Optimal Designs for The Panel Mixed Logit Model

by

Wei Zhang

(Under the Direction of
Abhyuday Mandal and John Stufken)

Abstract

We discuss optimal designs for the panel mixed logit model. The panel mixed logit model is usually used for the analysis of discrete choice experiments. The information matrix used in design criteria does not have a closed form expression and it is computationally difficult to evaluate the information matrix numerically. We derive the information matrix and use the obtained form to propose three methods to approximate the information matrix. The approximations are compared to the information matrix in simulations to see whether the design criteria based on them can yield similar orderings of designs as the criteria based on the information matrix. We also propose three alternatives to the information matrix based on approximate analysis methods for the generalized linear mixed models given the panel mixed logit model is a special case of the generalized linear mixed

models. The alternatives are used in computer search to find optimal designs and compared based on the efficiencies of the best designs and time needed to find those designs.

INDEX WORDS: Optimal designs, discrete choice models, panel mixed logit model, locally optimal designs

Optimal Designs for The Panel Mixed Logit Model

by

Wei Zhang

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2018

Optimal Designs for
The Panel Mixed Logit Model

by

Wei Zhang

Major Professors:	Abhyuday Mandal John Stufken
Committee:	Dan Hall Gauri Datta Jaxk Reeves William McCormick

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2018

Acknowledgments

In Georgia, I got a lot of help and support from fellow graduate students, faculty and staff in the department of Statistics. Thanks to my advisors Dr. Mandal and Dr. Stufken for their time, effort and encouragement. Thanks to my families Shuang Zhang, Jing Wei and Chengchang Liu.

Contents

Acknowledgments **iv**

List of Figures **viii**

List of Tables **x**

1 Introduction to Discrete Choice Experiments **1**

1.1 Introduction 1

1.2 Review of Discrete Choice Models 6

1.3 Review of Designs for Discrete Choice Models 13

1.4 Review of Generalized Linear Mixed Models 30

1.5 Summary and Discussions 31

1.6 References 33

2 Methods for Analysis of Generalized Linear Mixed Models **36**

2.1 Introduction 36

2.2 Different Methods for Estimation 39

2.3 Summary and Discussions 64

2.4 References 65

3 Information Matrix for Panel Mixed Logit Model **68**

3.1 Introduction 68

3.2 Model, Information Matrix and Design Criteria 72

3.3 Approximation of the Information Matrix 77
3.4 Simulation 87
3.5 Discussion and Conclusion 103
3.6 Appendix 105
3.7 References 116

4 Optimal Designs for the Panel Mixed Logit Model 120
4.1 Introduction 120
4.2 Maximum Likelihood Method 123
4.3 PQL and MQL Applied to Panel Mixed Logit Model 126
4.4 Method of Simulated Moments (MSM) Applied to Panel Mixed Logit Model 130
4.5 Searching for Optimal Designs 133
4.6 Second Look at the Search for Optimal Designs 139
4.7 Revisiting the Example 146
4.8 Discussion and Conclusion 146
4.9 Appendix 148
4.10 References 153

5 Conclusion 156

6 Appendix: Code for Chapter 3 and 4 159
6.1 Code for Laplace Approximation 159
6.2 Code for MSM 170
6.3 Code for PQL 179

6.4 Code for MQL 187

List of Figures

3.1 Comparisons of the three methods with A-optimality when the response accuracy is high and the respondent heterogeneity is high. . 91

3.2 Comparisons of the three methods with D-optimality when the response accuracy is high and the respondent heterogeneity is high. . 92

3.3 Comparisons of the three methods with A-optimality when the response accuracy is high and the respondent heterogeneity is low. . . 93

3.4 Comparisons of the three methods with D-optimality when the response accuracy is high and the respondent heterogeneity is low. . . 94

3.5 Comparisons of the three methods with A-optimality when the response accuracy is low and the respondent heterogeneity is high. . . 95

3.6 Comparisons of the three methods with D-optimality when the response accuracy is low and the respondent heterogeneity is high. . . 96

3.7 Comparisons of the three methods with A-optimality when the response accuracy is low and the respondent heterogeneity is low. . . 97

3.8 Comparisons of the three methods with D-optimality when the response accuracy is low and the respondent heterogeneity is low. . . 98

3.9 Relative difference (in %) between values from a sample size on the x-axis and the values from the largest sample size for the $3^2/5/4$ case with $b = (-3, 0, -3, 0)'$ and $\sigma = (3, 3, 3, 3)'$. 102

4.1 The 1000 designs from the coordinate exchange algorithm using criterion based on PQL and A-optimality for design problem with 5 attributes with 3 levels and out of which 3 attributes are random. 141

List of Tables

1.1 Six Attributes to be Used in an Experiment to Compare Pizza . . . 3

1.2 One Choice Set in an Experiment to Compare Pizza 3

1.3 The possible design 17

1.4 Optimal Designs for Multinomial Logit Model 20

1.5 Optimal Designs for Mixed Logit Model Assuming Independence . . 24

1.6 Optimal Design for Mixed Logit Model Assuming Correlation . . . 29

3.1 Correlations between the three methods 99

3.2 Time for evaluating 100 designs using the three methods 101

4.1 Results for $3^4/2/9$ 136

4.2 Results for $3^4/4/5$ 137

4.3 Reduced number of runs for $3^4/2/9$ 139

4.4 Robustness 140

4.5 3^5 with 2 random attributes 142

4.6 3^5 with 3 random attributes 144

4.7 3^5 with 4 random attributes 144

4.8 3^5 with 5 random attributes 145

4.9 $3^5/2/10$ with 4 random attributes when $b = (3, 0, 3, 0, 3, 0, 3, 0, 3, 0)'$ and $\sigma = 3 \cdot 1_8$. 147

Chapter 1

Introduction to Discrete Choice Experiments

1.1 Introduction

People make choices all the time, but how they make the choices is often not revealed. To understand the choice behaviors is very important, for some of them have great influence on the decisions that companies or governments make. Several models, which are known as the discrete choice models, have been proposed to analyze choice data. They can explain the choice behaviors by the influential factors, like features of the products and socio-economic status of the respondents. Discrete choice models are widely used in marketing, transportation, health care, and many other areas.

The choice data can be either observational from sources like supermarket scanners, or experimental from discrete choice experiments. The former only contain alternatives that are currently available, for products that are currently in the market; the latter can contain alternatives that are not available yet, for products that may be introduced to the market. Since choice experiments can simulate choice situations about how people make their choices, they play an important role in the study of choice behaviors. Researchers can use different experiments according to their research interests.

In a typical choice experiment, each respondent is shown one or more choice sets. A choice set is a hypothetical choice situation that consists of several hypothetical alternatives. The alternatives are described by the levels of the attributes. The following is an example of a simple choice experiment that compares pizza. This example is taken from Street and Burgess (2007) and is an example of a paired comparison, in which two alternatives are compared at a time. Table 1.1 shows the attributes and their levels. Table 1.2 shows a choice set of two alternatives. There can be many choice sets as in a choice experiment, one such example is given in Table 1.2 . The researcher needs to decide which choice sets to use in an experiment. The possible alternatives are the level combinations of the full factorial design with the attributes as its factors; for this example there are $2^6 = 64$ possible alternatives. The possible choice sets are all sets consisting of two of the alternatives; for this example there are $\binom{64}{2} = 2016$ possible choice sets. In this example, for a choice set and a respondent, the response is the pizza, A or B, which the respondent chooses. Also, a choice set can contain more than two alternatives, but not so many that a respondent cannot differentiate from.

Which choice sets will a respondent entering the experiment see? In the previous example, there are 2016 different choice sets; a respondent cannot possibly finish a questionnaire consisting of all 2016 choice sets. In practice, a respondent only sees a small subset of all possible choice sets. For different respondents, should they see the same set of choice sets or different sets of choice sets? The design problem is to decide which choice sets to include in the experiment and how to assign the choice sets to the respondents.

Table 1.1: Six Attributes to be Used in an Experiment to Compare Pizza

Attributes	Attribute levels
Pizza type	Traditional Courmet
Type of Crust	Thick Thin
Ingredients	All fresh Some tinned
Size	Small only Three sizes
Prices	$ 17 $ 13
Delivery time	30 minutes 45 minutes

Table 1.2: One Choice Set in an Experiment to Compare Pizza

	Option A	Option B
Pizza type	Traditional	Gourmet
Type of Crust	Thick	Thin
Ingredients	All fresh	Some tinned
Size	Small only	Small Only
Prices	$ 17	$ 13
Delivery time	30 minutes	30 minutes
Suppose that you have already narrowed down your choice of take-out pizza to the two alternatives above. Which of these two would your choose?		
	Option A	Option B

For the discrete choice models, the responses are categorical, so that linear models are not appropriate. Instead, special nonlinear models, called discrete choice modes, have been developed for this problem. For nonlinear models, information matrices depend on the unknown parameters. The design criteria are usually functions of the information matrices, so the values of the parameters are needed to get the optimal designs. Khuri et al. (2006) give a review of the designs for the generalized linear models, and provide several methods for dealing with the dependence problem. Their discussion includes the following methods:

1. Locally optimal design is the design that optimizes some design criterion for given values of the parameters.

2. Bayesian optimal design is the optimal design obtained under some Bayesian design criterion, which requires the prior distribution of the parameters. For a review of Bayesian designs, see Chaloner and Verdinelli (1995).

3. Sequential design is obtained from a iterative process of updating values of the parameters and adding designs, starting with an initial design.

Khuri et al. (2006) does not discuss minimax design criteria. A minimax design is the design that optimizes the worst criterion value obtained over a range of values for the parameters.

The above methods have been employed to find designs for discrete choice models. Under the null hypothesis that the respondents are indifferent about the alternatives offered, in which case the coefficients for the attributes are all set to zeros, Street and Burgess (2007) find the locally optimal designs. When

information about preference of the respondents is available, Huber and Zwerina (1996) argue that locally optimal designs constructed with appropriate nonzero coefficients are more efficient. Using assumed values of the parameters, Sándor and Wedel (2002), Bliemer and Rose (2010), and Liu and Arora (2011) construct locally optimal designs for different discrete choice models that have not been previously addressed. Sándor and Wedel (2001), Yu et al. (2009), and Vermeulen et al. (2008) apply the Bayesian design approach to different discrete choice models, for which the prior distribution of the parameters needs to be specified. Under the Bayesian framework, Yu et al. (2011) propose an efficient individually adapted sequential Bayesian (IASB) approach, which finds a sequential design specific for each individual in the experiment.

In the literature, one set of choice sets is usually used for all the respondents. The set is sometimes blocked into subsets, in which case a respondent only sees one of the subsets. "Showing subjects more than one choice set is economical, and in practice, most researchers almost always show multiple choice sets to each subject." (Kuhfeld (2006)). When more than one choice is observed from a respondent, these choices may be correlated. However, in the literature, designs are often constructed for models that assume independence for the choices in different choice sets made by the same respondent. Bliemer and Rose (2010) construct a locally optimal design for a model that takes into account the correlation among the responses within subjects. One difficulty in finding such design is the complicated structure of the information matrix. We find that the expression for the information matrix could be simplified. We will give the simplified expression and the simulation

results for the optimal design using the simplified expression.

1.2 Review of Discrete Choice Models

1.2.1 Multinomial Logit Model

Discrete choice models are usually derived under an assumption of utility-maximizing behavior by a respondent. Suppose a respondent n faces a choice set containing J alternatives and his utilities are represented by the sum of systematic and random component (Thurstone(1927); Manski(1977)), then his/her utility for the jth alternative is given by

$$U_{nj} = V_{nj} + \varepsilon_{nj},\ j = 1, \ldots, J,$$

where V_{nj} is the representative utility that is a function of the observed factors and ε_{nj} is treated as a random variable that captures the influence of the unobservable factors which cannot be included in V_{nj}. The observed factors may include attributes of the alternatives and socio-economic characteristics of the respondents. The respondent will choose the alternative with the greatest utility, i.e., he/she will choose alternative j in the choice set if $U_{nj} > U_{ni}, \forall i \neq j$. For a situation when respondent n chooses one of the J alternatives in a choice set, the response is given by $Y_n = (Y_{n1}, \ldots, Y_{nJ})'$, with $Y_{nj} = 1$ if the respondent chooses alternative j, and 0 otherwise. The probability that respondent n chooses alternative j is

$$
\begin{aligned}
P(Y_{nj} = 1) &= P(U_{nj} > U_{ni}, \forall i \neq j) \\
&= P(V_{nj} + \varepsilon_{nj} > V_{ni} + \varepsilon_{ni}, \forall i \neq j) \\
&= P(\varepsilon_{ni} < V_{nj} - V_{ni} + \varepsilon_{nj}, \forall i \neq j).
\end{aligned} \tag{1.1}
$$

The multinomial logit model is obtained by assuming ε_{nj}'s to be independent and identically distributed standard Gumbel. With this assumption, the choice probability in (1.1) has a closed-form expression. The density function of the standard Gumbel distribution is given by

$$f(\varepsilon) = e^{-\varepsilon} e^{-e^{-\varepsilon}},$$

and the cumulative distribution function is given by

$$F(\varepsilon) = e^{-e^{-\varepsilon}}.$$

So the probability that respondent n chooses alternative j is

$$
\begin{aligned}
P(Y_{nj}=1) &= P(\varepsilon_{ni} < V_{nj} - V_{ni} + \varepsilon_{nj}, \forall i \neq j) \\
&= \int_{-\infty}^{\infty} P(\varepsilon_{ni} < V_{nj} - V_{ni} + \varepsilon_{nj}, \forall i \neq j | \varepsilon_{nj}) f(\varepsilon_{nj}) d\varepsilon_{nj} \\
&= \int_{-\infty}^{\infty} \left(\prod_{i \neq j} e^{-e^{-(V_{nj} - V_{ni} + \varepsilon_{nj})}} \right) e^{-\varepsilon_{nj}} e^{-e^{-\varepsilon_{nj}}} d\varepsilon_{nj} \\
&= \int_{-\infty}^{\infty} \left(\prod_{i} \exp(-e^{-(V_{nj} - V_{ni} + \varepsilon_{nj})}) \right) e^{-\varepsilon_{nj}} d\varepsilon_{nj} \\
&= \int_{-\infty}^{\infty} \exp\left(-\sum_{i} e^{-(V_{nj} - V_{ni} + \varepsilon_{nj})} \right) e^{-\varepsilon_{nj}} d\varepsilon_{nj} \\
&= \int_{0}^{\infty} \exp\left(Z_{nj} \sum_{i} e^{-(V_{nj} - V_{ni})} \right), \text{ let } Z_{nj} = e^{-\varepsilon_{nj}} \\
&= \frac{\exp(V_{nj})}{\sum_{i=1}^{J} \exp(V_{ni})}.
\end{aligned}
$$

Let x_{nj} be the coded levels of the attributes of alternative j for respondent n, the representative utility V_{nj} is usually modeled using linear predictor $V_{nj} = x'_{nj}\beta$, where β be the corresponding coefficient vector. Details of coding of x_{nj} will be explained in Section 1.3. Then the probability of respondent n choosing alternative j is

$$
P(Y_{nj} = 1 | \beta) = \frac{\exp(x'_{nj}\beta)}{\sum_{i=1}^{J} \exp(x'_{ni}\beta)}.
$$

For a situation when respondent n chooses one of the J alternatives in a choice set, the data would look like $y_n = (y_{n1}, \ldots, y_{nJ})'$, with $y_{nj} = 1$ if the respondent chooses alternative j, and 0 otherwise. Clearly, $\sum_{j=1}^{J} y_{nj} = 1$ and probability of

observing y_n is

$$P(Y_n = y_n|\beta) = \prod_{j=1}^{J} P(Y_{nj} = 1|\beta)^{y_{nj}}.$$

Suppose there are N respondents, let $Y = (Y_1', Y_2', \ldots, Y_N')'$ be the responses from the N respondents. Since ε_{nj}'s are i.i.d. standard Gumbel, the probability of observing $y = (y_1', y_2', \ldots, y_N')'$ is

$$P(Y = y|\beta) = \prod_{n=1}^{N} \prod_{j=1}^{J} P(Y_{nj} = 1|\beta)^{y_{nj}}$$

If a respondent is shown more than one choice set, the choices observed in different choice sets are usually assumed to be independent.

1.2.2 Cross-sectional Mixed Logit Model

In the multinomial logit model, the same β is used for all the respondents, so β represents the average preference of population. But the preference often varies within the population, mixed logit model accommodates the heterogeneity in preference by assuming β to be a random effect. Let β_n be the random coefficient for respondent n, then the representative utility for respondent n is $V_{nj} = x_{nj}'\beta_n$. Given β_n, the conditional probability of respondent n choosing alternative j is given by

$$P(Y_{nj} = 1|\beta_n) = \frac{\exp(x_{nj}'\beta_n)}{\sum_{i=1}^{J} \exp(x_{ni}'\beta_n)}.$$

Let $f(\beta|\theta)$ be the distribution of the random coefficients, where θ is the vector of population parameters. The unconditional choice probability is given by

$$P(Y_{nj} = 1|\theta) = \int P(Y_{nj} = 1|\beta_n) f(\beta_n|\theta)\, \mathrm{d}\beta_n.$$

For a situation when respondent n chooses one of the J alternatives in a choice set, the data would look like $y_n = (y_{n1}, \ldots, y_{nJ})'$, with $y_{nj} = 1$ if the respondent chooses alternative j, and 0 otherwise. Clearly, here $\sum_{j=1}^{J} y_{nj} = 1$ and probability of observing y_n is

$$P(Y_n = y_n|\theta) = \prod_{j=1}^{J} P(Y_{nj} = y_{nj}|\theta)^{y_{nj}}.$$

Suppose there are N respondents, let $Y = (Y_1', Y_2', \ldots, Y_N')'$ be the responses from the N respondents. Since ε_{nj}'s are i.i.d. standard Gumbel, the probability of observing $y = (y_1', y_2', \ldots, y_N')'$ is

$$P(Y = y|\theta) = \prod_{n=1}^{N} \prod_{j=1}^{J} P(Y_{nj} = 1|\theta)^{y_{nj}}.$$

If a respondent is shown with more than one choice set, cross-sectional mixed legit model assumes that the choices observed in different choice sets are independent.

1.2.3 Panel Mixed Logit Model

If a respondent is shown with more than one choice sets, the choices observed in different choice sets should be correlated. Panel mixed logit model can account

for this correlation by assuming that the random effect for respondent n, β_n, is constant over the choice sets. Given β_n, the calculation of the choice probability in the S choice set is the same as the calculation of the choice probability in S choice sets for multinomial logit model with regression coefficient β_n. Suppose each respondent is shown with S choice sets composed by J alternatives, let $Y_{ns} = (Y_{ns1}, Y_{n12}, \ldots, Y_{nsJ})'$ be the choice made in choice set s which satisfies $\sum_{j=1}^{J} Y_{nsj} = 1$, and let $Y_n = (Y'_{n1}, Y'_{n2}, \ldots, Y'_{nS})'$ be the sequence of choices in the S choice sets from respondent n. Given β_n, the choice probability of observing $y_n = (y_{n11}, y_{n12}, \ldots, y_{nSJ})'$ is

$$P(Y_n = y_n|\beta_n) = \prod_{s=1}^{S}\prod_{j=1}^{J} P(Y_{nsj} = 1|\beta_n)^{y_{nsj}} = \prod_{s=1}^{S}\prod_{j=1}^{J} \Big(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)}\Big)^{y_{nsj}},$$

where x_{nsj} is the coded levels of the attributes of alternative j in choice set s for respondent n.

Let $f(\beta|\theta)$ be the distribution of the random coefficients, where θ is the vector of population parameters. The unconditional choice probability is given by

$$P(Y_n = y_n|\theta) = \int \prod_{s=1}^{S}\prod_{j=1}^{J} P(Y_{nsj} = 1|\beta_n)^{y_{nsj}} f(\beta_n|\theta)\,\mathrm{d}\beta_n.$$

From the above expression, we can see that the observations in different choice sets are independent given β_n, but are not independent in general.

Suppose there are N respondents, let $Y = (Y'_1, Y'_2, \ldots, Y'_N)'$ be the responses

from the N respondents, the probability of observing $y = (y_1', y_2', \ldots, y_N')'$ is

$$P(Y = y|\theta) = \prod_{n=1}^{N} P(Y_n = y_n|\theta).$$

1.2.4 Nested Logit Model

With multinomial logit model in 1.2.1, the ratio of the probabilities of choosing any two alternatives is independent of all other alternatives in the choice set. The ratio of probabilities of choosing alternative j and j' is given by

$$\frac{P(Y_{nj} = 1)}{P(Y_{nj'} = 1)} = \frac{\exp(V_{nj})}{\exp(V_{nj'})},$$

which does not include information about the other alternatives nor change if any of the other alternatives are added or removed from the choice set. When an alternative similar to j is added or removed from the choice set, the probability of choosing j may decrease or increase more than that of j' and the above ratio may change. The nested logit model partitions the J alternatives in a choice set into K subsets $B_1, \ldots, B_K$ with similar alternatives. The unobserved part of the utility, $\varepsilon_n = (\varepsilon_{n1}, \ldots, \varepsilon_{nJ})'$, is assumed to have cumulative distribution function as

$$\exp\Big(- \sum_{k=1}^{K} \big(\sum_{j \in B_k} e^{-\varepsilon_{nj}/\lambda_k} \big)^{\lambda_k} \Big),$$

where each ε_{nj}, $j = 1, \ldots, J$, follows univariate extreme value distribution, but ε_{nj}'s within a subset B_k are correlated, and λ_k is the degree of independence of

the alternatives within subset B_k, which reduces the model to the Multinomial Logit model if $\lambda_k = 1$ for all k. The probability of choosing alternative j in subset B_k is given by

$$P_{nj} = \frac{e^{V_{nj}/\lambda_k}(\sum_{i\in B_k} e^{V_{ni}/\lambda_k})^{\lambda_k - 1}}{\sum_{l=1}^{K}(\sum_{i\in B_l} e^{V_{ni}/\lambda_l})^{\lambda_l}}.$$

Now for j and j' in two subsets, the ratio of $P(Y_{nj} = 1)/P(Y_{nj'} = 1)$ also depends on the other alternatives in these two subsets.

