	0.81026 46175 568
0.81628 78828 58965	(-1)4.25607 25261 01	0.82868 73032 836
1.67355 16287 67471	(-2)5.45155 82819 13	0.89718 46002 252
2.65196 13568 35233	(-4)9.71781 24509 95	1.10133 07296 103
	$n=8$	
0.38118 69902 07322	(-1)6.61147 01255 82	0.76454 41286 517
1.15719 37124 46780	(-1)2.07802 32581 49	0.79289 00483 864
1.98165 67566 95843	(-2)1.70779 83007 41	0.86675 26065 634
2.93063 74202 57244	(-4)1.99604 07221 14	1.07193 01442 480
	$n=9$	
0.00000 00000 00000	(-1)7.20235 21560 61	0.72023 52156 061
0.72355 10187 52838	(-1)4.32651 55900 26	0.73030 24527 451
1.46855 32892 16668	(-2)8.84745 27394 38	0.76460 81250 946
2.26658 05845 31843	(-3)4.94362 42755 37	0.84175 27014 787
3.19099 32017 81528	(-5)3.96069 77263 26	1.04700 35809 767

$\pm x_i$	w_i	$w_i e^{x_i^2}$
	$n=10$	
0.34290 13272 23705	(- 1)6.10862 63373 53	0.68708 18539 513
1.03661 08297 89514	(- 1)2.40138 61108 23	0.70329 63231 049
1.75668 36492 99882	(- 2)3.38743 94455 48	0.74144 19319 436
2.53273 16742 32790	(- 3)1.34364 57467 81	0.82066 61264 048
3.43615 91188 37738	(- 6)7.64043 28552 33	1.02545 16913 657
	$n=12$	
0.31424 03762 54359	(- 1)5.70135 23626 25	0.62930 78743 695
0.94778 83912 40164	(- 1)2.60492 31026 42	0.63962 12320 203
1.59768 26351 52605	(- 2)5.16079 85615 88	0.66266 27732 669
2.27950 70805 01060	(- 3)3.90539 05846 29	0.70522 03661 122
3.02063 70251 20890	(- 5)8.57368 70435 88	0.78664 39394 633
3.88972 48978 69782	(- 7)2.65855 16843 56	0.98969 90470 923
	$n=16$	
0.27348 10461 3815	(- 1)5.07929 47901 66	0.54737 52050 378
0.82295 14491 4466	(- 1)2.80647 45852 85	0.55244 19573 675
1.38025 85391 9888	(- 2)8.38100 41398 99	0.56321 78290 882
1.95178 79909 1625	(- 2)1.28803 11535 51	0.58124 72754 009
2.54620 21578 4748	(- 4)9.32284 00862 42	0.60973 69582 560
3.17699 91619 7996	(- 5)2.71186 00925 38	0.65575 56728 761
3.86944 79048 6012	(- 7)2.32098 08448 65	0.73824 56222 777
4.68873 89393 0582	(-10)2.65480 74740 11	0.93687 44928 841
	$n=20$	
0.24534 07083 009	(- 1)4.62243 66960 06	0.49092 15006 667
0.73747 37285 454	(- 1)2.86675 50536 28	0.49384 33852 721
1.23407 62153 953	(- 1)1.09017 20602 00	0.49992 08713 363
1.73853 77121 166	(- 2)2.48105 20887 46	0.50967 90271 175
2.25497 40020 893	(- 3)3.24377 33422 38	0.52408 03509 486
2.78880 60584 281	(- 4)2.28338 63601 63	0.54485 17423 644
3.34785 45673 832	(- 6)7.80255 64785 32	0.57526 24428 525
3.94476 40401 156	(- 7)1.08606 93707 69	0.62227 86961 914
4.60368 24495 507	(-10)4.39934 09922 73	0.70433 29611 769
5.38748 08900 112	(-13)2.22939 36455 34	0.89859 19614 532

Compiled from H. E. Salzer, R. Zucker, and R. Capuano, Table of the zeros and weight factors of the first twenty Hermite polynomials, J. Research NBS **48,** 111–116, 1952, RP2294 (with permission).

Table 25.11 COEFFICIENTS FOR FILON'S QUADRATURE FORMULA

θ	α	β	γ
0.00	0.00000 000	0.66666 667	1.33333 333
0.01	0.00000 004	0.66668 000	1.33332 000
0.02	0.00000 036	0.66671 999	1.33328 000
0.03	0.00000 120	0.66678 664	1.33321 334
0.04	0.00000 284	0.66687 990	1.33312 001
0.05	0.00000 555	0.66699 976	1.33300 003
0.06	0.00000 961	0.66714 617	1.33285 340
0.07	0.00001 524	0.66731 909	1.33268 012
0.08	0.00002 274	0.66751 844	1.33248 020
0.09	0.00003 237	0.66774 417	1.33225 365
0.1	0.00004 438	0.66799 619	1.33200 048
0.2	0.00035 354	0.67193 927	1.32800 761
0.3	0.00118 467	0.67836 065	1.32137 184
0.4	0.00278 012	0.68703 909	1.31212 154
0.5	0.00536 042	0.69767 347	1.30029 624
0.6	0.00911 797	0.70989 111	1.28594 638
0.7	0.01421 151	0.72325 813	1.26913 302
0.8	0.02076 156	0.73729 136	1.24992 752
0.9	0.02884 683	0.75147 168	1.22841 118
1.0	0.03850 188	0.76525 831	1.20467 472