1.3 Review of Designs for Discrete Choice Models

Suppose there are N respondents in the choice experiment, each respondent is presented with S choice sets of size J. For respondent n, the design matrix X_n is given by

$$X_n = (\underbrace{x_{n11}, x_{n12}, \ldots, x_{n1J}}_{\text{choice set 1}}, \underbrace{x_{n21}, x_{n22}, \ldots, x_{n2J}}_{\text{choice set 2}}, \ldots, \underbrace{x_{nS1}, x_{nS2}, \ldots, x_{nSJ}}_{\text{choice set } S})',$$

where x'_{nsj} is coded levels of the attributes of alternative j in choice set s for respondent n. The design matrix for choice set s is $X_{ns} = (x_{ns1}, x_{ns2}, \ldots, x_{nsJ})'$. The choice sets presented to the respondents are not necessary the same sets for each individual. For respondent n, the sequence of choices, Y_n, is given by

$$Y_n = (\underbrace{Y_{n11}, Y_{n12}, \ldots, Y_{n1J}}_{\text{in choice set 1}}, \underbrace{Y_{n21}, Y_{n22}, \ldots, Y_{n2J}}_{\text{in choice set 2}}, \ldots, \underbrace{Y_{nS1}, Y_{nS2}, \ldots, Y_{nSJ}}_{\text{in choice set } S})',$$

where $Y_{nsj} = 1$, if respondent n chooses alternative j in choice set s; $Y_{nsj} = 0$, otherwise, and Y_n satisfies that $\sum_{j=1}^{J} Y_{nsj} = 1, \forall s$. The response in choice set s is $Y_{ns} = (Y_{ns1}, \ldots, Y_{nsJ})'$. Here, for simplicity in notation, the sizes of the choice situations, i.e. S and J, are assumed to be the same for all the respondents, they can be changed to accommodate more complex choice situations as needed. For the experiment, denote the design matrix as $X = (X_1', X_2', \ldots, X_N')'$ and the response as $Y = (Y_1', Y_2', \ldots, Y_N')'$.

Effects type coding is usually used for the attributes. For an attribute, with dummy coding, the effect of the base level is set to zero, and coefficients can be interpreted as the effects of the other levels compared to the base level; with effects type coding, the sum of the effects of the levels is set to zero, and coefficients can be interpreted as the effects of the levels around the grand mean. With effects type coding, e.g., an attribute A of two levels and an attribute B of three levels are coded as

$$\text{A} = \begin{cases} 1 & \text{for level 1} \\ -1 & \text{for level 2} \end{cases}, \quad \text{B} = \begin{cases} (1,0) & \text{for level 1} \\ (0,1) & \text{for level 2} \\ (-1,-1) & \text{for level 3} \end{cases}.$$

From (1.1), it can be shown that the probability will not change if a constant is added to all the utilities, so there is no intercept in the model. For an alternative with level 2 for attribute A and level 1 for attribute B, it is coded as $x = (x_A', x_B') = (1, 1, 0)'$.

For the discrete choice models, D-error is usually used in the literature as the

design criterion

$$\text{D-error} = \det[I(\theta|X)]^{-1/k},$$

where θ is the vector of parameters of length k, X is the design matrix, and $I(\theta|X)$ is the information matrix for θ that depends on the unknown parameter θ. The locally D-optimal design is the design which minimizes the D-error for a given value of θ.

D-error is motivated from the confidence ellipsoid for θ

$$\left\{\theta : (\theta - \hat{\theta})' I(\hat{\theta})(\theta - \hat{\theta}) \leq \text{constant}\right\},$$

where $\hat{\theta}$ is the ML estimator of θ. The ellipsoid has a volume proportional to $[\det I(\hat{\theta})]^{-1/2}$ (Silvey (1980)).

For the discrete choice models, Bayesian D-error used in the literature is

$$\text{D}_B\text{-error} = \int \det[I(\theta|X)]^{-1/k} \pi(\theta)\, \mathrm{d}\theta,$$

where $\pi(\theta)$ is the prior distribution for θ. The Bayesian D-optimal design is the design that minimizes D_B-error.

It should be noted that, in the general design literature, the criteria are defined similarly but in different forms. For example, D-optimality is defined as

$$D = \log\left(\det[I(\theta|X)^{-1}]\right),$$

in which the logarithmic transformation makes the D-optimality less sensitive to the extreme values of the determinant. Similarly, the Bayesian D-optimality is defined as

$$\mathrm{D}_B = \int \log(\det[I(\theta|X)])\pi(\theta)\,\mathrm{d}\theta.$$

The design that maximizes D_B also maximizes the expected Shannon information (Atkinson et al. (2007)).

1.3.1 Design for Multinomial Logit Model

With multinomial logit model, if the choices made by the same respondent are assumed to be independent, the probability of observing $y_{ns} = (y_{ns1}, \ldots, y_{nsJ})'$ in choice set s is given by,

$$P(Y_{ns} = y_{ns}|\beta) = \prod_{j=1}^{J} P(Y_{nsj} = y_{nsj}|\beta)^{y_{nsj}},$$

where

$$P(Y_{nsj} = y_{nsj}|\beta) = \frac{\exp(x'_{nsj}\beta)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta)}.$$

Since choices from different respondents are assumed to be independent by the multinomial logit model, the likelihood can be written as

$$L(\beta|y) = \prod_{n=1}^{N}\prod_{s=1}^{S} P(Y_{ns} = y_{ns}|\beta).$$

The information matrix of β for the multinomial logit model is

$$
\begin{aligned}
I(\beta|X) &= E\left(\frac{\partial \log L}{\partial \beta}\frac{\partial \log L}{\partial \beta'}\right) \\
&= \sum_{n=1}^{N}\sum_{s=1}^{S} X'_{ns}(P_{ns} - p_{ns}p'_{ns})X_{ns},
\end{aligned}
$$

where $p_{ns} = (P(Y_{ns1} = y_{ns1}|\beta), \ldots, P(Y_{nsJ} = y_{nsJ}|\beta))'$, and P_{ns} is a diagonal matrix with diagonal elements p_{ns}.

The following results are obtained assuming that the same design is used for all the respondents.

Assuming $\beta = 0_k$, Street and Burgess (2007) give the theoretical results of constructing locally D-optimal designs in a constrained design space. The details of the results are omitted here, and we will only show how they work in a simple example to find the locally optimal design. Suppose there are 2 alternatives with 2 binary attributes in each choice set. The levels of the binary attributes are given by 0 and 1. A choice set is given as a pair, e.g., (00,11). The constraint is that the pairs with v attributes different appear equally often, so the design space is restricted to the designs in Table 1.3. The three rows in Table 1.3 give three

Table 1.3: The possible design

	Pairs						S
Design 1			(00,11)	(01,10)			2
Design 2		(00,01)	(00,10)	(01,11)	(10,11)		4
Design 3	(00,11)	(01,10)	(00,01)	(00,10)	(01,11)	(10,11)	6

designs with 2, 4 and 6 choice sets respectively. The first design contains all pairs with 2 attributes different, the second design contains all pairs with 1 attribute

different, the third design contains the first two designs equally often. Street and Burgess (2007) prove that the D-optimal design contains fold-over pairs only, i.e., the pairs with all attributes different, so D-optimal design is the first design.

When $\beta = 0_k$, it can be shown that the information matrix is

$$NJ^{-1}\{X'X - \sum_{s=1}^{S} \frac{1}{J}(X_s' 1_J)(1_J' X_s)\}.$$

Goos et al. (2010) state that the above expression is the information matrix of the treatment effects β for a block design with blocks of J observations except for a multiplicative constant, so the D-optimal designs are equivalent for these two experiments.

Huber and Zwerina (1996) discuss the locally D-optimal designs with non-zero values for β. The locally optimal designs are more efficient if they are constructed with values of the parameters close to the true ones. Hence, if reasonable guesses for values of the parameter are available, it is more efficient to construct locally D-optimal designs with these values. Reasonable guesses can be obtained from, e.g., a pretest on a small sample.

Sándor and Wedel (2001) construct Bayesian D-optimal designs, which take the uncertainty about the assumed values of the parameters into account. The Bayesian D-optimal designs are thus expected to perform well over a wide range of values of the parameters. Two kind of designs with the following specifications are generated for comparisons: a locally D-optimal design with $\beta = \beta_0$ and a Bayesian D-optimal design with $\beta \sim N(\beta_0, \sigma_0^2 I)$, where σ_0^2 represents the uncertainty about

the assumed mean β_0. The two designs are compared over $\beta \sim N(\beta_0, \sigma^2 I)$, for different values of σ. The simulation results show that the locally D-optimal design is more efficient if σ is small, the Bayesian D-optimal design becomes more and more efficient compared to the locally D-optimal design as σ gets larger.

SAS macro %ChoiceEff can be used to generate locally D-optimal designs, the algorithm used is given in Zwerina et al.(1996; see updated [2005] version).

As an example, we will generate two locally D-optimal designs with β being zero and nonzero, and a Bayesian D-optimal design for the $3^4/2/9$ experiment. $3^4/2/9$ means there are 4 attributes with 3 levels and 9 choice sets of size 2 in the experiment. SAS macro %ChoiceEff is used to generate the two locally optimal designs. We write our own program to generate the Bayesian D-optimal design. With effects type coding, two parameters are needed for each attribute, so the parameter vector β is of length eight. The results are given in Table 1.4, where $b_0 = (1, 0, 1, 0, 1, 0, 1, 0)'$. We use pairs to denote the two levels of an attribute being compared in a choice set in the above designs. In Table 1.4, the three pairs (1, 2), (2, 3), and (1, 3) are compared exactly three times for each attribute in the locally D-optimal design with $\beta = 0_8$. (1, 1), (2, 2) or (3, 3) are not compared in the two locally D-optimal designs, but in the Bayesian D-optimal design, (1, 1), (2, 2), and (3, 3) are compared for several times.

1.3.2 Design for Cross-sectional Mixed Logit Model

With cross-sectional mixed logit model, the choices made by the same respondent in the S choice sets are assumed to be independent, so the probability of observing

Table 1.4: Optimal Designs for Multinomial Logit Model

Choice set	Alternative	Locally D-optimal $\beta = 0_8$ Attributes				Locally D-optimal $\beta = b_0$ Attributes				Bayesian D-optimal $\beta \sim N(b_0, I)$ Attributes			
		1	2	3	4	1	2	3	4	1	2	3	4
1	I	2	3	3	2	1	3	3	1	2	2	2	1
	II	1	2	2	3	2	1	2	2	1	2	2	2
2	I	3	1	1	3	2	2	2	2	1	3	1	1
	II	2	2	2	2	3	1	1	3	3	1	1	2
3	I	1	1	3	1	3	3	2	2	3	1	2	3
	II	2	2	1	3	2	1	3	3	3	2	1	2
4	I	1	2	1	1	3	2	1	2	2	1	1	2
	II	3	3	2	3	1	3	2	1	1	2	3	1
5	I	2	1	2	1	1	2	1	3	2	1	1	2
	II	1	3	1	2	3	1	2	1	3	1	1	1
6	I	3	3	2	1	2	3	3	2	1	2	1	3
	II	2	1	3	3	3	2	2	3	1	1	2	1
7	I	1	1	2	2	3	2	3	1	3	2	2	3
	II	3	2	3	1	2	3	2	3	3	3	2	3
8	I	3	1	1	2	1	3	1	2	1	2	2	2
	II	1	3	3	3	2	2	3	1	1	1	3	2
9	I	3	2	3	2	1	1	3	2	2	2	2	2
	II	2	3	1	1	2	3	1	1	1	1	1	3

$y_{ns} = (y_{ns1}, \ldots, y_{nsJ})'$ in choice set s is

$$\begin{aligned} P(Y_{ns} = y_{ns}|\theta) &= \prod_{j=1}^{J} P(Y_{nsj} = y_{nsj}|\theta)^{y_{nsj}} \\ &= \prod_{j=1}^{J} \left(\int P(Y_{nsj} = y_{nsj}|\beta_n) f(\beta_n|\theta) \, \mathrm{d}\beta_n \right)^{y_{nsj}}, \end{aligned}$$

where

$$P(Y_{nsj} = y_{nsj}|\beta_n) = \frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)}.$$

Since choices from different respondent are assumed to be independent by the mixed logit model, the likelihood function is

$$L(\theta|y) = \prod_{n=1}^{N}\prod_{s=1}^{S} P(Y_{ns} = y_{ns}|\theta).$$

The distribution of β_n is usually assumed to be $N_k(b, \Sigma)$, where $\Sigma = diag(\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2)$. β_n can be written as $\beta_n = b + U_n\sigma$, where $U_n = diag(u_n)$ with $u_n \sim N_k(0, I_k)$, and $\sigma = (\sigma_1, \ldots, \sigma_k)'$, so

$$\begin{aligned} P(Y_{nsj} = y_{nsj}|\theta) &= \int P(Y_{nsj} = y_{nsj}|\beta_n) f(\beta_n|b, \sigma)\, \mathrm{d}\beta_n \\ &= \int P(Y_{nsj} = y_{nsj}|u_n)\phi(u_n)\, \mathrm{d}u_n, \end{aligned}$$

where $P(Y_{nsj} = y_{nsj}|u_n) = \frac{\exp(x'_{nsj}(b+U_n\sigma))}{\sum_{i=1}^{J}\exp(x'_{ni}(b+U_n\sigma))}$. Then the information matrix is

$$I(b, \sigma|X) = \sum_{n=1}^{N}\sum_{s=1}^{S}\begin{bmatrix} M'_{ns}\Pi_{ns}^{-1}M_{ns} & M'_{ns}\Pi_{ns}^{-1}Q_{ns} \\ Q'_{ns}\Pi_{ns}^{-1}M_{ns} & Q'_{ns}\Pi_{ns}^{-1}Q_{ns} \end{bmatrix},$$

where

$$M_{ns} = \int [P_{ns}(u_n) - p_{ns}(u_n)p_{ns}(u_n)']X_{ns}\phi(u_n)\, \mathrm{d}u_n$$

and

$$Q_{ns} = \int [P_{ns}(u_n) - p_{ns}(u_n)p_{ns}(u_n)']X_{ns}U\phi(u_n)\,\mathrm{d}u_n$$

with $p_{ns}(u_n) = (P(Y_{ns1} = y_{ns1}|u_n), \ldots, P(Y_{nsJ} = y_{nsJ}|u_n))'$ and $P_{ns}(u_n)$ is a diagonal matrix with diagonal elements $p_{ns}(u_n)$, and $\Pi_{ns} = E(P_{ns}(u_n))$. The information matrix does not have a closed-form expression.

The following results are obtained assuming that the same design is used for all the respondents.

Sándor and Wedel (2002) derive the above expression of the information matrix. They compare the D-optimal design for the mixed logit model with that for the multinomial logit model, when the true model is the mixed logit model. From the simulation results, they conclude that the design generated with large heterogeneity parameter σ is more robust to the mis-specification of the mean parameters, and designs are more efficient with more alternatives in a choice set.

Yu et al. (2009) discuss the Bayesian designs for the mixed logit model. To calculate the D_B-error, the information matrix is integrated over the prior distribution of the hyper-parameters (b, σ), which makes the computation more difficult. They propose to use a small prior draw of only 20 points so that the optimal design can still be found. In the simulation study, they compare eight designs including a nearly orthogonal design, a locally D-optimal design for the multinomial logit model, two Bayesian D-optimal designs for the multinomial logit model, a locally D-optimal design for the mixed logit model, and three Semi-Bayesian D-optimal design for the mixed logit model. A Semi-Bayesian design is generated with a

prior distribution for b and a prior value for σ, since generating full Bayesian design with prior distributions for both b and σ is computationally impractical. They conclude that the semi-Bayesian design generated with large prior value for the heterogeneity parameter σ is more robust to the misspecification of the mean parameters, the design is not very sensitive to the misspecification of the prior value for the heterogeneity parameter σ. The nearly orthogonal design and the locally D-optimal design for multinomial logit model perform poorly compared to the other designs when the true model is the mixed logit model.

As an example, we will generate a locally D-optimal design and a Bayesian D-optimal design for the $3^4/2/18$ case. $3^4/2/18$ means there are 4 attributes with 3 levels and 18 choice sets of size 2 in the experiment. We write our own programs to generate both designs. The results are given in Table 1.5, where $b_0 = (1, 0, 1, 0, 1, 0, 1, 0)'$. In Table 1.5, for each attribute, the number of times that (1, 1), (2, 2), and (3, 3) are compared is around 6, which is one third of the number of choice sets.

1.3.3 Design for Panel Mixed Logit Model

If the respondents are presented with more than one choice set, the data has a panel structure. The panel mixed logit model can be used to model the correlation in panel data. The simplest specification is to assume β_n be constant in different choice sets. Given β_n, the conditional probability of observing y_n is

Table 1.5: Optimal Designs for Mixed Logit Model Assuming Independence

Choice set	Alternative	Locally D-optimal $b_0, \sigma = 1_8$ Attributes				Bayesian D-optimal $b \sim N(b_0, I), \sigma = 1_8$ Attributes			
		1	2	3	4	1	2	3	4
1	I	3	1	1	1	2	1	2	3
	II	2	3	3	3	2	1	1	3
2	I	2	1	2	1	1	2	2	3
	II	3	2	2	2	3	3	1	1
3	I	3	1	1	1	2	2	3	2
	II	1	2	2	2	1	2	2	2
4	I	2	2	2	3	3	3	3	2
	II	2	3	1	1	3	3	3	3
5	I	2	3	2	2	2	3	1	3
	II	1	3	1	2	1	2	1	3
6	I	2	3	2	1	1	1	2	2
	II	2	3	3	2	1	2	3	3
7	I	3	3	1	1	2	3	3	1
	II	1	2	1	1	3	3	3	2
8	I	1	1	2	3	2	2	1	3
	II	1	3	2	3	2	1	1	1
9	I	3	1	3	2	2	3	1	1
	II	1	3	1	1	2	3	3	2
10	I	2	3	2	2	2	3	1	2
	II	1	2	1	3	3	2	3	2
11	I	1	1	1	3	3	1	1	1
	II	3	1	1	3	2	3	2	3
12	I	1	3	2	2	1	1	3	2
	II	2	3	2	3	2	2	1	3
13	I	1	1	1	3	3	2	3	1
	II	3	2	2	1	1	1	3	1
14	I	1	1	3	1	1	3	3	3
	II	2	2	1	2	1	1	3	3
15	I	2	2	1	1	1	3	1	2
	II	2	2	3	1	3	1	2	1
16	I	2	1	3	2	2	2	2	2
	II	3	3	2	3	1	3	2	1
17	I	2	2	3	1	2	1	1	3
	II	1	1	2	2	2	2	2	1
18	I	1	2	2	1	3	3	2	2
	II	1	2	2	3	1	2	3	1

$$
\begin{aligned}
P(Y_n = y_n|\beta_n) &= \prod_{s=1}^{S}\prod_{j=1}^{J} P(Y_{nsj} = y_{nsj}|\beta_n)^{y_{nsj}} \\
&= \prod_{s=1}^{S}\prod_{j=1}^{J} \left(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)}\right)^{y_{nsj}}.
\end{aligned}
$$

Given β_n, the choices made by respondent n in the S choice sets are independent. But the choices in different choice sets are not independent in general. The unconditional choice probability of observing y_n is

$$
P(Y_n = y_n|\theta) = \int P(Y_n = y_n|\beta_n) f(\beta_n|\theta)\,\mathrm{d}\beta_n.
$$

Since choices from different respondent are assumed to be independent by the mixed logit model, the likelihood function is

$$
L = \prod_{n=1}^{N} P(Y_n = y_n|\theta).
$$

Bliemer and Rose (2010) discuss locally D-optimal designs for the above model. They find that the information matrix does not have a closed-form expression, and it is more complicated than the information matrix for the mixed logit model assuming independence, for it also involves expectation of functions of y_n, e.g., $E_{y_n}(P(y_n|\theta))$. Hence $E_{y_n}(P(y_n|\theta))$ does not have a closed-form expression. We find the expression they use for the information matrix can be simplified, and the information matrix will be given in the next section.

Yu et al. (2011) propose an individually adapted sequential Bayesian (IASB)

approach for this model. Instead of the ordinary Fisher information matrix, they use the generalized Fisher information matrix for the multinomial logit model. Yu et al. (2008) show that the generalized Fisher information gives a more reliable approximation of the posterior covariance matrix in small data sets. The generalized Fisher information is defined as

$$I_{GFIM}(\beta_n|X) = -E\left[\frac{\partial^2 \log q(\beta_n|Y,X)}{\partial\beta_n\partial\beta_n'}\right],$$

where $q(\beta_n|Y,X)$ is the posterior distribution of β_n. Suppose the prior distribution for β_n, $\pi(\beta_n)$, is a multivariate normal distribution with covariance matrix Σ_{β_n}, then the generalized Fisher information is given by

$$I_{GFIM}(\beta_n, X) = \sum_{s=1}^{S} X_{ns}'(P_{ns} - p_{ns}p_{ns}')X_{ns} + \Sigma_{\beta_n},$$

where $p_{ns} = (P(Y_{ns1} = y_{ns1}|\beta_n), \ldots, P(Y_{nsJ} = y_{nsJ}|\beta_n))'$ and P_{ns} is a diagonal matrix with elements p_{ns}. The Bayesian D_B-error is

$$\text{D}_B\text{-error} = \int det[I_{GFIM}(\beta_n|X)]^{-1/k}\pi(\beta_n)\,\text{d}\beta_n,$$

where $\pi(\beta_n)$ is the prior for β_n. The design is generated in two stages, an initial static stage and an adaptive sequential stage. In the initial static stage, S_1 choice set $X_n^{S_1}$ are generated for respondent n. The posterior for β_n, $q(\beta_n|y_n^{S_1}, X_n^{S_1})$, can be computed after the responses $y_n^{S_1}$ on $X_n^{S_1}$ are observed. $q(\beta_n|y_n^{S_1}, X_n^{S_1})$ is used as the prior in D_B to find the first choice set added in the adaptive sequential stage.

In the adaptive sequential stage, one choice set is added at a time and the posterior is updated after each choice made on the newly added choice set. The updated posterior is then used as the prior in calculating D_B to find a new choice set. The adaptive sequential stage stops after a given number of choice sets are added. Yu et al.(2011) compare four designs, an IASB design, a design for cross-sectional mixed logit model, a design for multinomial logit model, and a nearly orthogonal design, through simulation. The simulation results show that the IASB design can capture the individual preference well: it provides more accurate estimation for individual parameter β_n, more accurate estimation for hyper-parameter (b, σ), and better predictions of individual choices.

We will use simulation to find a locally D-optimal design for the $3^4/2/18$ case. Since the information matrix does not have a closed-form expression, samples of u_n and y_n are needed to approximate the expectations. Also, a searching algorithm is needed to find the locally optimal design. Samples of u_n and y_n are generated with Modified Latin Hypercube Sampling method (see Hess et al.(2006)).

Coordinate-exchange algorithm (Meyer and Nachtsheim(1995)) is used to search for the optimal design. The algorithm starts with an initial set of choice sets of the desired size. The exchanges are carried out in the following way: for an attribute of an alternative in a choice set, exchange the current level of the attribute with the possible levels it can take, if the exchange results in an improvement in the design criterion, the exchange is kept. The exchanges are done for every attribute of every alternative in every choice set: from the first attribute of the first alternative in the first choice set to the last attribute of the last alternative in the last

choice set, and from the first attribute of the first alternative in the first choice set again repeatedly, until no exchange can be made. This is called a run of the algorithm. Many runs with different initial sets of choice sets are used to avoid local optima.

We will use 100 runs for the coordinate-exchange algorithm, 200 as the sample size for u_n, and 500 as the sample size for y_n. Assuming $b = (1, 0, 1, 0, 1, 0, 1, 0)', \sigma = 1_8$, the design obtained from the computer search is given in Table 1.6. Compare to the designs in Table 1.5, (1, 1), (2, 2), and (3, 3) are compared less often in the design in Table 1.6.

1.3.4 Designs for Other Models

Goos, Vermeulen and Vandebroek (2010) discuss designs for a no-choice nested logit model, which consists of two nests: one containing all real alternatives and the other containing the no-choice option. The row in the design matrix for the no-choice option is set to zero, so the representative utility of the no-choice option is also zero. Results regarding locally optimal designs under the indifference condition are given for any linear model. Locally optimal designs with and without the indifference assumption and Bayesian optimal designs are compared. The locally optimal design without the indifferent assumption and Bayesian optimal designs are generated by computer search. Given the same parameter values and the same design criterion, adding a no-choice option to every choice set of a optimal design for the multinomial logit model result in a less efficient designsfor no-choice nested logit models.

Table 1.6: Optimal Design for Mixed Logit Model Assuming Correlation

Choice set	Alternative	Locally D-optimal Attributes			
		1	2	3	4
1	I	1	3	3	1
	II	2	1	2	2
2	I	1	2	1	1
	II	3	3	2	3
3	I	2	3	1	1
	II	3	1	3	1
4	I	2	2	2	1
	II	1	3	1	2
5	I	1	1	2	1
	II	2	1	3	3
6	I	1	2	3	2
	II	2	3	1	3
7	I	3	3	1	2
	II	2	1	3	3
8	I	2	1	3	1
	II	2	2	2	2
9	I	2	2	2	1
	II	1	3	1	2
10	I	3	3	1	1
	II	1	2	3	3
11	I	1	3	3	1
	II	3	2	1	2
12	I	1	2	2	1
	II	2	2	3	3
13	I	2	2	2	1
	II	3	3	3	3
14	I	2	3	2	2
	II	1	1	3	3
15	I	2	3	3	1
	II	2	3	2	2
16	I	2	2	2	1
	II	1	3	1	2
17	I	3	3	1	1
	II	1	2	3	3
18	I	1	3	3	1
	II	3	2	1	2

1.4 Review of Generalized Linear Mixed Models

Multinomial logit model (MNL) can be considered as a generalization of Logistics regression for a response with more than two categories, which is a special case of generalized linear model (GLM). Generalized linear mixed model (GLMM) is obtained by adding random effects to GLM. Thus, mixed logit model, which is developed by adding random effects to MNL, is a special case of GLMM. Mixed logit model is mostly used and studied in marketing, leading to some gap between analysis of mixed logit model and analysis of GLMM in the statistical literature.

The likelihood function of GLMM does not have a closed form expression. In order to evaluate the likelihood function, four types of approaches are taken to solve the problem. First, "brute force" methods approximate the likelihood function with numerical integration, e.g., quadrature methods and importance sampling. Second, Generalized Estimating Equations (GEE) construct estimating equations in a way that solutions of the equations are unbiased estimates of the model parameters under certain conditions. With GEE, only marginal distribution are specified not the full likelihood function, which greatly reduces amount of the calculation. Third, penalized quasi-likelihood (PQL) is derived based on a Laplace's approximation to the likelihood function. PQL does not involve a numerical integration, which is the part that takes the most effort in the estimation process, but the estimates are not asymptotically unbiased because of the simplifications made in its derivation. Fourth, Monte Carlo EM (MCEM) combines EM algorithm with Metropolis algorithm and Monte Carlo Newton-Raphson (MCNR) combines Newton-Raphson (NR) with Metropolis algorithm. Metropolis

algorithm is used to take samples from the posterior distribution of random effects given responses, which EM and NR requires in the calculation instead of the likelihood function. Details of the first three approaches will be listed in Chapter 2, which will also be used in Chapter 4 for the design of panel mixed logit model. The Metropolis algorithm in the fourth approach will be given in Chapter 3, but MCEM and MCNR will not be covered at full details since our goal is to find techniques applicable to designs.

1.5 Summary and Discussions

In choice experiments, respondents state about what they prefer when are presented with several alternatives, which contain valuable information about how respondents make their trade offs among the alternatives. Thus, designs that outline the questions to ask is very important. The amount of information can be obtained is determined by the design. Also, such experiments are under constraints of people's cognitive abilities. i.e., a person cannot compare too many alternative in one question and finish too many questions. Optimal designs for choice experiments have been studied by researchers from marketing, transportation and statistics. Previous work focuses on developing optimal designs for models that assume independence between different choice sets, which ignores the correlation in choices made by the same person.

We consider optimal designs for the panel mixed logit model, where choices made by the same respondent are assumed to be correlated. The design criteria

based on the information matrix do not have a closed form expression. Bliemer and Rose (2010) use numerical integration of the information matrix, which is generally slow. We note that the panel mixed logit model is a special case of the generalized linear mixed models. In the analysis of generalized linear mixed models (GLMM), there are approximate methods for analysis that can reduce the amount of calculation from using numerical integration of the likelihood function. We will apply these methods to approximate information matrix for the panel mixed logit model. In Chapter 2, we give a review of approximate analysis methods for GLMM, which leads to approximations of the information matrix for GLMM. We show one of the approximations can give a closed form approximation to the information matrix of the Poisson mixed effects model. In Chapter 3, we consider approximations to the information matrix of the panel mixed logit model, including different sampling methods to get the numerical integration and Laplace's method. In Chapter 4, we apply methods in Chapter 2 to the panel mixed logit model and use computer search to get optimal designs with criteria based on the approximations for these mothods.

1.6 References

Atkinson, A. C., Donev, A. N., and Tobias, R. D. (2007). *Optimum experimental designs, with SAS* , New York: Oxford University Press.

Bliemer, M. C., and Rose, J. M. (2010). "Construction of experimental designs for mixed logit models allowing for correlation across choice observations", *Transportation Research Part B: Methodological*, 44(6), 720−734.

Chaloner, K., and Verdinelli, I. (1995). "Bayesian experimental design: A review", *Statistical Science*, 273−304.

Danthurebandara, V. M., Yu, J., and Vandebroek, M. (2011). "Sequential choice designs to estimate the heterogeneity distribution of willingness-to-pay". *Quantitative Marketing and Economics*, 9(4), 429−448.

Goos, P., Vermeulen, B., and Vandebroek, M. (2010). "D-optimal conjoint choice designs with no-choice options for a nested logit model", *Journal of Statistical Planning and Inference*, 140(4), 851−861.

Hess, S., Train, K. E., and Polak, J. W. (2006). "On the use of a Modified Latin Hypercube Sampling (MLHS) method in the estimation of a Mixed Logit model for vehicle choice", *Transportation Research Part B: Methodological*, 40(2), 147−163.

Huber, J., and Zwerina, K. (1996). "The importance of utility balance in efficient choice designs", *Journal of Marketing research*, 307−317.

Khuri, A. I., Mukherjee, B., Sinha, B. K., and Ghosh, M. (2006). "Design issues for generalized linear models: A review", *Statistical Science*, 376−399.

Kuhfeld, W. F. (2006). "Construction of Efficient Designs for Discrete Choice Experiments", *The Handbook of Marketing Research: Uses, Misuses, and Future Advances*, 312.

Liu, Q., and Arora, N. (2011). "Efficient choice designs for a consider-then-choose model", *Marketing Science*, 30(2), 321−338.

Manski, C. (1977). "The Structure of Random Utility Models", *Theory And Decision* **8**, 229−254.

Meyer, R. K., and Nachtsheim, C. J. (1995). "The coordinate-exchange algorithm for constructing exact optimal experimental designs", *Technometrics*, 37(1), 60−69.

Peter E. Rossi, Greg M. Allenby, Robert McCulloch (2006). *Bayesian Statistics and Marketing*, John Wiley and Sons, Ltd.

Sándor, Z., and Wedel, M. (2001). "Designing conjoint choice experiments using managers' prior beliefs", *Journal of Marketing Research*, 430−444.

Sándor, Z., and Wedel, M. (2002). "Profile construction in experimental choice designs for mixed logit models", *Marketing Science*, 21(4), 455−475.

Silvey, Samuel David (1980), *OptimalDesign: An Introductionto the Theoryfor ParameterEstimation*, London: Chapmanand Hall.

Street, D. J., and Burgess, L. (2007). *The construction of optimal stated choice experiments: theory and methods* , Wiley-Interscience.

Thurstone, L. L. (1929). "A Law of Comparative Judgement", *Psychological Review* **34**, 273−286.

Vermeulen, B., Goos, P., and Vandebroek, M. (2008). "Models and optimal designs for conjoint choice experiments including a no-choice option", *International Journal of Research in Marketing*, 25(2), 94−103.

Yu, J., Goos, P., and Vandebroek, M. (2009). "Efficient conjoint choice designs in the presence of respondent heterogeneity", *Marketing Science*, 28(1), 122−135.

Yu, J., Goos, P., and Vandebroek, M. (2011). "Individually adapted sequential Bayesian conjoint-choice designs in the presence of consumer heterogeneity", *International Journal of Research in Marketing*, 28(4), 378−388.

Zwerina, Klaus, Joel Huber, and Warren F. Kuhfeld (1996), "A General Method for Constructing Efficient Choice Designs", working paper, Fuqua School of Business, Duke University. (Updated version [2005] available at `http://support.sas.com/techsup/tnote/tnote_stat.html.`)

Chapter 2

Methods for Analysis of Generalized Linear Mixed Models

2.1 Introduction

In optimal designs, a design is often evaluated by a measure based on the information matrix (Atkins et al.(2007)). For generalized linear mixed models, the information matrix does not have a closed-form expression. Hence, design measures calculated from the information matrix also do not have closed-form expressions. For these models, computer searches are often used to find designs that optimize a given design measure. However, the complex form of the information matrix often makes the search computationally difficult. For this reason, alternatives to the information matrix have been used in Moerbeek and Mass(2005), Niaparast(2009), Niaparast and Schwabe(2013), Ogungbenro and Aarons(2011), Tekle et al.(2008) and Waite et al.(2012). Except for Ogungbenro and Aarons(2011), they consider models with known variance of the random effects. We will relax this assumption and develop alternatives to the information matrix.

Let $Y = (Y_1, \ldots, Y_N)'$ be the response vector. Given the q-vector of random effects

u, the elements of Y are independent and the conditional distribution of Y_i is given by the exponential family,

$$P_\beta(Y_i = y_i|u) = \exp\Big[\big(y_i\gamma_i - b(\gamma_i)\big)/a(\phi) + c(y_i,\phi)\Big],$$

where $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ are known functions, ϕ is a dispersion parameter which may or may not be known and γ_i is an unknown parameter. A link function $g(\mu_i) = x_i'\beta + z_i'u$ relates the conditional mean $E(y_i|u) = \mu_i$ to the linear predictor of the fixed and random effects, where x_i' the ith row of the model matrix X for the fixed effects, z_i' the ith row of the model matrix Z for the random effects and β is the p-vector of parameters for the fixed effects. The conditional variance can be written as $V(y_i|u) = a(\phi)b''(\gamma_i)$, where $b''(\gamma_i)$ is usually called the variance function. The distribution of the random effects is given by $u \sim f_\psi(u)$, where ψ is the parameter vector. The likelihood for the unknown parameter vector $\theta = (\beta', \psi')'$ is

$$L = P_\theta(Y) = \int P_\beta(Y = y|u) f_\psi(u)\,\mathrm{d}u,$$

where $P_\beta(Y = y|u) = \prod_i P_\beta(Y_i = y_i|u)$. The score for the fixed effects is

$$\begin{aligned}
\frac{\partial \log L}{\partial \beta} &= \frac{\frac{\partial}{\partial \beta}\int P(Y=y|u)f_\psi(u)\,\mathrm{d}u}{P_\theta(Y=y)} \\
&= \frac{\int [\frac{\partial}{\partial \beta}P_\beta(Y=y|u)]f_\psi(u)\,\mathrm{d}u}{P_\theta(Y=y)} \\
&= \int \frac{\partial \log P(Y=y|u)}{\partial \beta} f_\theta(u|y)\,\mathrm{d}u \\
&= \int X'W(y-\mu)f_\theta(u|y)\,\mathrm{d}u \\
&= X'E_U(W|y)y - X'E_U(W\mu|y), \qquad (2.1)
\end{aligned}$$

where $\frac{\partial \log P(Y=y|u)}{\partial \beta}$ is the score of a generalized linear model and $W = diag([a(\phi)\nu(\mu_i)g'(\mu_i)]^{-1})$.

The score for ψ is

$$\begin{aligned}
\frac{\partial \log L}{\partial \psi} &= \int \frac{\partial \log f_\psi(u)}{\partial \psi} f_\theta(u|y)\,\mathrm{d}u \\
&= E_U(\frac{\partial \log f_\psi(u)}{\partial \psi}|y). \qquad (2.2)
\end{aligned}$$

The information matrix is

$$\begin{aligned}
I &= \begin{bmatrix} I_{\beta\beta} & I_{\beta\psi} \\ I_{\psi\beta} & I_{\psi\psi} \end{bmatrix} \\
&= \begin{bmatrix} E_Y(\frac{\partial \log L}{\partial \beta}\frac{\partial \log L}{\partial \beta'}) & E_Y(\frac{\partial \log L}{\partial \beta}\frac{\partial \log L}{\partial \psi'}) \\ E_Y(\frac{\partial \log L}{\partial \psi}\frac{\partial \log L}{\partial \beta'}) & E_Y(\frac{\partial \log L}{\partial \psi}\frac{\partial \log L}{\partial \psi'}) \end{bmatrix}.
\end{aligned}$$

The terms in the information matrix do not have closed form expressions and can

only be evaluated numerically. For example,

$$\begin{aligned} I_{\beta\beta} &= E_Y\left(\frac{\partial \log L}{\partial \beta}\frac{\partial \log L}{\partial \beta'}\right) \\ &= E_Y\Big(\left[X'E_U(W|y)y - X'E_U(W\mu|y)\right]\left[X'E_U(W|y)y - X'E_U(W\mu|y)\right]'\Big). \end{aligned}$$

The above expression contains expectations with respect to U and Y jointly. Since Y is categorical, the expectation with respect to Y is the sum of the product of the score function and its transpose over all possible value of Y. As shown in (2.1) and (2.2), the score involves evaluating the posterior mean of a function, which usually does not have a closed-form expression. The other terms in the information matrix can be shown to be in similar forms.

Analysis of this model is discussed in McCulloch (1997), where the goal is to obtain estimates of the parameters and an observed information matrix can be used given observed values of Y. However, observed value of Y are not available before the experiment.

2.2 Different Methods for Estimation

Since the likelihood function does not have a closed form expression, several numerical optimization algorithms have been proposed to get the maximum likelihood estimator.

2.2.1 Penalized Quasi-Likelihood

Schall (1991), Liu (1993), Brewlow & Clayton (1993), Lin & Breslow (1996) apply Laplace's method to approximate the likelihood function. For an integral of the form $\int e^{-\kappa(u)}\,\mathrm{d}u$, the Laplace approximation to the integral is

$$\int e^{-\kappa(u)}\,\mathrm{d}u \approx (2\pi)^{q/2}|\kappa''(\tilde{u})|^{-1/2}e^{-\kappa(\tilde{u})},$$

where $\tilde{u}$ is the mode of $\kappa(u)$. Breslow and Clayton (1993) propose penalized quasi-likelihood(PQL), where no assumption is made about $P_\beta(Y_i = y_i|u)$. The conditional mean and variance is given by $E(y_i|u) = \mu_i$ and $Var(y_i|u) = a(\phi)\nu(\mu_i)$. Also, u is assumed to follow a multivariate normal distribution with mean 0 and variance-covariance matrix Σ of which the unknown parameters are contained in vector σ.

The quasi-likelihood function for unknown parameter vector $\theta = (\beta', \sigma')'$ is given by

$$e^{ql(\beta,\sigma)} \propto |\Sigma|^{-1/2} \int \exp\Big[-\frac{1}{2}\sum_{i=1}^{n} d_i(y_i, \mu_i) - \frac{1}{2}u'\Sigma^{-1}u\Big]\mathrm{d}u,$$

where

$$d_i(y, \mu) = -2\int_y^\mu \frac{y_i - t}{a(\phi)\nu(t)}\mathrm{d}t$$

is the deviance. If the conditional distribution of y_i belongs to the exponential family with variance function $\nu(\cdot)$, $ql(\beta, \sigma)$ is the same as the log-likelihood function.

Let $\kappa(u) = \frac{1}{2}\sum_{i=1}^{N} d_i(y_i;\mu_i) + \frac{1}{2}u'\Sigma^{-1}u$. Applying Laplace's method to the integrated quasi-likelihood, the approximation to the log quasi-likelihood is

$$ql(\beta,\sigma) \approx -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\log|\kappa''(\hat{u})| - \kappa(\hat{u}) + c,$$

where c is a constant, $\kappa''(\cdot)$ is the $q \times q$ matrix of second-order derivative of $\kappa(\cdot)$ with respect to u, $\hat{u}$ minimizes $\kappa(u)$ and therefore is the solution to

$$\kappa'(u) = \Sigma^{-1}u - \sum_{i=1}^{N} \frac{y_i - \mu_i}{a(\phi)\nu(\mu_i)g'(\mu_i)} z_i = 0,$$

where $\kappa'(\cdot)$ is the first-order derivative of $\kappa(\cdot)$ with respect to u. It can be shown that

$$\kappa''(u) = \Sigma^{-1} + \sum_{i=1}^{N} \frac{z_i z_i'}{a(\phi)\nu(\mu_i)(g'(\mu_i))^2} + R \approx \Sigma^{-1} + Z'WZ,$$

where W^* is the $N \times N$ matrix with diagonal terms $w_i = \{a(\phi)\nu(\mu_i)[g'(\mu_i)]^2\}^{-1}$. With canonical link function, the remainder term R is 0. The approximation can be written as

$$ql(\beta,\sigma) \approx -\frac{1}{2}\log|I + Z'WZ\Sigma| - \frac{1}{2}\sum_{i=1}^{N} d_i(y_i,\hat{\mu}) - \frac{1}{2}\hat{u}'\Sigma^{-1}\hat{u}.$$

Assuming that the ω_i's vary slowly(or not at all) as a function of the mean, $ql(\beta,\sigma)$ can be further simplified to

$$-\frac{1}{2}\sum_{i=1}^{N} d_i(y_i,\tilde{\mu}) - \frac{1}{2}\hat{u}'\Sigma^{-1}\hat{u}. \tag{2.3}$$

The estimate of β can then be obtained by maximizing $\kappa(\hat{u})$. Thus, $(\hat{\beta}', \hat{u}')'$ can be obtained by jointly maximizing the Penalized Quasi-likelihood(PQL)

$$-\frac{1}{2}\sum_{i=1}^{N} d_i(y_i;\mu_i) - \frac{1}{2}u'\Sigma^{-1}u,$$

which will be the solution to the following score equations from PQL

$$\begin{aligned}
\sum_{i=1}^{N} \frac{(y_i-\mu_i)x_i}{a(\phi)\nu(\mu_i)g'(\mu_i)} &= 0, \\
\sum_{i=1}^{N} \frac{(y_i-\mu_i)z_i}{a(\phi)\nu(\mu_i)g'(\mu_i)} &= \Sigma^{-1}u.
\end{aligned}$$

Fisher scoring algorithm can be used to get the solution of the above equations. Let $y_i^* = \eta_i + (y_i - \mu_i)g'(\mu_i)$ be the ith element of the working vector y^* at the current estimate of $(\beta', u')'$. The new estimate of $(\beta', u')'$ is the solution to the following equations

$$\begin{bmatrix} X'WX & X'WZ \\ Z'WX & \Sigma^{-1} + Z'WZ \end{bmatrix} \begin{pmatrix} \beta \\ u \end{pmatrix} = \begin{pmatrix} X'Wy^* \\ Z'Wy^* \end{pmatrix},$$

where W is evaluated at the current estimate of $(\beta', u')'$. The variance-covariance matrix for the final estimate $\hat{\beta}$ is given by $(X'V^{-1}X)^{-1}$, where $V = W^{-1} + Z\Sigma Z'$.