See **25.4.47.**

Appendix C

Iteration R code example

```
dyn.load("c:/mingw/programs/goodwin.dll")

zhaoyi<-function(x,y,n)
{
ans=.C("goodwin_",INX=as.real(x),INY=as.real(y),N=as.integer(n),F=as.real(F))
return(ans$F)
}

sigma<-0.3
beta0<-0
beta1<-1
E<-matrix(,500,500)
Z<-matrix(,500,500)
P<-matrix(,500,500)
W<-matrix(,500,500)
Y<-matrix(,500,500)
PP<-matrix(,500,500)
PPP<-matrix(,500,500)
PPPP<-matrix(,500,500)
PPPPP<-matrix(,500,500)
for(i in 1:500)
{
E[,i]<-rnorm(500,0,sigma)
Z[,i]<-runif(500,0,4)
}
W=E+Z
for(i in 1:500)
for(j in 1:500)
{
P[,i]<-exp(beta0+beta1*W[,i])/(1+exp(beta0+beta1*W[,i]))
Y[j,i]<-rbinom(1,1,P[j,i])
PP[j,i]<-exp(W[j,i])/(1+exp(W[j,i]))
PPP[j,i]<-log(P[j,i]/(1-P[j,i]))
PPPP[j,i]<-zhaoyi(-(beta0+beta1*Z[j,i]),-sigma*sqrt(2)*beta1,0)
PPPPP[j,i]<-log(PPPP[j,i]/(1-PPPP[j,i]))
}

### Gaussian n=20 ###

sk<-numeric(20)
sk[1]<-0.2453407083009
sk[2]<-0.7374737258454
sk[3]<-1.2340762153953
sk[4]<-1.7385377121166
sk[5]<-2.2549740020893
sk[6]<-2.7888060584281
sk[7]<-3.3478545673832
sk[8]<-3.9447640401156
sk[9]<-4.6036824495507
```

```
sk[10]<-5.3874808900112
sk[11]<--0.2453407083009
sk[12]<--0.7374737258454
sk[13]<--1.2340762153953
sk[14]<--1.7385377121166
sk[15]<--2.2549740020893
sk[16]<--2.7888060584281
sk[17]<--3.3478545673832
sk[18]<--3.9447640401156
sk[19]<--4.6036824495507
sk[20]<--5.3874808900112

ck<-numeric(20)
ck[1]<-4.622436696006*10^(-1)
ck[2]<-2.866755053628*10^(-1)
ck[3]<-1.090172060200*10^(-1)
ck[4]<-2.481052088746*10^(-2)
ck[5]<-3.243773342238*10^(-3)
ck[6]<-2.283386360163*10^(-4)
ck[7]<-7.802556478532*10^(-6)
ck[8]<-1.086069370769*10^(-7)
ck[9]<-4.399340992273*10^(-10)
ck[10]<-2.229393645534*10^(-13)
ck[11]<-4.622436696006*10^(-1)
ck[12]<-2.866755053628*10^(-1)
ck[13]<-1.090172060200*10^(-1)
ck[14]<-2.481052088746*10^(-2)
ck[15]<-3.243773342238*10^(-3)
ck[16]<-2.283386360163*10^(-4)
ck[17]<-7.802556478532*10^(-6)
ck[18]<-1.086069370769*10^(-7)
ck[19]<-4.399340992273*10^(-10)
ck[20]<-2.229393645534*10^(-13)

ptm<-proc.time()
u2<-matrix(,2,500)
for(k in 1:500)
{
model1<-function(beta,Z,Y)
{
I<-matrix(,500,20)
for(i in 1:500)
for(j in 1:20)
{
I[i,j]<-(1/sqrt(pi))*ck[j]/(1+exp(-(beta[1]+beta[2]*Z[i,k])-sigma*sqrt(2)*beta[2]*sk[j]))
}
II<-numeric(500)
for(i in 1:500)
{
```

```
II[i]<-sum(I[i,])
}
logit<-sum(Y[,k]*log(II)+(1-Y[,k])*log(1-II))
return(-logit)
}
beta<-c(0,0)
mle1<-optim(beta,model1,Y=Y,Z=Z)
u2[,k]<-c(mle1$par)
}
proc.time()-ptm
sprintf("%0.15f", mean(u2[1,])
sprintf("%0.15f", mean(u2[2,])

### new method ###

ptm<-proc.time()
u1<-matrix(,2,500)
for(j in 1:500)
{
model2<-function(beta,Z,Y)
{
logit3<-numeric(500)
for(i in 1:500)
{
logit3[i]<-Y[i,j]*log(zhaoyi(-(beta[1]+beta[2]*Z[i,j]),-sigma*sqrt(2)*beta[2],0))+(1-Y[i,j])*log(1-
zhaoyi(-(beta[1]+beta[2]*Z[i,j]),-sigma*sqrt(2)*beta[2],0))
}
logit1<-sum(logit3)
return(-logit1)
}
beta<-c(0,0)
mle2<-optim(beta,model2,Y=Y,Z=Z)
u1[,j]<-c(mle2$par)
}
proc.time()-ptm

mean(u1[1,])
mean(u1[2,])
var(u1[1,])
var(u1[2,])
mean(u2[1,])
mean(u2[2,])
var(u2[1,])
var(u2[2,])

sprintf("%0.15f", mean(u1[1,]))
```