The update formula for σ given $(\hat{\beta}(\sigma)', \hat{u}(\sigma)')'$ is derived from REML version of the approximate likelihood function. The (j,k)th element of information matrix

J for σ is given by

$$J_{jk} = -\frac{1}{2} tr(P\frac{\partial V}{\partial \sigma_j} P \frac{\partial V}{\partial \sigma_k}),$$

where $P = V^{-1} - V^{-1}X(X'V^{-1}X)^{-1}X'V^{-1}$.

Assuming $\hat{\beta}$ and $\hat{\sigma}$ are independent, the variance-covariance matrix of $(\hat{\beta}', \hat{\sigma}')'$ from PQL is

$$\begin{pmatrix} (X'V^{-1}X)^{-1} & 0 \\ 0 & J^{-1} \end{pmatrix}.$$

Ogungbenro and Aarons (2011) applied the above expression to pharmacokinetic experiments with repeated measurements, which are usually analyzed by a generalized mixed effects model.

2.2.2 Marginal Quasi-Likelihood

To get marginal quasi-likelihood, Breslow and Clayton(1993) represent the response y_i given u in PQL as

$$y_i = \mu_i + \epsilon_i,$$

where $E(\epsilon_i) = 0$ and $Var(\epsilon_i) = a(\phi)\nu(\mu_i)$. Treating the right hand side of the above equation as a function of u, we apply a first order taylor approximation around 0 as

$$\begin{aligned} y_i &= h(\eta_i) + \epsilon_i \\ &\approx h(x_i'\beta) + h'(x_i'\beta)z_i'u + \epsilon_i, \end{aligned}$$

where $h(\cdot) = g^{-1}(\cdot)$. From the above approximation, the marginal mean of Y_i is approximated by $\mu_i^0 = h(x_i'\beta)$ and the marginal variance of Y_i is approximately

$$Var(Y_i) \approx V_0 + \Delta_0^{-1} Z \Sigma Z' \Delta_0^{-1},$$

where $V_0 = diag(a(\phi)\nu(\mu_i^0))$ and $\Delta_0 = diag(g'(\mu_i^0))$

If σ is known, the estimate of β is the solution to the following estimating equations

$$(\frac{\partial \mu^0}{\partial \beta})' Var^{-1}(y)(y - \mu^0) = 0,$$

where $\mu^0 = (\mu_1^0, \ldots, \mu_N^0)'$. It can be shown that the above estimating equations are given by

$$X'(W_0^{-1} + ZDZ')^{-1}\Delta_0(y - \mu^0) = 0,$$

where W_0 is W evaluated at $u = 0$.

The covariance matrix of $(\hat{\theta}, \hat{\sigma})$ has the same form as that of PQL except that u is evaluated at 0 and $\hat{\beta}$ and $\hat{\sigma}$ from MQL are orthogonal while they not in PQL.

Moerbeek and Mass (2005) applied the covariance matrix to find optimal designs for multilevel logistics models with two binary predictors.

2.2.3 Method of Simulated Moments

The method of simulated moments (MSM) are used in the econometrics literature by McFadden (1989) and Lee (1992). Jiang (1995) propose a method based on

simulated moments for generalized linear mixed models, which provide computational advantage and consistent estimator. Assuming q-vector of random effects can be partitioned as $u = (u_1', u_2', \ldots, u_r')'$ with $u_v \sim N(0, \sigma_v^2 I_{q_v})$ for $1 \le v \le r$, we write u as $u = D\gamma$ where $D = diag(\sigma_1{}^2 I_{q_1}, \ldots, \sigma_r{}^2 I_{q_r})$ and $\gamma \sim N(0, I_q)$. Accordingly, the columns of the design matrix for the random effects is partitioned as $Z = (Z^1, \ldots, Z^r)$, where the ith row is $z_i' = (z_{i1}^1, \ldots, z_{iq_1}^1, \ldots, z_{i1}^r, \ldots, z_{iq_r}^r)$. Then, the linear predictor can be written as $\eta_i = x_i'\beta + z_i' D\gamma$. Using canonical link functions, the likelihood up to a multiplicative constant is

$$\begin{aligned} L &\propto \int \prod_{i=1}^{N} \exp\Big[\frac{y_i\eta_i - b(\eta_i)}{a(\phi)} + c(y_i,\phi) - \gamma'\gamma/2\Big]\,\mathrm{d}\gamma \\ &\propto \int \prod_{i=1}^{N} \exp\Big[\frac{y_i(x_i'\beta + z_i'D\gamma) - b(x_i'\beta + z_i'D\gamma)}{a(\phi)} + c(y_i,\phi) - \gamma'\gamma/2\Big]\,\mathrm{d}\gamma \\ &= \int \exp\Big[\sum_{i=1}^{N} y_i x_i'\beta/a(\phi) + \sum_{i=1}^{N} y_i z_i' D\gamma/a(\phi) - \sum_{i=1}^{N} b(x_i'\beta + z_i'D\gamma)/a(\phi) + \sum_{i=1}^{N} c(y_i,\phi) \\ &\qquad -\gamma'\gamma/2\Big]\,\mathrm{d}\gamma \end{aligned}$$

If ϕ is known, a set of sufficient statistics for $\theta = (\beta', \sigma')'$ is $(\sum_{i=1}^N y_i x_i', \sum_{i=1}^N y_i z_i')'$. The method of moments estimating equations can be formulated as

$$\begin{aligned} \sum_{i=1}^{N} x_{ij} y_i &= \sum_{i=1}^{N} x_{ij} E(y_i), \quad 1 \le j \le p, \\ \sum_{l=1}^{q_v} \big(\sum_{i=1}^{N} z_{il}^v y_i\big)^2 &= \sum_{l=1}^{q_v} E\big(\sum_{i=1}^{N} z_{il}^v y_i\big)^2, \quad 1 \le v \le r. \end{aligned}$$

If ϕ is unknown, the second set of equations need to be modified so it is free of

ϕ, since the second moments of y_i may involve ϕ. The modified second set of equations is

$$\begin{aligned} & \sum_{l=1}^{q_v}(\sum_{i=1}^{N} z_{il}^v y_i)^2 - \sum_{l=1}^{q_v}\sum_{i=1}^{N}(z_{il}^v y_i)^2 \\ = & \sum_{l=1}^{q_v}\sum_{s\neq t} z_{sl}^v z_{tl}^v y_s y_t = \sum_{l=1}^{q_v}\sum_{s\neq t} z_{sl}^v z_{tl}^v E(y_s y_t), \quad 1 \leq v \leq r. \end{aligned}$$

If $Z^v (1 \leq v \leq r)$ is a standard design matrix in the sense that Z^v consists of only 0's and 1's and there is only one 1 in each row and at least one 1 in each column, the following expressions for MM equations can be obtained. Any row of Z^v satisfies$|z_i^v|^2 = 1$ and any two different rows of Z^v satisfies $(z_s^v)' z_t^v = 0$ or 1 for $s \neq t$. Let $N_v = \{(s,t) : 1 \leq s \neq t \leq n, (z_s^v)' z_t^v = 1\} = \{(s,t) : 1 \leq s \leq t \leq n, z_s^v = z_t^v\}$. Then, the right hand side of the second set of equations can be written as

$$\begin{aligned} \sum_{l=1}^{q_v} E(\sum_i z_{il}^v y_i)^2 &= E\sum_i (\sum_{l=1}^{q_v} (z_{il}^v)^2) y_i^2 + \sum_{s\neq t} E(\sum_{l=1}^{q_v} z_{sl}^v z_{tl}^v) y_s y_t \\ &= \sum_i E(y_i^2) + \sum_{(s,t)\in N_v} E(y_s y_t) \end{aligned}$$

The first term on the right hand side depends on ϕ. The modified version based only on the second term on the right hand side is

$$\sum_{(s,t)\in N_v} y_s y_t = \sum_{(s,t)\in N_v} E(y_s y_t).$$

Let x^j be the jth column of X and $H_v = (1((s,t) \in N_v))_{1 \le s,t \le N}$ where $1(\cdot)$ is 1 if the argument is true. H_v is symmetric and with 0's on its diagonal, since $(s,t) \in N_v$ iff $(t,s) \in N_v$. The modified MM equations can be written as

$$\begin{aligned} \sum_i x_{ij} y_i &= \sum_i x_{ij} E(y_i) = (x^j)' E(\mu), \quad 1 \le j \le p, \\ \sum_{(s,t) \in N_v} y_s y_t &= \sum_{(s,t) \in N_v} E(y_s y_t) = E(\mu' H_v \mu), \quad 1 \le v \le r. \end{aligned}$$

The solution to the above equations can be found with Newton-Raphson algorithm. The first derivatives are

$$\begin{aligned} \frac{\partial}{\partial \beta_k} (\sum_i x_{ij} E(y_i)) &= (x^j)' E(B) x^k, \quad 1 \le j,k \le p, \\ \frac{\partial}{\partial \sigma_{k'}} (\sum_i x_{ij} E(y_i)) &= (x^j)' E(B Z^{k'} \gamma_{k'}), \quad 1 \le j \le p, \ 1 \le k' \le q, \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial \beta_k} (\sum_{(s,t) \in N_v} E(y_s y_t)) &= 2E(\mu' B H_v B) x^k, \quad 1 \le v \le r, \ 1 \le k \le p, \\ \frac{\partial}{\partial \sigma_{k'}} (\sum_{(s,t) \in N_v} E(y_s y_t)) &= 2E(\mu' B H_v B Z^{k'} \gamma_{k'}), \quad 1 \le v \le r, \ 1 \le k' \le q, \end{aligned}$$

where $B = diag(b''(\eta_i))$.

The MM estimator is consistent but may not be efficient, Jiang and Zhang (2001) give a second step estimator which is more efficient than the MM estimator. The estimator is defined in the following framework. Let S be a N-vector of base

statistics (often a longer vector than the vector of parameter θ). Under the assumption that the mean and variance-covariance matrix of S are known functions of θ, the class of estimating functions are defined as $\mathcal{H}= \{C(S-\lambda(\theta))\}$, where C is a $(p+q)\times N$ matrix. The first-step estimator $\tilde{\theta}$ is the solution to $C(S-\lambda(\theta))=0$. A first order Taylor expansion of $C(S-\lambda(\tilde{\theta}))$ around θ is

$$C(S-\lambda(\tilde{\theta})) \approx C(S-\lambda(\theta)) - C\Lambda(\tilde{\theta}-\theta),$$

where $\Lambda = \partial\lambda(\theta)/\partial\theta'$. Since $\tilde{\theta}$ is the solution to $C(S-\lambda(\theta))=0$, then the above expression can be rearranged to $\tilde{\theta}-\theta \approx (C\Lambda)^{-1}C(S-\lambda(\theta))$. Let the variance of S be V_s, then

$$Var(\tilde{\theta}) \approx (C\Lambda)^{-1}(CV_sC')\left[(C\Lambda)^{-1}\right]'.$$

The variance of $\tilde{\theta}$ is minimized when $C=\Lambda' V_s^{-1}$. Hence, the optimal estimating equation is $\Lambda' V^{-1}(S-\lambda(\theta))=0$. With a first step estimator $\tilde{\theta}$, this suggests that a second-step estimator $\hat{\theta}$ can be obtained by solving

$$(\tilde{\Lambda}'\tilde{V}_s)^{-1}S = (\tilde{\Lambda}'\tilde{V}_s)^{-1}\lambda(\theta).$$

It can be shown that under suitable conditions the second-step estimator is consistent and has asymptotic covariance matrix $(\Lambda' V_s^{-1}\Lambda)^{-1}$. Also, simulation results show that the second-step estimator is more efficient than the first-step estimator.

For the MM estimating equations, the set of base statistics is

$$\begin{aligned} S_j &= \sum_i x_{ij} y_i, \quad 1 \le j \le p, \\ S_{p+k} &= \sum_{s \neq t} z_{sk} z_{tk} y_s y_t \quad 1 \le k \le q, \end{aligned}$$

where z_{ik} is the (i,k) element of Z. If $C = diag(I_p, 1'_{q_1}, \ldots, 1'_{q_r})$, $CS = E(CS)$ are the MM equations. The corresponding U, V_s defined above can be obtained respectively for this set of base statistics. The asymptotic covariance matrix for the second-step estimator can be used as an alternative for the inverse of the information matrix.

Assumption the variance of the random effects are known, Niaparast (2009), Niaparast and Schwabe (2013) use the above variance covariance matrix to get optimal designs for the Poisson mixed model and Waite et al.(2012) use it to get optimal designs for generalized linear models with random block effects.

For the poisson mxied model, the asymptotic variance-covariance matrix for the second step estimator has a closed-form expression.

Example 2.2.1. *In a poisson mixed effects model, suppose the response for an individual i is $Y_i = (Y_{i1}, \ldots, Y_{in_i})$ and the elements of Y_i gvien the q-vector of random effects u_i are independent from a poisson distribution*

$$p(Y_{ij} = y_{ij} | u_i) = \frac{\exp(-\mu_{ij}) {\mu_{ij}}^{y_{ij}}}{y_{ij}!},$$

where $\mu_{ij} = \exp(x'_{ij}\beta + z'_{ij}u_i)$, x_{ij} is the ith row of the model matrix X_i for fixed

effects, z_{ij} is the ith row of the model matrix Z_i for random effects, β is the p-vector of parameters for the fixed effects and $u_i \sim N_q(0, \Sigma)$ with $\Sigma = diag(\sigma_1^2, \ldots, \sigma_q^2)$. Suppose there are N individuals, the model matrix for the fixed effects is $X = (X_1', \ldots, X_N')'$ and the model matrix for the random effects is $Z = (Z_1', \ldots, Z_N')'$. Let $\sigma = (\sigma_1, \ldots, \sigma_q)'$, u_i can be written as $u_i = diag(\sigma)\gamma_i$ where $\gamma_i \sim N(0, I_q)$. Then, the likelihood is

$$\begin{aligned} L &= \prod_i \int \prod_j \frac{\exp(-\mu_{ij})\mu_{ij}{}^{y_{ij}}}{y_{ij}!} \frac{1}{(2\pi)^{q/2}} \exp(-\frac{1}{2}\gamma_i'\gamma_i)\, \mathrm{d}\gamma_i \\ &= \prod_i \int \exp\big[-\sum_j \exp(x_{ij}\beta + z_{ij}' diag(\sigma)\gamma_i) + \sum_j y_{ij}(x_{ij}'\beta + z_{ij}' diag(\sigma)\gamma_i) - \sum_j \log(y_{ij}!)\big] \\ &\quad \cdot \frac{1}{(2\pi)^{q/2}} \exp(-\frac{1}{2}\gamma_i'\gamma_i)\, \mathrm{d}\gamma_i. \end{aligned}$$

A set of sufficient statistics is given by $\big((\sum_j x_{ij}y_{ij})', (\sum_j z_{ij}y_{ij})'\big)', 1 \le i \le N$. The MM equations are

$$\begin{aligned} \sum_i \sum_j x_{ijl} y_{ij} &= \sum_i \sum_j x_{ijl} E(y_{ij}),\ 1 \le l \le p, \\ \sum_i (\sum_j z_{ijm} y_{ij})^2 &= \sum_i E(\sum_j z_{ijm} y_{ij})^2,\ 1 \le m \le q. \end{aligned}$$

Since $u_i \sim N(0, \Sigma)$, $exp(z_{ij}' u_i)$ follows a log-normal distribution. It can be shown

that

$$\begin{aligned}
E(y_{ij}) &= E(\mu_{ij}) = E\big[\exp(x'_{ij}\beta + z'_{ij}u_i)\big] = \exp(x'_{ij}\beta)E\big[\exp(z'_{ij}u_i)\big] = \exp(x'_{ij}\beta + \frac{1}{2}z'_{ij}\Sigma z_{ij}),\\
E(y_{ij}^2) &= E(\mu_{ij} + \mu_{ij}^2) = E(\exp(x'_{ij}\beta + z'_{ij}u_i) + \exp(2x'_{ij}\beta + 2z'_{ij}u_i))\\
&= \exp(x'_{ij}\beta + \frac{1}{2}z'_{ij}\Sigma z_{ij}) + \exp(2x'_{ij}\beta + 2z'_{ij}\Sigma z_{ij}),\\
E(y_{ij}y_{ij'}) &= E(\mu_{ij}\mu_{ij'}) = E\big[\exp((x_{ij} + x_{ij'})'\beta + (z_{ij} + z_{ij'})'u_i)\big]\\
&= \exp\big[(x_{ij} + x_{ij'})'\beta + \frac{1}{2}(z_{ij} + z_{ij'})'\Sigma(z_{ij} + z_{ij'})\big].
\end{aligned}$$

So the MM equations are

$$\begin{aligned}
\sum_i\sum_j x_{ijl}y_{ij} &= \sum_i\sum_j x_{ijl}E(\mu_{ij}),\ 1 \le l \le p,\\
\sum_i(\sum_j z_{ijm}y_{ij})^2 &= \sum_i(\sum_j z_{ijm}^2 E(y_{ij}^2) + \sum_{j\neq j'} z_{ijm}z_{ij'm}E(y_{ij}y_{ij'}))\\
&= \sum_i(\sum_j z_{ijm}^2 E(\mu_{ij}) + (\sum_j z_{ijm}E(\mu_{ij}))^2),\ 1 \le m \le q.
\end{aligned}$$

The set of base statistics is given by

$$\begin{aligned}
S_{il} &= \sum_j x_{ijl}y_{ij},\ 1 \le i \le N,\ 1 \le l \le p,\\
S_{i(p+m)} &= (\sum_j z_{ijm}y_{ij})^2,\ 1 \le i \le N,\ 1 \le m \le q,
\end{aligned}$$

and S is constructed as a $N \times (p+q)$ vector by stacking $S_i = (S_{i1}, \ldots, S_{i(p+q)})'$ for $1 \le i \le N$. When $C = 1'_N \otimes I_{p+q}$, $CS = E(CS)$ gives the above MM

equations. For the second-step estimator, since responses from different individuals are independent, V_s is a block diagonal matrix. The second-step equations are

$$\sum_i \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta'}\Big)'|_{\theta=\tilde{\theta}} V_i^{-1}|_{\theta=\tilde{\theta}}(S_i - \lambda_i(\theta)) = 0,$$

where $\lambda_i(\theta) = E(S_i)$ is a $p+q$ vector and the diagonal elements of V_s are given by $(p+q)\times(p+q)$ matrix $V_i = Var(S_i)$. The asymptotic covariance matrix for the second step estimator is

$$\left[\sum_i \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta'}\Big)' V_i^{-1} \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta'}\Big)\right]^{-1},$$

which has a closed-form expression.

First, we derive the expression for $\frac{\partial \lambda_i(\theta)}{\partial \theta'}$. The expectations of the base statistics are

$$\begin{aligned}
\lambda_{il}(\theta) &= E(S_{il}) = \sum_j x_{ijl} E(\mu_{ij}),\ 1 \le i \le N,\ 1 \le l \le p, \\
\lambda_{i(p+m)}(\theta) &= E(S_{i(p+m)}) = \sum_j z_{ijm}^2 E(\mu_{ij}) + \sum_j \sum_{j'} z_{ijm} z_{ij'm} E(\mu_{ij}\mu_{ij'}),\ 1 \le m \le q.
\end{aligned}$$

So elements of $\frac{\partial \lambda_i(\theta)}{\partial \theta}$ *are given by*

$$\begin{aligned}
\frac{\partial \lambda_{il}(\theta)}{\partial \theta_k} &= \frac{\partial(\lambda_{il}(\theta))}{\partial \beta_k} = \sum_j x_{ijl} x_{ijk} E(\mu_{ij}),\ 1 \le l, k \le p, \\
\frac{\partial \lambda_{il}(\theta)}{\partial \theta_{p+k'}} &= \frac{\partial(\lambda_{il}(\theta))}{\partial \sigma_{k'}} = \sum_j x_{ijl} z_{ijk'}^2 \sigma_{k'} E(\mu_{ij}),\ 1 \le k' \le q, \\
\frac{\partial \lambda_{i(p+m)}(\theta)}{\partial \theta_k} &= \frac{\partial(\lambda_{i(p+m)}(\theta))}{\beta_k} \\
&= \sum_j z_{ijm}^2 x_{ijk} E(\mu_{ij}) + \sum_j \sum_{j'} z_{ijm} z_{ij'm} (x_{ijk} + x_{ij'k}) E(\mu_{ij}\mu_{ij'}),\ 1 \le k \le p,\ 1 \le m \le \\
\frac{\partial \lambda_{i(p+m)}(\theta)}{\partial \theta_{p+k'}} &= \frac{\partial(\lambda_{i(p+m)}(\theta))}{\sigma_{k'}} \\
&= \sum_j z_{ijm}^2 z_{ijk'}^2 \sigma_{k'} E(\mu_{ij}) + \sum_j \sum_{j'} z_{ijm} z_{ij'm} (z_{ijk'} + z_{ij'k'})^2 \sigma_{k'} E(\mu_{ij}\mu_{ij'}),\ 1 \le m, k' \le q.
\end{aligned}$$

Second, We derive $E(S_i S_i')$ *in* $V_i = E(S_i S_i') - E(S_i)E(S_i')$. *Elements of* $E(S_i S_i')$ *are given by*

$$\begin{aligned}
E(S_i S_i')_{ll'} &= E(S_{il} S_{il'}) = E\big[(\sum_j x_{ijl} y_{ij})(\sum_j x_{ijl'} y_{ij})\big] \\
&= \sum_j x_{ijl} x_{ijl'} E(\mu_{ij}) + \sum_j \sum_{j'} x_{ijl} x_{ij'l'} E(\mu_{ij}\mu_{ij'}),\ 1 \le l, l' \le p,
\end{aligned}$$

$$E(S_i S_i')_{l(p+m)} = E(S_{il} S_{i(p+m)}) = E\big[(\sum_j x_{ijl} y_{ij})(\sum_j z_{ijm} y_{ij})^2\big]$$

$$
\begin{aligned}
&= \sum_j x_{ijl} z_{ijm} z_{ijm} E(y_{ij}^3) + \sum_{j \neq j'} (2 x_{ijl} z_{ij'm} z_{ijm} + x_{ij'l} z_{ijm}^2) E(y_{ij}^2 y_{ij'}) \\
&\quad + \sum_{j \neq j' \neq j''} x_{ijl} z_{ij'm} z_{ij''m} E(y_{ij} y_{ij'} y_{ij''}) \\
&= \sum_j x_{ijl} z_{ijm} z_{ijm} E(\mu_{ij} + 3\mu_{ij}^2 + \mu_{ij}^3) + \sum_{j \neq j'} (2 x_{ijl} z_{ij'm} z_{ijm} + x_{ij'l} z_{ijm}^2) E(\mu_{ij}\mu_{ij'} + \mu_{ij}^2 \\
&\quad + \sum_{j \neq j' \neq j''} x_{ijl} z_{ij'm} z_{ij''m} E(\mu_{ij} \mu_{ij'} \mu_{ij''}) \\
&= \sum_j x_{ijl} z_{ijm} z_{ijm} E(\mu_{ij}) + \sum_j \sum_{j'} (2 x_{ijl} z_{ij'm} z_{ijm} + x_{ij'l} z_{ijm}^2) E(\mu_{ij}\mu_{ij'}) \\
&\quad + \sum_j \sum_{j'} \sum_{j''} x_{ijl} z_{ij'm} z_{ij''m} E(\mu_{ij}\mu_{ij'}\mu_{ij''}),\ 1 \leq l \leq p,\ 1 \leq m \leq q, \\
E(S_i S_i')_{(p+m)l} &= (E(S_i S_i'))_{l(p+m)},\ 1 \leq l \leq p,\ 1 \leq m \leq q,
\end{aligned}
$$

$$
\begin{aligned}
& E(S_i S_i')_{(p+m)(p+m')} \\
= \;& E(S_{i(p+m)} S_{i(p+m')}) = E\big[(\sum_j z_{ijm} y_{ij})^2 (\sum_j z_{ijm'} y_{ij})^2\big] \\
= \;& \sum_j z_{ijm}^2 z_{ijm'}^2 E(y_{ij}^4) + \sum_{j \neq j'} (2 z_{ijm}^2 z_{ijm'} z_{ij'm'} E(y_{ij}^3 y_{ij'}) + 2 z_{ij'm} z_{ijm} z_{ijm'}^2 E(y_{ij'} y_{ij}^3)) \\
& + \sum_{j \neq j'} (z_{ijm}^2 z_{ij'm'}^2 E(y_{ij}^2 y_{ij'}^2) + 2 z_{ijm} z_{ij'm} z_{ijm'} z_{ij'm'} E(y_{ij}^2 y_{ij'}^2)) \\
& + \sum_{j \neq j' \neq j''} (z_{ijm}^2 z_{ij'm'} z_{ij''m'} + 4 z_{ijm} z_{ij'm} z_{ijm'} z_{ij''m'} + z_{ij'm} z_{ij''m} z_{ijm'}^2) E(y_{ij}^2 y_{ij'} y_{ij''}) \\
& + \sum_{j \neq j' \neq j'' \neq j'''} z_{ijm} z_{ij'm} z_{ij''m'} z_{ij'''m'} E(y_{ij} y_{ij'} y_{ij''} y_{ij'''}) \\
= \;& \sum_j z_{ijm}^2 z_{ijm'}^2 E(\mu_{ij} + 7\mu_{ij}^2 + 6\mu_{ij}^3 + \mu_{ij}^4) \\
& + \sum_{j \neq j'} (2 z_{ijm}^2 z_{ijm'} z_{ij'm'} + 2 z_{ij'm} z_{ijm} z_{ijm'}^2) E(\mu_{ij}(1 + 3\mu_{ij} + \mu_{ij}^2)\mu_{ij'}) \\
& + \sum_{j \neq j'} (z_{ijm}^2 z_{ij'm'}^2 + 2 z_{ijm} z_{ij'm} z_{ijm'} z_{ij'm'}) E(\mu_{ij}(1 + \mu_{ij})\mu_{ij'}(1 + \mu_{ij'})) \\
& + \sum_{j \neq j' \neq j''} (z_{ijm}^2 z_{ij'm'} z_{ij''m'} + 4 z_{ijm} z_{ij'm} z_{ijm'} z_{ij''m'} + z_{ij'm} z_{ij''m} z_{ijm'}^2) E(\mu_{ij}(1 + \mu_{ij})\mu_{ij'}\mu_{ij''})
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{j\neq j'\neq j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
= &\sum_{j} z_{ijm}^2 z_{ijm'}^2 E(\mu_{ij}+7\mu_{ij}^2+6\mu_{ij}^3+\mu_{ij}^4) \\
&+ \sum_{j\neq j'}(2z_{ijm}^2 z_{ijm'}z_{ij'm'}+2z_{ij'm}z_{ijm}z_{ijm'}^2)E(\mu_{ij}\mu_{ij'}+3\mu_{ij}^2\mu_{ij'}+\mu_{ij}^3\mu_{ij'}) \\
&+ \sum_{j\neq j'}(z_{ijm}^2 z_{ij'm'}^2+2z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'})E(\mu_{ij}\mu_{ij'}+\mu_{ij}^2\mu_{ij'}+\mu_{ij}\mu_{ij'}^2+\mu_{ij}^2\mu_{ij'}^2) \\
&+ \sum_{j\neq j'\neq j''}(z_{ijm}^2 z_{ij'm'}z_{ij''m'}+4z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}+z_{ij'm}z_{ij''m}z_{ijm'}^2)E(\mu_{ij}\mu_{ij'}\mu_{ij''}+\mu_{ij}^2\mu_{ij'}\mu_{ij''}) \\
&+ \sum_{j\neq j'\neq j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
= &\sum_{j} z_{ijm}^2 z_{ijm'}^2 E(\mu_{ij}) \\
&+ \sum_{j}\sum_{j'}(2z_{ijm}^2 z_{ijm'}z_{ij'm'}+2z_{ijm}z_{ij'm}z_{ijm'}^2+z_{ijm}^2 z_{ij'm'}^2+2z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'})E(\mu_{ij}\mu_{ij'}) \\
&+ \sum_{j}\sum_{j'}\sum_{j''}(z_{ijm}^2 z_{ij'm'}z_{ij''m'}+4z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}+z_{ij'm}z_{ij''m}z_{ijm'}^2)E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
&+ \sum_{j}\sum_{j'}\sum_{j''}\sum_{j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}),\ 1\leq m,m'\leq q.
\end{aligned}
$$

Hence, elements of $V_i = E(S_iS_i') - E(S_i)E(S_i')$ *are given by*

$$
\begin{aligned}
(V_i)_{ll'} &= E(S_{il}S_{il'}) - E(S_{il})E(S_{il'}) \\
&= \sum_j x_{ijl}x_{ijl'}E(\mu_{ij}) + \sum_j\sum_{j'} x_{ijl}x_{ij'l'}E(\mu_{ij}\mu_{ij'}) - (\sum_j x_{ijl}E(\mu_{ij}))(\sum_j x_{ijl'}E(\mu_{ij})) \\
&= \sum_j x_{ijl}x_{ijl'}E(\mu_{ij}) + \sum_j\sum_{j'} x_{ijl}x_{ij'l'}(E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'})),\ 1\leq l,l'\leq p,
\end{aligned}
$$

where $E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'}) = E(\mu_{ij})E(\mu_{ij'})(\exp(z'_{ij}\Sigma z_{ij'}) - 1)$,

$$
\begin{aligned}
(V_i)_{l(p+m)} &= E(S_{il}S_{i(p+m)}) - E(S_{il})E(S_{i(p+m)}) \\
&= \sum_j x_{ijl}z_{ijm}z_{ijm}E(\mu_{ij}) + \sum_j\sum_{j'}(2x_{ijl}z_{ij'm}z_{ijm} + x_{ij'l}z^2_{ijm})E(\mu_{ij}\mu_{ij'}) \\
&\quad + \sum_j\sum_{j'}\sum_{j''} x_{ijl}z_{ij'm}z_{ij''m}E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
&\quad -(\sum_j x_{ijl}E(\mu_{ij}))(\sum_j z^2_{ijm}E(\mu_{ij}) + \sum_j\sum_{j'} z_{ijm}z_{ij'm}E(\mu_{ij}\mu_{ij'})) \\
&= \sum_j x_{ijl}z_{ijm}z_{ijm}E(\mu_{ij}) + 2\sum_j\sum_{j'} x_{ijl}z_{ij'm}z_{ijm}E(\mu_{ij}\mu_{ij'}) \\
&\quad + \sum_j\sum_{j'} x_{ij'l}z^2_{ijm}(E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'})) \\
&\quad + \sum_j\sum_{j'}\sum_{j''} x_{ijl}z_{ij'm}z_{ij''m}(E(\mu_{ij}\mu_{ij'}\mu_{ij''}) - E(\mu_{ij})E(\mu_{ij'}\mu_{ij''})), \\
(V_i)_{(p+m)l} &= (V_i)_{l(p+m)},\ 1 \le l \le p,\ 1 \le m \le q,
\end{aligned}
$$

where

$$E(\mu_{ij}\mu_{ij'}\mu_{ij''}) - E(\mu_{ij})E(\mu_{ij'}\mu_{ij''}) = E(\mu_{ij})E(\mu_{ij'}\mu_{ij''})(\exp(\frac{1}{2}(z_{ij}+z_{ij'})'\Sigma(z_{ij}+z_{ij''})) - 1),$$

and

$$
\begin{aligned}
& (V_i)_{(p+m)(p+m')} \\
= \; & E(S_{i(p+m)}S_{i(p+m')}) - E(S_{i(p+m)})E(S_{i(p+m')}) \\
= \; & \sum_j z_{ijm}^2 z_{ijm'}^2 E(\mu_{ij}) \\
& + \sum_j \sum_{j'} (2z_{ijm}^2 z_{ijm'} z_{ij'm'} + 2z_{ijm} z_{ij'm} z_{ijm'}^2 + z_{ijm}^2 z_{ij'm'}^2 + 2z_{ijm} z_{ij'm} z_{ijm'} z_{ij'm'}) E(\mu_{ij}\mu_{ij'}) \\
& + \sum_j \sum_{j'} \sum_{j''} (z_{ijm}^2 z_{ij'm'} z_{ij''m'} + 4z_{ijm} z_{ij'm} z_{ijm'} z_{ij''m'} + z_{ij'm} z_{ij''m} z_{ijm'}^2) E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
& + \sum_j \sum_{j'} \sum_{j''} \sum_{j'''} z_{ijm} z_{ij'm} z_{ij''m'} z_{ij'''m'} E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
& - (\sum_j z_{ijm}^2 E(\mu_{ij}) + \sum_j \sum_{j'} z_{ijm} z_{ij'm} E(\mu_{ij}\mu_{ij'}))(\sum_j z_{ijm'}^2 E(\mu_{ij}) + \sum_j \sum_{j'} z_{ijm'} z_{ij'm'} E(\mu_{ij}\mu_{ij'})) \\
= \; & \sum_j z_{ijm}^2 z_{ijm'}^2 E(\mu_{ij})
\end{aligned}
$$

$$
\begin{aligned}
& + \sum_j \sum_{j'} (2z_{ijm}^2 z_{ijm'} z_{ij'm'} + 2z_{ijm} z_{ij'm} z_{ijm'}^2 + 2z_{ijm} z_{ij'm} z_{ijm'} z_{ij'm'}) E(\mu_{ij}\mu_{ij'}) \\
& + \sum_j \sum_{j'} z_{ijm}^2 z_{ij'm'}^2 (E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'})) \\
& + \sum_j \sum_{j'} \sum_{j''} 4z_{ijm} z_{ij'm} z_{ijm'} z_{ij''m'} E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
& + \sum_j \sum_{j'} \sum_{j''} (z_{ijm}^2 z_{ij'm'} z_{ij''m'} + z_{ij'm} z_{ij''m} z_{ijm'}^2)(E(\mu_{ij}\mu_{ij'}\mu_{ij''}) - E(\mu_{ij})E(\mu_{ij'}\mu_{ij''}))) \\
& + \sum_j \sum_{j'} \sum_{j''} \sum_{j'''} z_{ijm} z_{ij'm} z_{ij''m'} z_{ij'''m'} (E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) - E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''})), \; 1 \le m, m' \le
\end{aligned}
$$

where $E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) - E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''}) = E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''})\big[\exp((z_{ij} + z_{ij'})'\Sigma(z_{ij''} + z_{ij'''})) - 1\big]$.

If modified MM equations are used, the following shows that only conditional mean and conditional variance need to be specified, while the unmodified equations requires conditional third and fourth moments.

$$\begin{aligned}
\sum_i \sum_j x_{ijl} y_{ij} &= \sum_i \sum_j x_{ijl} E(y_{ij}),\ 1 \le l \le p, \\
\sum_i (\sum_{j \ne j'} z_{ijm} z_{ij'm} y_{ij} y_{ij'}) &= \sum_i E(\sum_{j \ne j'} z_{ijm} z_{ij'm} y_{ij} y_{ij'}),\ 1 \le m \le q.
\end{aligned}$$

So the modified MM equations are

$$\begin{aligned}
\sum_i \sum_j x_{ijl} y_{ij} &= \sum_i \sum_j x_{ijl} E(\mu_{ij}),\ 1 \le l \le p, \\
\sum_i (\sum_{j \ne j'} z_{ijm} z_{ij'm} y_{ij} y_{ij'}) &= \sum_i \sum_{j \ne j'} z_{ijm} z_{ij'm} E(\mu_{ij} \mu_{ij'}),\ 1 \le m \le q.
\end{aligned}$$

The set of base statistics is given by

$$S_{il} = \sum_j x_{ijl} y_{ij},\ 1 \le i \le N,\ 1 \le l \le p,$$

$$S_{i(p+m)} = \sum_{j \ne j'} z_{ijm} z_{ij'm} y_{ij} y_{ij'},\ 1 \le i \le N,\ 1 \le m \le q,$$

and S is constructed as a $N \times (p+q)$ vector by stacking $S_i = (S_{i1}, \ldots, S_{i(p+q)})'$, $1 \le i \le N$. If $C = 1_N' \otimes I_{p+q}$, then $CS = E(CS)$ gives the above MM equations. For the second-step estimator, since responses from different individuals are independent,

V_s is a block diagonal matrix, so the second-step equations are

$$\sum_i \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta'}\Big)'|_{\theta=\tilde{\theta}} V_i^{-1}|_{\theta=\tilde{\theta}}(S_i - \lambda_i(\theta)) = 0,$$

where $\lambda_i(\theta) = E(S_i)$ is a $p+q$ vector and $V_i = Var(S_i)$ is a $(p+q)\times(p+q)$ matrix. The asymptotic covariance matrix for the second step estimator is

$$[\sum_i \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta}\Big)' V_i^{-1} \Big(\frac{\partial \lambda_i(\theta)}{\partial \theta}\Big)]^{-1},$$

which has a closed-form expression.

First, the expression for $\frac{\partial \lambda_i(\theta)}{\partial \theta}$ is derived. The expectations of the base statistics are

$$\begin{aligned} \lambda_{il}(\theta) &= E(S_{il}) = \sum_j x_{ijl} E(\mu_{ij}),\ 1 \le i \le N,\ 1 \le l \le p, \\ (\lambda_{i(p+m)}(\theta)) &= E(S_{i(p+m)}) = \sum_{j \ne j'} z_{ijm} z_{ij'm} E(\mu_{ij}\mu_{ij'}),\ 1 \le i \le N,\ 1 \le m \le q. \end{aligned}$$

So elements of $\frac{\partial \lambda_i(\theta')}{\partial \theta}$ are given by

$$\frac{\partial \lambda_{il}(\theta)}{\partial \theta_k} = \frac{\partial(\lambda_{il}(\theta))}{\partial \beta_k} = \sum_j x_{ijl} x_{ijk} E(\mu_{ij}),\ 1 \le l, k \le p,$$

$$\frac{\partial \lambda_{il}(\theta)}{\partial \theta_{p+k'}} = \frac{\partial(\lambda_{il}(\theta))}{\partial \sigma_{k'}} = \sum_j x_{ijl} z_{ijk'}^2 \sigma_{k'} E(\mu_{ij}),\ 1 \le m, k' \le q,$$

$$\frac{\partial \lambda_{i(p+m)}(\theta)}{\partial \theta_k} = \frac{\partial(\lambda_{i(p+m)}(\theta))}{\beta_k}$$
$$= \sum_{j \neq j'} z_{ijm} z_{ij'm} (x_{ijk} + x_{ij'k}) E(\mu_{ij}\mu_{ij'}),\ 1 \leq k \leq p,\ 1 \leq m \leq q,$$

$$\frac{\partial \lambda_{i(p+m)}(\theta)}{\partial \theta_{p+k'}} = \frac{\partial(\lambda_{i(p+m)}(\theta))}{\sigma_{k'}}$$
$$= \sum_{j \neq j'} z_{ijm} z_{ij'm} (z_{ijk'} + z_{ij'k'})^2 \sigma_{k'} E(\mu_{ij}\mu_{ij'}),\ 1 \leq m, k' \leq q.$$

Second, elements of $E(S_i S_i')$ *are given by*

$$\begin{aligned} E(S_i S_i')_{ll'} &= E(S_{il} S_{il'}) = E((\sum_j x_{ijl} y_{ij})(\sum_j x_{ijl'} y_{ij})) \\ &= \sum_j x_{ijl} x_{ijl'} E(\mu_{ij}) + \sum_j \sum_{j'} x_{ijl} x_{ij'l'} E(\mu_{ij}\mu_{ij'}),\ 1 \leq l, l' \leq p, \end{aligned}$$

$$\begin{aligned} E(S_i S_i')_{l(p+m)} &= E(S_{il} S_{i(p+m)}) = E((\sum_j x_{ijl} y_{ij})(\sum_{j \neq j'} z_{ijm} z_{ij'm} y_{ij} y_{ij'})) \\ &= 2\sum_{j \neq j'} x_{ijl} z_{ij'm} z_{ijm} E(y_{ij}^2 y_{ij'}) + \sum_{j \neq j' \neq j''} x_{ijl} z_{ij'm} z_{ij''m} E(y_{ij} y_{ij'} y_{ij''}) \\ &= 2\sum_{j \neq j'} x_{ijl} z_{ij'm} z_{ijm} E(\mu_{ij}\mu_{ij'} + \mu_{ij}^2 \mu_{ij'}) + \sum_{j \neq j' \neq j''} x_{ijl} z_{ij'm} z_{ij''m} E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\ &= 2\sum_{j \neq j'} x_{ijl} z_{ij'm} z_{ijm} E(\mu_{ij}\mu_{ij'}) + \sum_j \sum_{j' \neq j''} x_{ijl} z_{ij'm} z_{ij''m} E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\ E(S_i S_i')_{(p+m)l} &= (E(S_i S_i'))_{l(p+m)},\ 1 \leq l \leq p,\ 1 \leq m \leq q, \end{aligned}$$

$$
\begin{aligned}
E(S_iS_i')_{(p+m)(p+m')} &= \sum_{j\neq j'}\sum_{j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(y_{ij}y_{ij'}y_{ij''}y_{ij'''}) \\
&= 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(y_{ij}^2y_{ij'}^2) + 4\sum_{j\neq j'\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(y_{ij}^2y_{ij'}y_{ij''}) \\
&\quad + \sum_{j\neq j'\neq j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(y_{ij}y_{ij'}y_{ij''}y_{ij'''})
\end{aligned}
$$

$$
\begin{aligned}
&= 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(\mu_{ij}(1+\mu_{ij})\mu_{ij'}(1+\mu_{ij'})) \\
&\quad +4\sum_{j\neq j'\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(\mu_{ij}(1+\mu_{ij})\mu_{ij'}\mu_{ij''}) \\
&\quad + \sum_{j\neq j'\neq j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
&= 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(\mu_{ij}\mu_{ij'}+\mu_{ij}^2\mu_{ij'}+\mu_{ij}\mu_{ij'}^2+\mu_{ij}^2\mu_{ij'}^2) \\
&\quad +4\sum_{j\neq j'\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}+\mu_{ij}^2\mu_{ij'}\mu_{ij''}) \\
&\quad + \sum_{j\neq j'\neq j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
&= 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(\mu_{ij}\mu_{ij'}) + 4\sum_{j\neq j',j\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
&\quad +\sum_{j\neq j'}\sum_{j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
&\quad 1\leq m,m'\leq q.
\end{aligned}
$$

Hence, elements of V_i are given by

$$\begin{aligned}
(V_i)_{ll'} &= E(S_{il}S_{il'}) - E(S_{il})E(S_{il'}) \\
&= \sum_j x_{ijl}x_{ijl'}E(\mu_{ij}) + \sum_j \sum_{j'} x_{ijl}x_{ij'l'}E(\mu_{ij}\mu_{ij'}) - (\sum_j x_{ijl}E(\mu_{ij}))(\sum_j x_{ijl'}E(\mu_{ij})) \\
&= \sum_j x_{ijl}x_{ijl'}E(\mu_{ij}) + \sum_j \sum_{j'} x_{ijl}x_{ij'l'}(E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'})),\ 1 \le l, l' \le p,
\end{aligned}$$

where $E(\mu_{ij}\mu_{ij'}) - E(\mu_{ij})E(\mu_{ij'}) = E(\mu_{ij})E(\mu_{ij'})(\exp(z'_{ij}\Sigma z_{ij'}) - 1)$,

$$\begin{aligned}
(V_i)_{l(p+m)} &= E(S_{il}S_{i(p+m)}) - E(S_{il})E(S_{i(p+m)}) \\
&= 2\sum_{j \neq j'} x_{ijl}z_{ij'm}z_{ijm}E(\mu_{ij}\mu_{ij'}) + \sum_j \sum_{j' \neq j''} x_{ijl}z_{ij'm}z_{ij''m}E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
&\quad -(\sum_j x_{ijl}E(\mu_{ij}))(\sum_{j \neq j'} z_{ijm}z_{ij'm}E(\mu_{ij}\mu_{ij'}))
\end{aligned}$$

$$\begin{aligned}
&= 2\sum_j \sum_{j'} x_{ijl}z_{ij'm}z_{ijm}E(\mu_{ij}\mu_{ij'}) \\
&\quad + \sum_j \sum_{j' \neq j''} x_{ijl}z_{ij'm}z_{ij''m}(E(\mu_{ij}\mu_{ij'}\mu_{ij''}) - E(\mu_{ij})E(\mu_{ij'}\mu_{ij''})), \\
(V_i)_{(p+m)l} &= (V_i)_{l(p+m)},\ 1 \le l \le p,\ 1 \le m \le q,
\end{aligned}$$

where

$$E(\mu_{ij}\mu_{ij'}\mu_{ij''}) - E(\mu_{ij})E(\mu_{ij'}\mu_{ij''}) = E(\mu_{ij})E(\mu_{ij'}\mu_{ij''})(\exp(\frac{1}{2}(z_{ij}+z_{ij'})'\Sigma(z_{ij}+z_{ij''})) - 1),$$

and

$$
\begin{aligned}
& (V_i)_{(p+m)(p+m')} \\
= \; & E(S_{i(p+m)}S_{i(p+m')}) - E(S_{i(p+m)})E(S_{i(p+m')}) \\
= \; & 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(\mu_{ij}\mu_{ij'}) + 4\sum_{j\neq j', j\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
& + \sum_{j\neq j'}\sum_{j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) \\
& - (\sum_{j\neq j'} z_{ijm}z_{ij'm}E(\mu_{ij}\mu_{ij'}))(\sum_{j\neq j'} z_{ijm'}z_{ij'm'}E(\mu_{ij}\mu_{ij'})) \\
= \; & 2\sum_{j\neq j'} z_{ijm}z_{ij'm}z_{ijm'}z_{ij'm'}E(\mu_{ij}\mu_{ij'}) + 4\sum_{j\neq j', j\neq j''} z_{ijm}z_{ij'm}z_{ijm'}z_{ij''m'}E(\mu_{ij}\mu_{ij'}\mu_{ij''}) \\
& + \sum_{j\neq j'}\sum_{j''\neq j'''} z_{ijm}z_{ij'm}z_{ij''m'}z_{ij'''m'}(E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) - E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''})) \\
& 1 \leq m, m' \leq q,
\end{aligned}
$$

where $E(\mu_{ij}\mu_{ij'}\mu_{ij''}\mu_{ij'''}) - E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''}) = E(\mu_{ij}\mu_{ij'})E(\mu_{ij''}\mu_{ij'''})(\exp((z_{ij} + z_{ij'})'\Sigma(z_{ij''} + z_{ij'''})) - 1)$.

2.3 Summary and Discussions

In this chapter, three approximate methods, i.e., PQL, MQL and MSM, for the analysis of the generalized linear mixed models are discussed. These methods are used by Moerbeek and Mass (2005), Niaparast (2009), Niaparast and Schwabe (2013), Ogungbenro and Aarons (2011), Tekle et al. (2008) and Waite et al.(2012) to get optimal designs for generalized linear mixed models. The estimates from PQL and MQL are biased, while the estimate from MSM is consistent. Also, the

efficiency of MSM estimator can be improved by using a second step estimator. In an example, we show that the variance-covariance matrix from MSM for a poisson mixed model has a closed-form expression. The methods surveyed in this chapter will be used in chapter 4 for finding optimal designs for the panel mixed logit model, which is a special case of the generalized linear mixed models.

2.4 References

Atkinson, A. C., Donev, A. N., and Tobias, R. D. (2007). *Optimum experimental designs, with SAS* . New York: Oxford University Press.

Breslow, N. E., & Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**(421), 9−25.

Jiang, J. (1998). Consistent estimators in generalized linear mixed models. *Journal of the American Statistical Association*, **93**(442), 720−729.

Jiang, J., & Zhang, W. (2001). Robust estimation in generalised linear mixed models. *Biometrika*, **88**(3), 753−765.

Lin, X., & Breslow, N. E. (1996). Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association*, **91**(435), 1007−1016.

Liu, Q. (1993). Laplace approximations to likelihood functions for generalized linear mixed models.

McCulloch, C. E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American statistical Association*, **92**(437), 162−170.

McFadden, D. (1989). A method of simulated moments for estimation of discrete response models without numerical integration. *Econometrica: Journal of the Econometric Society*, 995−1026.

Lee, L. F. (1992). On efficiency of methods of simulated moments and maximum simulated likelihood estimation of discrete response models. *Econometric Theory*, **8**(4), 518−552.

Moerbeek, M., & Maas, C. J. (2005). Optimal experimental designs for multilevel logistic models with two binary predictors. *Communications in Statistics− Theory and Methods*, **34**(5), 1151−1167.

Niaparast, M. (2009). On optimal design for a Poisson regression model with random intercept. *Statistics & Probability Letters*, **79**(6), 741−747.

Niaparast, M., & Schwabe, R. (2013). Optimal design for quasi-likelihood estimation in Poisson regression with random coefficients. *Journal of Statistical Planning and Inference*, **143**(2), 296−306.

Ogungbenro, K., & Aarons, L. (2011). Population Fisher information matrix and optimal design of discrete data responses in population pharmacodynamic experiments. *Journal of pharmacokinetics and pharmacodynamics*, **38**(4), 449−469.

Schall, R. (1991). Estimation in generalized linear models with random effects. *Biometrika*, **78**(4), 719−727.

Tekle, F. B., Tan, F. E., & Berger, M. P. (2008). Maximin D-optimal designs for binary longitudinal responses. *Computational Statistics & Data Analysis*, **52**(12), 5253−5262.

Waite, T. W., Woods, D. C., and Waterhouse, T. H. (2012). Designs for generalized linear models with random block effects.

Chapter 3

Information Matrix for Panel Mixed Logit Model

3.1 Introduction

In marketing, transportation and health care, researchers are interested in understanding how people make their choices. Such consumer behaviors can be analyzed with discrete choice models (Train (2009), Rossi, Allenby and McCulloch (2006) and Hensher, Rose and Greene (2005)). One of the most popular discrete choice models is the multinomial logit model, but it has several limitations in representing the choice behaviors (McFadden (1974)). Recently, mixed logit models have become more popular, because they can relax assumptions in the multinomial logit model (McFadden and Train (2000), Bhat (1998), Brownstone and Train (1999), Erdem (1996), Revelt and Train (1998) and Bhat (2000)). However, mixed logit models belong to the class of generalized linear mixed models, for which designing an experiment and analyzing the data are difficult, since the likelihood functions do not have closed-form expressions (McCulloch (1997), Booth and Hobert (1999), Breslow and Clayton (1993), Wand (2007), Moerbeek and Maas (2005) and Waite

and Woods (2014)).

When respondents choose from several products, discrete choice models can be used to explore the relationship between their choices and the attributes of the products. The multinomial logit model is popular for its simple analytical form, but it assumes a homogenous population (Train (2009)). Mixed logit models (McFadden and Train (2000)) can account for the heterogeneity in the population. If respondents are asked to choose from more than one choice set, the mixed logit model used is called a panel mixed logit model (Erdem (1996), Revelt and Train (1998) and Bhat (2000)). In a panel mixed logit model, a respondent is assumed to use similar rules to make a sequence of choices, so the choices from the same respondent are correlated.

Unlike multinomial logit models, mixed logit models do not have closed-form likelihood functions, so designing an experiment and analyzing the data are difficult. For the analysis, likelihood functions are simulated by Monte Carlo methods (Revelt and Train (1998)). For the design, information matrices are often used to form criteria that measure qualities of the designs (Atkinson, Donev and Tobias (2007)). Since information matrices also do not have closed-form expressions, we need a method to evaluate information matrices.

For mixed logit models, the expression for the information matrix, which does not have a closed-form expression, is often derived and simplified first, followed by an approximation method based on the simplified expression. For the cross-sectional mixed logit model, Sándor and Wedel (2002) provide an expression for

the information matrix that makes the evaluation straightforward using Monte Carlo method. Sándor and Wedel (2002) used cross-sectional mixed logit model for panel data, where responses from the same respondent are assumed to be independent. For the panel mixed logit model, Bliemer and Rose (2010) derive an expression for the information matrix, which is more complex than that for the cross-sectional mixed logit model. Their expression is also too complex to explore the structures in the information matrix. We simplify their expression and make use of the new expression to propose more efficient methods for approximating the information matrix. With respect to a design criterion, the optimal designs are the ones that optimize the criterion and search algorithms can be used to find efficient designs. Since many information matrices are evaluated in search algorithms, efficient methods of approximating the information matrix can reduce the time of the search considerably.

In this Chapter, we will first derive the simplified expression for the information matrix under a panel mixed logit model. As in Bliemer and Rose (2010), the expression consists of two expectations, but the two expectations involved are different. For the two expectations in our expression, one is with respect to the posterior distribution of the random effects given the responses, the other is with respect to the distribution of the responses. The former is nested within the latter. We can evaluate the expression in two ways − independently or together. If the two expectations are approximated independently, the expectation with respect to the responses is considered first. Then to approximate the expectation with respect to the posterior distribution, we consider techniques from the literature of discrete

choice models and generalized linear mixed models: McCulloch (1997) and Rossi, Allenby and McCulloch (2006) use a Metropolis algorithm, Booth and Hobert (1999) use rejection sampling, McCulloch (1997) and Booth and Hobert (1999) use importance sampling, and Tierney and Kadane (1986) and Tierney, Kass and Kadane (1989) apply Laplace's method to approximate the posterior mean. We find that the Metropolis algorithm is too time consuming for approximating the information matrix, rejection sampling is not applicable for the posterior distribution considered here, and importance sampling and the Laplace approximation are viable to use here. If we consider the two expectations together, we propose another method which uses samples from the joint distribution of the responses and the random effects. The three methods, importance sampling, Laplace approximation and joint sampling, are compared in a simulation study. We find that although the Laplace approximation is not as accurate as the other two methods, it can still be used to rank designs and is much faster than the other two methods. Since our ultimate goal is to find efficient designs and not to approximate information matrices, the ranking of the designs is more important than the actual information matrices. We conclude that the Laplace approximation is the most efficient method to use in search algorithms.

The Chapter is organized as follows. In Section 3.2, we introduce the panel mixed logit model and give the simplified expression of the information matrix. Methods for approximating the information matrix are discussed in Section 3.3 and three methods are proposed. In Section 3.4, we use simulations to compare the three methods. The Chapter concludes with a discussion in Section 3.5.

3.2 Model, Information Matrix and Design Criteria

We start by introducing the formulation of the panel mixed logit model.

3.2.1 Panel Mixed Logit Model

In a typical choice experiment, there are several questions that ask the respondents to choose one from several alternatives presented to them. The set consisting of the alternatives in each question is called a choice set. From the respondents' choices in the choice sets, we can get information about the preferences of the respondents. The alternatives are identified by the level combinations of the attributes. For example, suppose a beverage has price (low and high) and volume (small and large) as attributes. One beverage with low price and small volume corresponds to a product that is different from another product—a beverage with low price and large volume.

Let S denote the number of choice sets presented to each respondent and J the number of alternatives in each choice set. Let x_{nsj} be the k-dimensional vector containing the coded levels of the q attributes for alternative j in choice set s for respondent n and denote by β_n the corresponding k-dimensional coefficient vector. The details of the coding are given in Section 3.4. Then, the coded design matrix for respondent n is given by a $SJ \times k$ matrix $X_n = (x_{n11}, x_{n12}, \ldots, x_{nSJ})'$. The corresponding response vector is given by $Y_n = (Y_{n11}, Y_{n12}, \ldots, Y_{nSJ})'$, where

$Y_{nsj} = 1$ if respondent n chooses alternative j in choice set s and $Y_{nsj} = 0$ otherwise. In each choice set, $\sum_{j=1}^{J} Y_{nsj} = 1$ where $1 \leq s \leq S$, because the respondent chooses only one alternative in each choice set.

We now introduce the panel mixed logit model. In choice set s, if β_n is given, the probability of respondent n choosing alternative j is

$$P(Y_{nsj} = 1|\beta_n) = \frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J} \exp(x'_{nsi}\beta_n)}.$$

In the above formula, β_n is assumed to be constant across the S (> 1) choice sets. Given β_n, the choices made by respondent n are independent and the conditional probability of observing a sequence of choices y_n is

$$P(Y_n = y_n|\beta_n) \quad = \quad \prod_{s=1}^{S} \prod_{j=1}^{J} \left(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J} \exp(x'_{nsi}\beta_n)} \right)^{y_{nsj}}.$$

The above expression is the probability of observing y_n in a multinomial logit model where β_n is a fixed parameter vector. In a mixed logit model, β_n is assumed to be a random vector, whose density function is $f_\theta(\beta_n)$ with θ being the vector of unknown parameters. The unconditional probability of observing y_n is

$$P_\theta(Y_n = y_n) = \int P(Y_n = y_n|\beta_n) f_\theta(\beta_n)\, \mathrm{d}\beta_n = \int \prod_{s=1}^{S} \prod_{j=1}^{J} \left(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J} \exp(x'_{nsi}\beta_n)} \right)^{y_{nsj}} f_\theta(\beta_n)\, \mathrm{d}\beta_n.$$

The above expression reflects that choices by the same respondent in different choice sets are not independent.

For a sample $y = (y_1', y_2', \ldots, y_N')'$ of N respondents, the likelihood function of θ is

$$L(\theta|Y = y) = \prod_{n=1}^{N} P_\theta(Y_n = y_n).$$

3.2.2 Information Matrix

The asymptotic variance-covariance matrix of the maximum likelihood estimator of θ is equal to the inverse of the information matrix. The information matrix can be calculated as

$$I(\theta|X) = E_Y\Big((\frac{\partial \log L(\theta|Y)}{\partial \theta})(\frac{\partial \log L(\theta|Y)}{\partial \theta})'\Big),$$

where $X = (X_1', X_2', \ldots, X_N')'$ is the $NSJ \times k$ coded design matrix for the N respondents.

Usually, β_n is assumed to be a random vector from a multivariate normal distribution $N_k(b, \Sigma)$ with $b = (b_1, b_2, \ldots, b_k)'$ and $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2)$. The normal random vector β_n can be written as $\beta_n = b + u_n$ where $u_n \sim N_k(0, \Sigma)$. Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)'$, then the vector of unknown parameters is $\theta = (b', \sigma')'$. The information matrix for θ is

$$I(\theta|X) = \sum_{n=1}^{N} \begin{pmatrix} E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \\ E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \end{pmatrix},$$

where $L_n = P_\theta(Y_n = y_n)$ is the likelihood function for respondent n and is given

by

$$\begin{aligned} P_\theta(Y_n = y_n) &= \int P_b(Y_n = y_n|u_n) f_\sigma(u_n)\,\mathrm{d}u_n \\ &= \int \prod_{s=1}^{S}\prod_{j=1}^{J}\left(\frac{\exp(x'_{nsj}(b+u_n))}{\sum_{i=1}^{J}\exp(x'_{nsi}(b+u_n))}\right)^{y_{nsj}} (2\pi)^{-k/2}|\Sigma|^{-1/2}\exp(-\frac{1}{2}u'_n\Sigma^{-1}u_n)\,\mathrm{d}u_n. \end{aligned}$$

The score function for respondent n is

$$\frac{\partial \log L_n}{\partial b} = \frac{1}{L_n}\frac{\partial L_n}{\partial b} = X'_n\big(y_n - E_{u_n}(p_n|y_n)\big), \tag{3.1}$$

where $p_n = (p'_{n1}, p'_{n2}, \ldots, p'_{nS})'$ with $p_{ns} = (p_{ns1}, p_{ns2}, \ldots, p_{nsJ})'$ and $p_{nsj} = P_b(Y_{nsj} = 1|u_n) = \frac{\exp(x'_{nsj}(b+u_n))}{\sum_{i=1}^{J}\exp(x'_{nsi}(b+u_n))}$; and

$$\frac{\partial \log L_n}{\partial \sigma} = -\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}\right)' + E_{u_n}\left[\left(\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}\right)'|y_n\right], \tag{3.2}$$

where u_{ni} is the ith element of u_n, $1 \le i \le k$. The above expressions are derived in Appendix 3.6.1

Then, it can be shown that expressions in the information matrix are given by

$$
\begin{aligned}
E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) &= X_n'\Big(E_{u_n}(\Delta_n) - E_{u_n}(p_n p_n') + E_{Y_n}\big[E_{u_n}(p_n|Y_n)E_{u_n}(p_n'|Y_n)\big]\Big)X_n, \\
E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \sigma})'\Big) &= X_n'\left(E_{u_n}\left[p_n\left(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\right)\right]\right. \\
&\qquad \left. -E_{Y_n}\left[E_{u_n}(p_n|Y_n)E_{u_n}\left(\left(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\right)|Y_n\right)\right]\right), \\
E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma})'\Big) &= -\left(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\right)'\left(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\right) \\
&\qquad + E_{Y_n}\left[E_{u_n}\left(\left(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\right)'|Y_n\right)E_{u_n}\left(\left(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\right)|Y_n\right)\right],
\end{aligned}
\tag{3.3}
$$

where $\Delta_n = \text{diag}(\Delta_{ns})$ with $\Delta_{ns} = \text{diag}(p_{ns}) - p_{ns}p_{ns}'$. These expressions are also derived in Appendix 3.6.1. They will be used to evaluate the information matrix in order to identify optimal designs, as discussed below.

3.2.3 Design Criteria

For a univariate estimator, one with a small variance is desirable. For a multivariate estimator, the generalization of variance is the variance-covariance matrix. As mentioned in Subsection 3.2.2, the asymptotic variance-covariance matrix of the maximum likelihood estimator is equal to the inverse of the information matrix. Hence, a real-valued function of the information matrix is usually used to formulate the design criterion. D-optimality is usually used as the design criterion, which seeks to minimize $\det\big[I(\theta|X)\big]^{-1/2k}$ (often called D-error in the context of discrete choice experiments) over all possible choices of X, where $2k$ is the num-

ber of parameters in θ. A-optimality is another frequently used design criterion, for which the average of the eigenvalues of $I(\theta|X)^{-1}$, i.e., the trace of $I(\theta|X)^{-1}$ divided by $2k$, is minimized.

Note that $I(\theta|X)$ depends on the parameter vector θ, which is unknown prior to the experiment. To overcome this problem, an estimated value of θ from previous studies or an educated guess can be used. Optimal designs found by this method are called locally optimal designs (Chernoff (1953)). Here, locally D-optimal designs are the designs that minimize the D-optimality criterion for a given value of θ. Similarly, the locally A-optimal designs are the designs that minimize the A-optimality criterion for a given value of θ.

3.3 Approximation of the Information Matrix

The expressions of information matrices for different respondents are the same, but different choices of X_n can be used. Hence, for the demonstration of how to approximate the information matrix, we will use X_1 $(SJ \times k)$ for respondent 1 as an example. Correspondingly, Y_1 $(SJ \times 1)$ and u_1 $(k \times 1)$ are the response and random effect for respondent 1.

The expressions in (3.3) cannot be evaluated explicitly, because they contain intractable integrals. In (3.3), the terms $E_{u_1}(\Delta_1)$, $E_{u_1}(p_1 p_1')$ and $E_{u_1}\left[p_1(\frac{u_{11}^2}{\sigma_1^3}, \ldots, \frac{u_{1k}^2}{\sigma_k^3})\right]$ only involve expectations with respect to u_1, so Monte Carlo methods can be applied directly to evaluate these terms.

However, the following terms involve additional expectations with respect to Y_1:

$$E_{Y_1}\big[E_{u_1}(p_1|Y_1)E_{u_1}(p_1'|Y_1)\big], E_{Y_1}\Big[E_{u_1}(p_1|Y_1)E_{u_1}\big((\frac{u_{11}^2}{\sigma_1^3},\ldots,\frac{u_{1k}^2}{\sigma_k^3})|Y_1\big)\Big],$$
$$\text{and } E_{Y_1}\Big[E_{u_1}\big((\frac{u_{11}^2}{\sigma_1^3},\ldots,\frac{u_{1k}^2}{\sigma_k^3})'|Y_1\big)E_{u_1}\big((\frac{u_{11}^2}{\sigma_1^3},\ldots,\frac{u_{1k}^2}{\sigma_k^3})|Y_1\big)\Big]. \tag{3.4}$$

The two layers of expectations make the approximation of these terms computationally expensive. For simplicity, we denote these terms in a general form as

$$E_{Y_1}\big[E_{u_1}(g(u_1)|Y_1)E_{u_1}(h(u_1)'|Y_1)\big],$$

where both $g(u_1)$ and $h(u_1)$ are vectors of functions of u_1. The approximation methods that we propose for such expressions can be classified into two categories, which are differentiated by whether samples of Y_1 and samples of u_1 are drawn independently or jointly. In Subsection 3.3.1 , we will discuss different methods for sampling independently, while Subsection 3.3.2 discusses the method for sampling jointly.

3.3.1 Approximations Using Samples from Marginal Distributions

For methods in this section, the approximation is done in two steps.

In the first step, a sample is drawn from the marginal distribution of Y_1 to approximate the expectation $E_{Y_1}\big[E_{u_1}(g(u_1)|Y_1)E_{u_1}(h(u_1)'|Y_1)\big]$ with respect to Y_1.

The marginal sample can be easily obtained from a joint sample, so we introduce how to get the joint sample next. The density function for the joint distribution of Y_1 and u_1 is given by $f_\theta(y_1, u_1) = P_b(Y_1 = y_1|u_1)f_\sigma(u_1)$. To get the ith sample point (y_1^i, u_1^i) from the joint distribution, first a u_1^i is drawn from $f_\sigma(u_1)$, then a y_1^i is generated from $P_b(Y_1 = y_1|u_1^i)$ in two steps:

1. In choice set s, given u_1^i the response $(Y_{1s1}, Y_{1s2}, \ldots, Y_{1sJ})'$ follows a multinomial distribution with probabilities $(p_{1s1}^i, p_{1s2}^i, \ldots, p_{1sJ}^i)'$, where $p_{1sj}^i = \frac{\exp(x_{1sj}'(b+u_1^i))}{\sum_{l=1}^J \exp(x_{1sl}'(b+u_1^i))}$. Given u_1^i, a $(y_{1s1}^i, y_{1s2}^i, \ldots, y_{1sJ}^i)'$ is simulated for each choice set s, $1 \le s \le S$.

2. Noting that given u_1^i the responses in different choice sets are independent, the ith sample y_1^i can be obtained by juxtaposing the simulated responses for all choice sets in the previous step.

Suppose the sample size is n_y, then the joint sample is $(y_1^1, u_1^1), \ldots, (y_1^{n_y}, u_1^{n_y})$. Finally, a sample of Y_1 from the marginal distribution can be obtained by using the y part in the joint sample $(y_1^1, u_1^1), \ldots, (y_1^{n_y}, u_1^{n_y})$, which is $y_1^1, \ldots, y_1^{n_y}$.

Now, $E_{Y_1}\left[E_{u_1}(g(u_1)|Y_1)E_{u_1}(h(u_1)'|Y_1)\right]$ is approximated by

$$\frac{1}{n_y}\sum_{i=1}^{n_y} E_{u_1}(g(u_1)|y_1^i)E_{u_1}(h(u_1)'|y_1^i). \tag{3.5}$$

In the second step, $E_{u_1}(g(u_1)|y_1^i)$, $1 \le i \le n_y$, is considered. Note that $E_{u_1}(g(u_1)|y_1^i)$

is a posterior mean and the posterior density is given by

$$\begin{aligned} f_\theta(u_1|y_1^i) &\propto P_b(Y_1 = y_1^i|u_1) \times f_\sigma(u_1) \\ &\propto \prod_{s=1}^{S}\prod_{j=1}^{J}\left(\frac{\exp(x'_{1sj}(b+u_1))}{\sum_{l=1}^{J}\exp(x'_{1sl}(b+u_1))}\right)^{y^i_{1sj}} \times (2\pi)^{-k/2}|\Sigma|^{-1/2}\exp(-\frac{1}{2}u_1'\Sigma^{-1}u_1). \end{aligned}$$

From the literature, the following methods can be used to approximate $E_{u_1}\big(g(u_1)|y_1^i\big)$.

1. **Metropolis Algorithm:** For generalized linear mixed models, McCulloch (1997) uses a Metropolis algorithm to take samples from the posterior distribution and then form Monte Carlo approximations to the desired posterior means in the Monte Carlo EM algorithm. Rossi, Allenby and McCulloch (2006) consider two Metropolis variants to take samples from the posterior distribution for the multinomial logit model.

   To approximate the information matrix, we need to approximate $E_{u_1}\big(g(u_1)|y_1^i\big)$ where $1 \leq i \leq n_y$, so a sample of $u_1|y_1^i$ is required for every i. Since samples drawn by this method are dependent, a large sample size is usually required for it to work. Additionally, when we search for optimal designs in Section 3.4, we also need to approximate the information matrices of a large number of designs. Hence, it is not feasible to use the Metropolis algorithm in practice for our problem.

2. **Rejection Sampling:** For generalized linear mixed models, Booth and Hobert (1999) use rejection sampling to take samples from the posterior distribution in the Monte Carlo EM algorithm. The method they use is

carried out in two steps. In step 1, a u_1^1 is drawn from $f_\sigma(u_1)$ and a w is drawn from the uniform(0,1) distribution. In step 2, if $w \leq P_b(Y_1 = y_1^i|u_1^1)/\tau$ where $\tau = \sup_{u_1} P_b(Y_1 = y_1^i|u_1)$, then u_1^1 is accepted; otherwise, start from step 1 again. This procedure stops when a desired sample size is attained. In step 2, $P_b(Y_1 = y_1^i|u_1)$ is maximized as a function of u_1.

Here, since y_1^i is the response vector from respondent 1 and the number of choice sets for a respondent cannot be very large, it is not always possible to find a u_1 that maximizes $P_b(Y_1 = y_1^i|u_1)$. Hence, the previous rejection sampling method is not applicable for the posterior distribution considered here.

3. **Importance sampling:** For generalized linear mixed models, McCulloch (1997) and Booth and Hobert (1999) also use importance sampling, with the former using it to approximate the log-likelihood and the latter for the posterior means in the EM algorithm. To approximate the likelihood function, McCulloch (1997) uses the density function of the random effects as the importance density. Booth and Hobert (1999) use a multivariate t density whose mean and variance match the mode and curvature of the posterior distribution as the importance density.

   For our problem, since the posterior mean can be written as the ratio of two expectations, importance sampling is used to approximate both expectations. Let $u_1^{i1}, u_1^{i2}, \ldots, u_1^{in_u}$ be a set of random samples from the importance density $q(u_1)$ that has the same support as $f_\theta(u_1|y_1^i)$. Then, $E_{u_1}(g(u_1)|y_1^i)$ is

approximated by

$$E_{u_1}(g(u_1)|y_1^i) \approx \frac{\sum_{j=1}^{n_u} g(u_1^{ij}) P_b(Y_1 = y_1^i|u_1^{ij}) f_\sigma(u_1^{ij})/q(u_1^{ij})}{\sum_{j=1}^{n_u} P_b(Y_1 = y_1^i|u_1^{ij}) f_\sigma(u_1^j)/q(u_1^{ij})}.$$

For our problem, we will use the density of the random effects, $f_\sigma(u_1)$, as the importance density.

As an alternative to (3.5), $E_{Y_1}\left[E_{u_1}(g(u_1)|Y_1)E_{u_1}(h(u_1)'|Y_1)\right]$ can be calculated directly as

$$\sum_{y_1^i \in A} E_{u_1}(g(u_1)|y_1^i)E_{u_1}(h(u_1)'|y_1^i)P_\theta(Y_1 = y_1^i),$$

where A is the set that contains all possible values for Y_1. In situations where the number of possible values for Y_1 is not very large, we can make use of the above expression. We only need to find a way to approximate $P_\theta(Y_1 = y_1^i)$. Since we have a sample $u_1^{i1}, u_1^{i2}, \ldots, u_1^{in_u}$ from importance density $f_\sigma(u_1)$, we can approximate $P_\theta(Y_1 = y_1^i)$ as $\frac{1}{n_u}\sum_{j=1}^{n_u} P_b(Y_1 = y_1^i|u_1^{ij})$.

4. **Laplace approximation:** Let the lth element of $g(u_1)$ be $g_l(u_1)$. Assuming for now u_1 is univariate and $g_l(u_1)$ is a smooth and positive function of u_1, the posterior mean of $g_l(u_1)$ can be written as

$$E_{u_1}[g_l(u_1)|y_1^i] = \frac{\int e^{\log g_l(u_1) + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)}\, \mathrm{d}u_1}{\int e^{\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)}\, \mathrm{d}u_1}.$$

With $Q(u_1) = \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$ and $q_l(u_1) = \log g_l(u_1) +$

$\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$, the above expression can be written as

$$E_{u_1}[g_l(u_1)|y_1^i] = \frac{\int e^{q_l(u_1)}\,\mathrm{d}u_1}{\int e^{Q(u_1)}\,\mathrm{d}u_1}.$$

Tierney and Kadane (1986) apply Laplace's method to integrals in the numerator and the denominator and obtain an approximation of the posterior mean. Let $\hat{u}_1$ be the mode of $Q(u_1)$ and $d^2 = -1/Q''(u_1)|_{u_1=\hat{u}_1}$. Then, Laplace's method approximates the integral in the denominator by

$$\int e^{Q(u_1)}\mathrm{d}u_1 \approx \int \exp\left[\frac{Q(\hat{u}_1) - (u_1 - \hat{u}_1)^2}{2d^2}\right]\mathrm{d}u_1 = \sqrt{2\pi}|d|e^{Q(\hat{u}_1)}.$$

Similarly, if $\hat{u}_{1_l}$ is the mode of $q_l(u_1)$ and ${d_l}^2 = -1/(q_l(u_1))''|_{u_1=\hat{u}_{1_l}}$, then Laplace's method approximates integral in the numerator by $\sqrt{2\pi}|d_l|\exp(q_l(\hat{u}_{1_l}))$. Taking the ratio of these two approximations, the Laplace approximation of $E_{u_1}[g_l(u_1)|y_1^i]$ is given by

$$E_{u_1}[g_l(u_1)|y_1^i] \approx \frac{|d_l|}{|d|}\exp\left[q_l(\hat{u}_{1_l}) - Q(\hat{u}_1)\right].$$

If u_1 is multivariate, a similar approximation can be obtained by

$$E_{u_1}[g_l(u_1)|y_1^i] \approx \Big(\frac{|D_l|}{|D|}\Big)^{1/2}\exp\left[q_l(\hat{u}_{1_l}) - Q(\hat{u}_1)\right],$$

where $\hat{u}_{1_l}$ and $\hat{u}_1$ maximize $q_l(u_1)$ and $Q(u_1)$ respectively, D_l is the negative of the inverse of the Hessian of $q_l(u_1)$ evaluated at $\hat{u}_{1_l}$ and D is the negative

of the inverse of the Hessian of $Q(u_1)$ evaluated at $\hat{u}_1$.

Applying this approximation to $E_{u_1}(p_{1sj}|y_1^i)$, where $1 \leq s \leq S$ and $1 \leq j \leq J$, we have

$$\begin{aligned} E_{u_1}(p_{1sj}|y_1^i) &= \frac{\int p_{1sj} P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1) \mathrm{d}u_1}{\int P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1) \mathrm{d}u_1} \\ &\approx \Big(\frac{|H_{sj}|}{|H|}\Big)^{1/2} \frac{p_{1sj} P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1)|_{u_1=\hat{u}_{1_{sj}}}}{P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1)|_{u_1=\hat{u}_1}}, \end{aligned} \tag{3.6}$$

where $\hat{u}_{1_{sj}}$ maximizes $\log p_{1sj} + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$, $\hat{u}_1$ maximizes $\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$,

$$\begin{aligned} H_{sj} &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log p_{1sj} + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_{1_{sj}}} \\ &= -(-X_{1s}'\Delta_{1s}X_{1s} - X_1'\Delta_1 X_1 - \Sigma^{-1})^{-1}|_{u_1=\hat{u}_{1_{sj}}}, \end{aligned}$$

where $X_{1s} = (x_{1s1}, x_{1s2}, \ldots, x_{1sJ})'$, and

$$\begin{aligned} H &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_1} \\ &= -(-X_1'\Delta_1 X_1 - \Sigma^{-1})^{-1}|_{u_1=\hat{u}_1}. \end{aligned}$$

The expressions are derived in Appendix 3.6.2. The previous approximation only applies to a positive function $g_l(u)$, but the elements of $(\frac{u_{11}^2}{\sigma_1^3}, \ldots, \frac{u_{1k}^2}{\sigma_k^3})$ could be zero. Tierney et al. (1989) suggest to add a large constant c to $g_l(u_1)$, so that $g_l(u_1) + c$ is a positive function. Applying this procedure to

$E\big((\frac{u_{1j}^2}{\sigma_j^3})|y_1^i\big)$, where $1 \le j \le k$, we get

$$\begin{aligned} E\Big(\frac{u_{1j}^2}{\sigma_j^3}|y_1^i\Big) &= E\Big(\frac{u_{1j}^2}{\sigma_j^3} + c|y_1^i\Big) - c \\ &\approx \Big(\frac{|H_j|}{|H|}\Big)^{1/2} \frac{\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3} P_b(Y_1 = y_1^i|u_1) \log f_\sigma(u_1)|_{u_1=\hat{u}_{1_j}}}{P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1)|_{u_1=\hat{u}_1}} - c \end{aligned} \quad (3.7)$$

where $\hat{u}_{1_j}$ maximizes $\log(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3}) + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$ and

$$\begin{aligned} H_j &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log(\frac{u_{1j}^2 + c\sigma_j^3}{\sigma_j^3}) + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_{1_j}} \\ &= -\left(\frac{2(c\sigma_j^3 - u_{1j}^2)}{(u_{1j}^2 + c\sigma_j^3)^2} e_j e_j' - X_1'\Delta_1 X_1 - \Sigma^{-1}\right)^{-1}|_{u_1=\hat{u}_{1_j}}. \end{aligned}$$

The above expressions are also derived in Appendix 3.6.2.

The Laplace approximation for $E_{u_1}(g(u_1)|y_1^i)$ should run faster than the Monte Carlo method, since optimization usually requires less computation than sampling. In addition, we do not have to decide the sample size of u_1 as in the Monte Carlo method, which is good since we also need to decide the sample size of Y_1.

3.3.2 Approximation Using Samples from the Joint Distribution

Previously, a sample from the marginal distribution of Y_1 is used and we discuss several methods to approximate posterior means with respect to u_1 given the sam-

ple of Y_1. In the second approach, a sample of size n_{yu} from the joint distribution of (Y_1, u_1) is used. The method to take samples from the joint distribution has been described in Subsection 3.3.1. We denote the joint sample as $(y_1^i, u_1^i),\ 1 \le i \le n_{yu}$.

Suppose there are M unique vectors of y_1 in the joint sample, and denote these by $z_1^1, \ldots, z_1^M$. Then, $E_{u_1}\big(g(u_1)|Y_1 = z_1^m\big)$, $1 \le m \le M$, is approximated by

$$\frac{\sum_{\{i:y_1^i = z_1^m\}} g(u_1^i)}{\#\{i : y_1^i = z_1^m\}},$$

where $\{i : y_1^i = z_1^m\}$ is a set of integers at which y_1^i is equal to z_1^m and $\#\{i : y_1^i = z_1^m\}$ is the number of elements in this set. Next, $E_{Y_1}\big[E_{u_1}(g(u_1)|y_1^i)E_{u_1}(h(u_1)'|y_1^i)\big]$ is approximated by

$$\sum_{j=1}^{M} \frac{\sum_{\{i:y_1^i = z_1^m\}} g(u_1^i)}{\#\{i : y_1^i = z_1^m\}} \frac{\sum_{\{i:y_1^i = z_1^m\}} h(u_1^i)'}{\#\{i : y_1^i = z_1^m\}} \frac{\#\{i : y_1^i = z_1^m\}}{n_{yu}}.$$

In Subsection 3.3.1, when we use importance sampling, the same sample size of n_u is used for every given y_1^i. Here, when we use the joint sampling, the sample size of u_1 for a given y_1^i is determined from the joint sample. Hence, the sample size of u_1 can be adjusted as needed. Also, we only need to decide the sample size n_{yu} for the joint sample.

3.4 Simulation

In Section 3.3, we discuss three methods to approximate the information matrix: importance sampling, Laplace approximation and joint sampling. In this section, we will compare the three methods in simulations.

We consider a case where 2 attributes of 3 levels are of interest and a design with 9 choice sets of size 2 is used for all the respondents. The number of choice sets and the number of alternatives in each choice set cannot be large due to cognitive constraints. We use $3^2/2/9$ to denote this choice design, while other choice designs considered are $3^2/3/6$, $3^2/4/5$ and $3^2/5/4$.

We use effects-type coding for the attributes (Hensher, Rose and Greene (2005)). For example, if the coefficients of the first two levels of an attribute are given by $(\beta_1, \beta_2)'$, where the attribute has 3 levels, then the coefficient of the third level is $-\beta_1 - \beta_2$. With effects-type coding, the sum of coefficients for an attribute is zero and the coefficient of each level can be interpreted as its effect relative to the average effect of the attribute, which is zero. Hence, two independent parameters are needed for an attribute of three levels. Here, with effects-type coding, the three levels of an attribute are coded as $(1, 0)$, $(0, 1)$ and $(-1, -1)$. Then, the distribution of random effects is $N_4(b, \Sigma)$, where $b = (b_1, b_2, b_3, b_4)'$ and $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2)$. The unknown parameter vector is $\theta = (b', \sigma')'$, where $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)'$. Following Arora and Huber (2001), Toubia et al. (2004) and Yu et al. (2011), values of the parameters are varied in terms of response accuracy and respondent heterogeneity. We take $b = (a, 0, a, 0)'$, where $a = .5$ is used to represent low response accuracy

and $a = 3$ is used to represent high response accuracy. With this specification, it is implied that the mean for the third level is $-a$ for each attribute. Arora and Huber (2001) state that it is more meaningful to select the variance relative to the mean. As in Toubia et al. (2004), we take $\sigma = (\sqrt{3a}, \sqrt{3a}, \sqrt{3a}, \sqrt{3a})'$ in the case of high respondent heterogeneity and $\sigma = (\sqrt{0.5a}, \sqrt{0.5a}, \sqrt{0.5a}, \sqrt{0.5a})'$ in the case of low respondent heterogeneity. Thus, the 4 sets of parameter values used in our simulations are (a) high accuracy and high heterogeneity: $b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$, (b) high accuracy and low heterogeneity: $b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$, (c) low accuracy and high heterogeneity: $b = (0.5, 0, 0.5, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$, and (d) low accuracy and low heterogeneity: $b = (0.5, 0, 0.5, 0)'$ and $\sigma = (0.5, 0.5, 0.5, 0.5)'$.

We are only interested in finding good designs, so the (dis)similarities of the three methods are compared on good designs. For a choice design with given values of the parameters, we handpick 100 good designs and approximate the information matrices for these designs using the three methods. The 100 designs are good designs from a computer search (We use a coordinate exchange algorithm with the Laplace approximation, A-optimality, and a sample size of $n_y = 10000$. The setting of the coordinate exchange algorithm is chosen based on preliminary simulation results.).

In the simulation, we use large sample sizes for the three methods so that the approximated values have stabilized and would have very small variation. For importance sampling, if there are 9 choice sets of size 2, there are $2^9 = 512$ possible values for Y. Since 512 is not a large number in this context, instead of taking

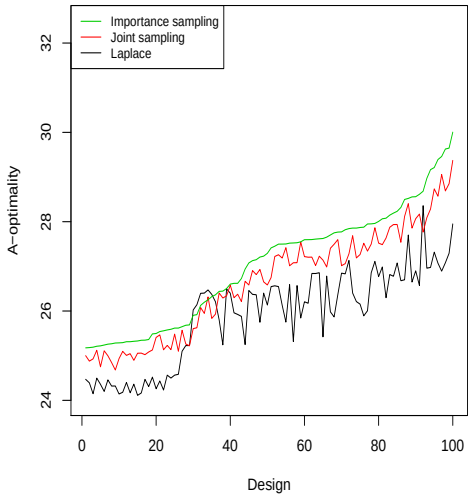
a sample of Y, we use all possible values of Y with $n_u = 10^6$ in the simulation. We can also use all possible values of Y in the other cases (6 choice sets of size 3, 5 choice sets of size 4 and 4 choice sets of size 5). For joint sampling, we use $n_{yu} = 10^6$. For the Laplace method, we use $n_y = 10^6$. Importance sampling is considered to be the most accurate method because we use all possible values for Y and use 10^6 as the sample size for u.

Importance sampling and joint sampling are Monte Carlo methods, so the simulated information matrices will converge to the information matrices if the corresponding sample sizes (n_y and n_u for importance sampling and n_{yu} for joint sampling) go to infinity. Since the Laplace approximation is a combination of Monte Carlo method and Laplace's method, the simulated information matrices will not converge to the information matrices, but to the approximations of the information matrices, when the sample size (n_y for the Laplace method) goes to infinity. Our eventual goal is to find optimal designs, and not the actual values of the information matrices. Thus, we only want to see whether the three methods can rank the designs similarly.

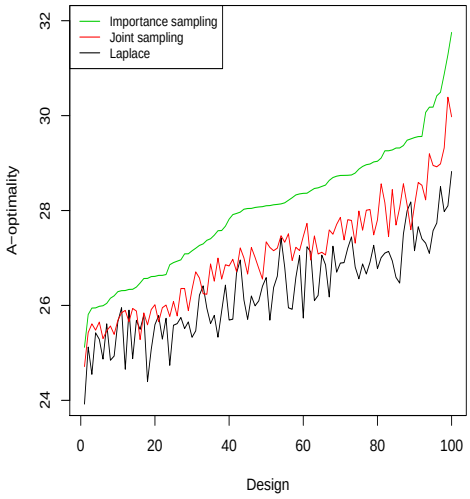
Figures 3.1 to 3.8 show the comparisons of the three methods for cases $3^2/2/9$, $3^2/3/6$, $3^2/4/5$ and $3^2/5/4$. The 100 designs are ordered by the values from importance sampling and the x-axis gives the order of the designs. We can see that values from importance sampling and joint sampling are very close. Although values from the Laplace approximation are different from values from the other two methods, the patterns are similar. The three methods largely agree in ordering those 100 good designs.

Another way to assess agreement between the three methods is by studying pairwise correlations of values for a given criterion for the 100 designs. The scatter plot of values from any two of the methods shows that there is a linear pattern. The closer the scatter plot resembles a straight line, the more the two methods would agree in ordering the designs. Correlations depend on the 100 designs used here, since it is more difficult to get high correlations when the designs are similar. Hence, the correlation cannot be used as a useful measure of how the three methods agree. Table 3.1 shows the correlations between any two of the methods. We see that the correlations between importance sampling and joint sampling are larger than 0.9 in all cases. When the accuracy is high and heterogeneity is high, the correlations between the Laplace method and the other two methods are lower, except for $3^2/5/4$ with A-optimality. When the accuracy is high and the heterogeneity is low, the correlations between the Laplace method and the other two methods are lower, which are around 0.8, in $3^2/2/9$, $3^2/3/6$ and $3^2/4/5$ and all with A-optimality. For these two sets of parameter values, the correlations between the Laplace method and the other two methods are larger in $3^2/5/4$ than in $3^2/2/9$. For the other two sets of parameter values, the correlations between the Laplace method and the other two methods are higher than 0.90. This table is consistent with what we observe in Figures 3.1 to 3.8.

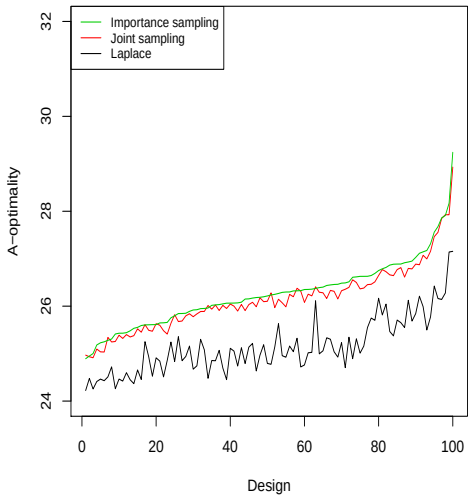
In order to use the three methods in practice, we need to find appropriate sample sizes for the methods. For each method, relative differences are used to show how values change with sample sizes. We will use the $3^2/5/4$ case with $b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$ as an example for illustration. For importance sampling,



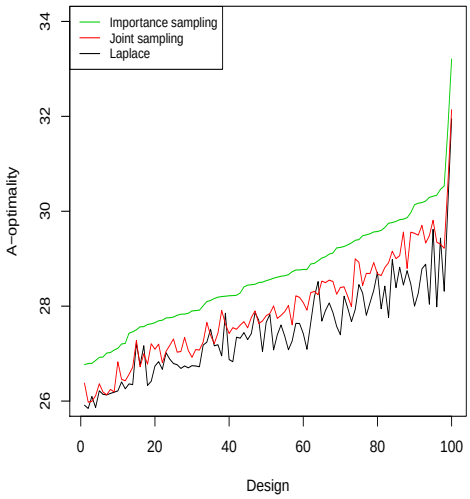
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

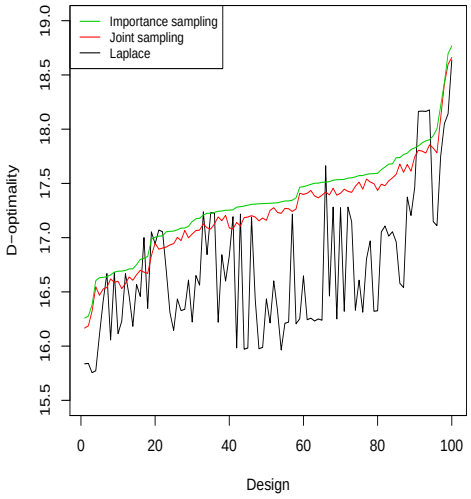


(c) 5 choice sets of size 4

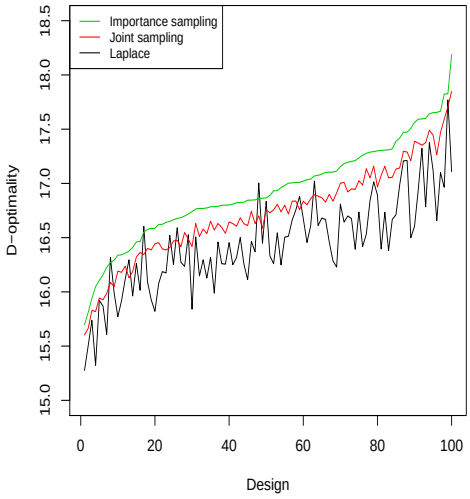


(d) 4 choice sets of size 5

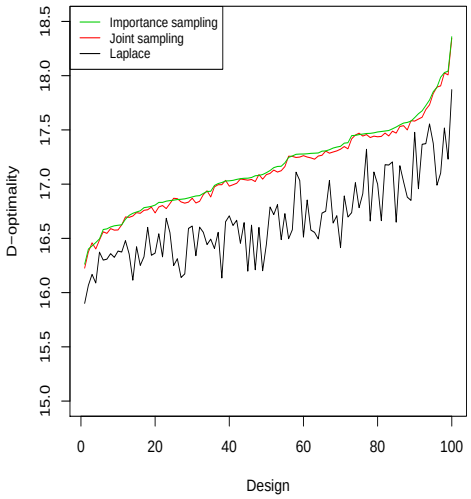
Figure 3.1: Comparisons of the three methods with A-optimality when the response accuracy is high and the respondent heterogeneity is high.



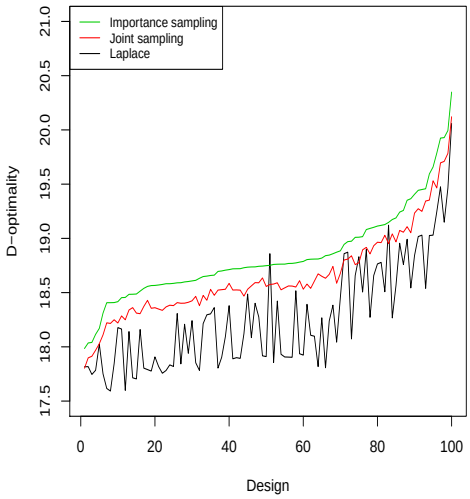
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

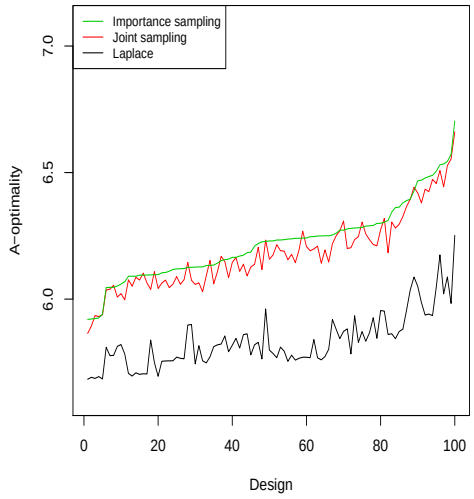


(c) 5 choice sets of size 4

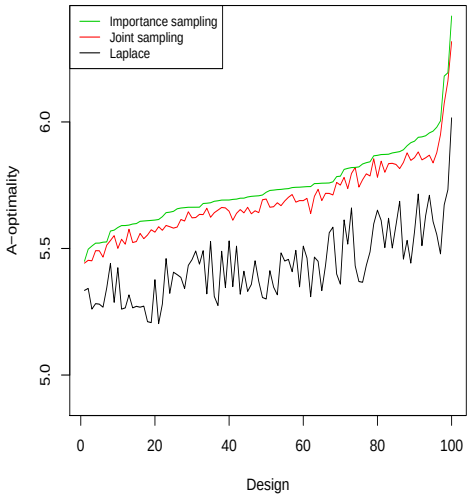


(d) 4 choice sets of size 5

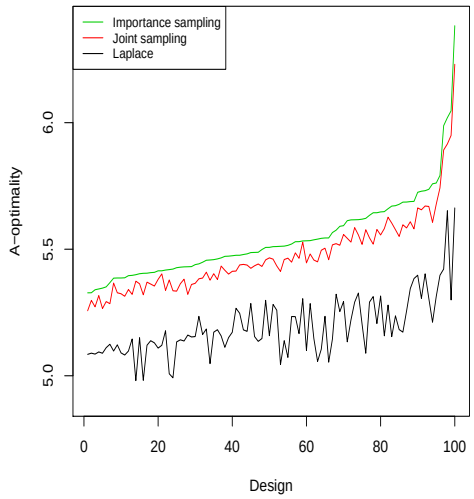
Figure 3.2: Comparisons of the three methods with D-optimality when the response accuracy is high and the respondent heterogeneity is high.



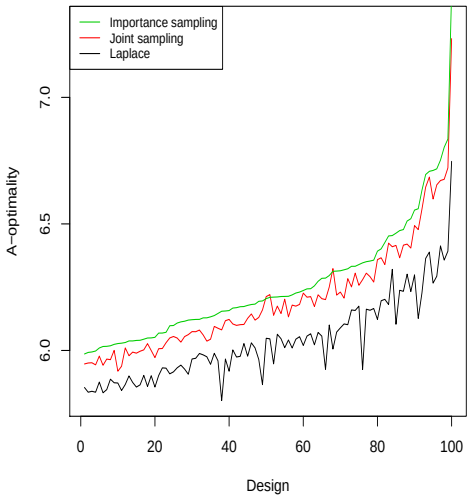
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

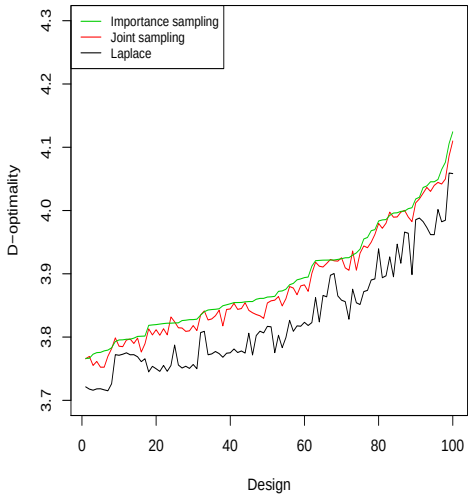


(c) 5 choice sets of size 4

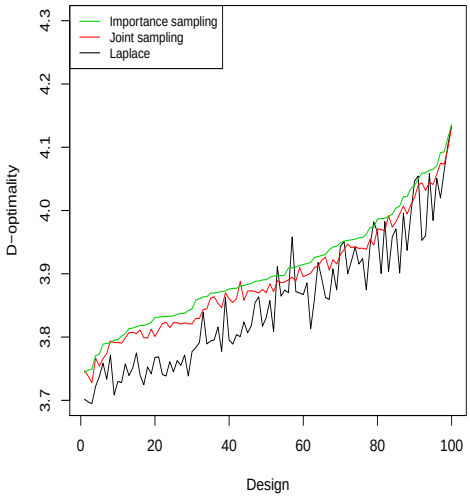


(d) 4 choice sets of size 5

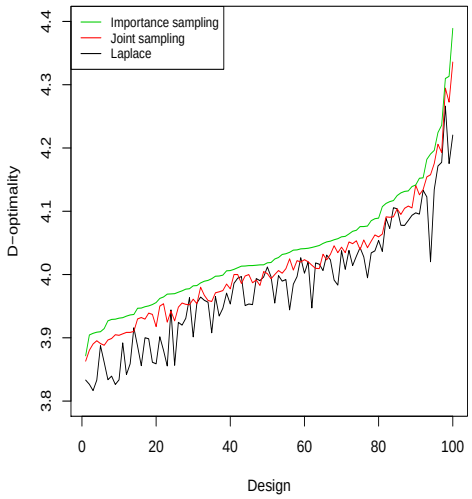
Figure 3.3: Comparisons of the three methods with A-optimality when the response accuracy is high and the respondent heterogeneity is low.



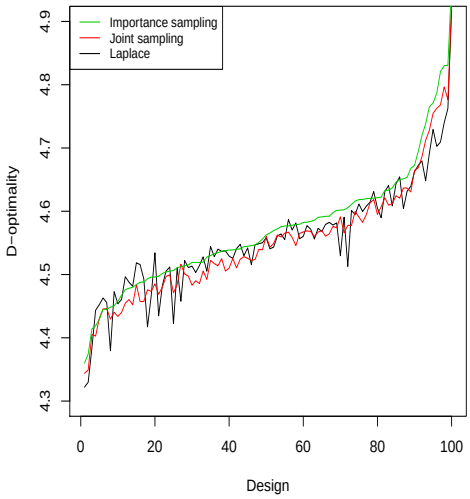
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

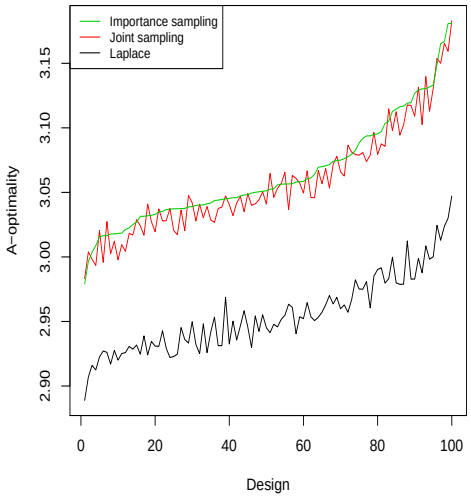


(c) 5 choice sets of size 4

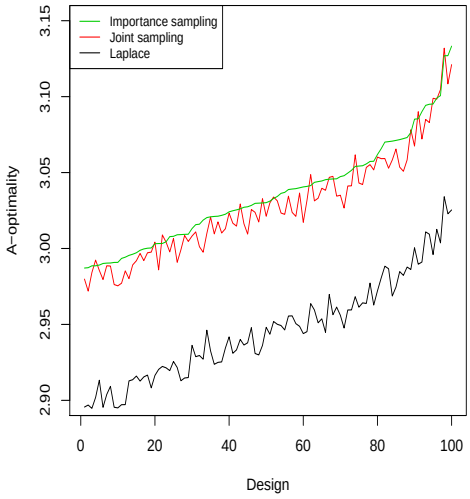


(d) 4 choice sets of size 5

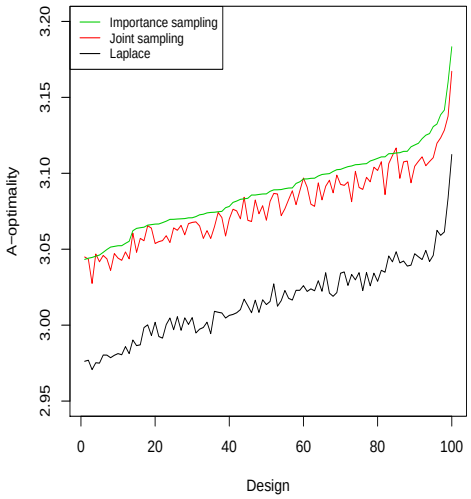
Figure 3.4: Comparisons of the three methods with D-optimality when the response accuracy is high and the respondent heterogeneity is low.



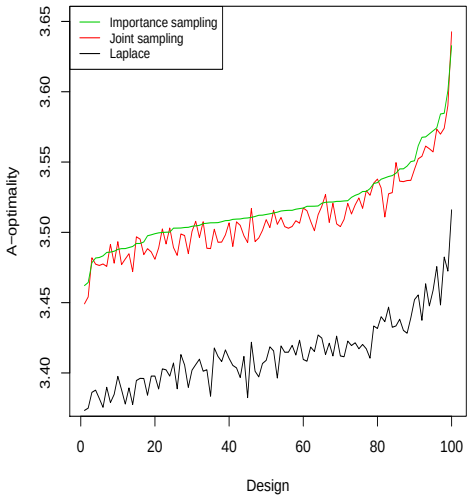
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

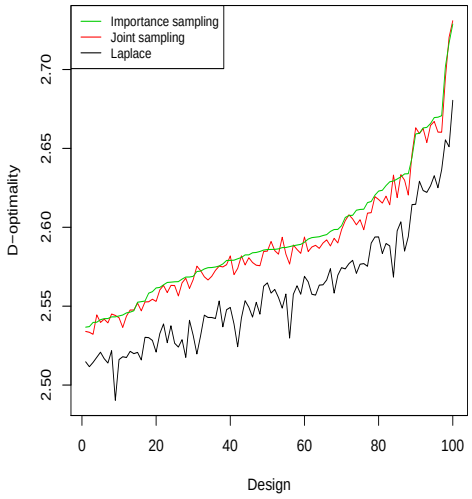


(c) 5 choice sets of size 4

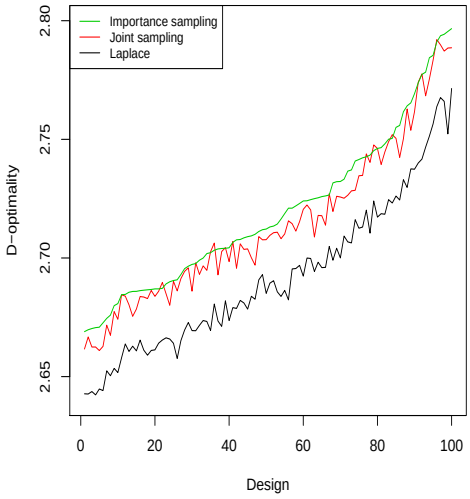


(d) 4 choice sets of size 5

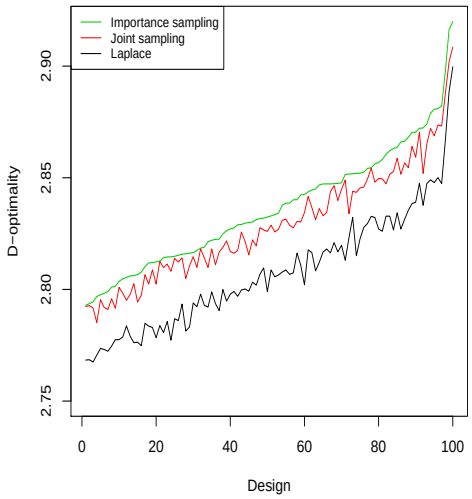
Figure 3.5: Comparisons of the three methods with A-optimality when the response accuracy is low and the respondent heterogeneity is high.



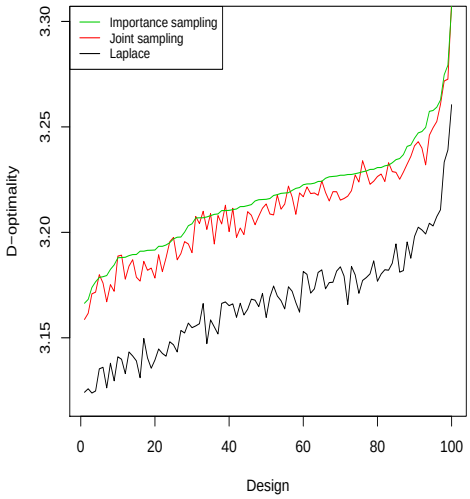
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

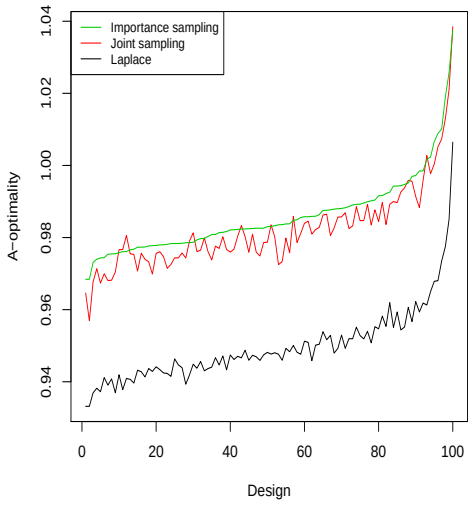


(c) 5 choice sets of size 4

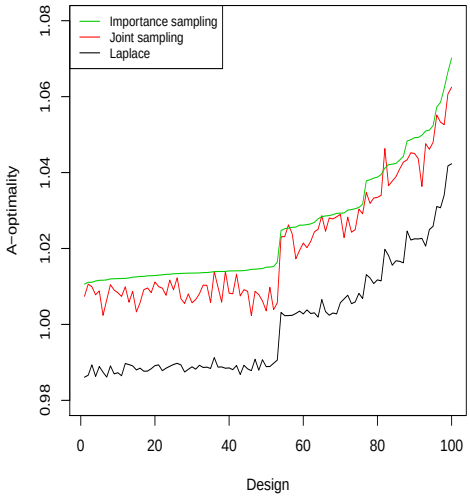


(d) 4 choice sets of size 5

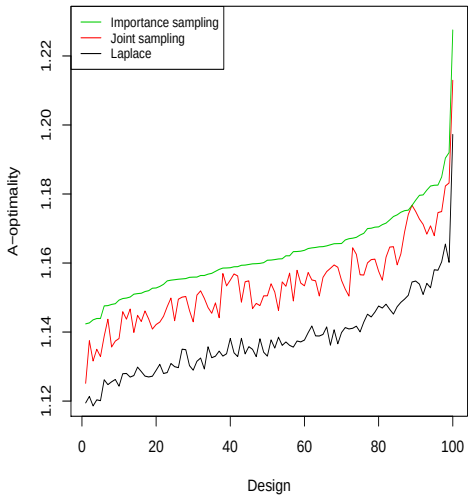
Figure 3.6: Comparisons of the three methods with D-optimality when the response accuracy is low and the respondent heterogeneity is high.



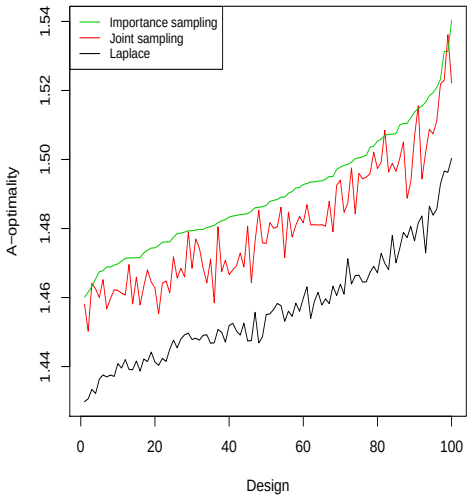
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3

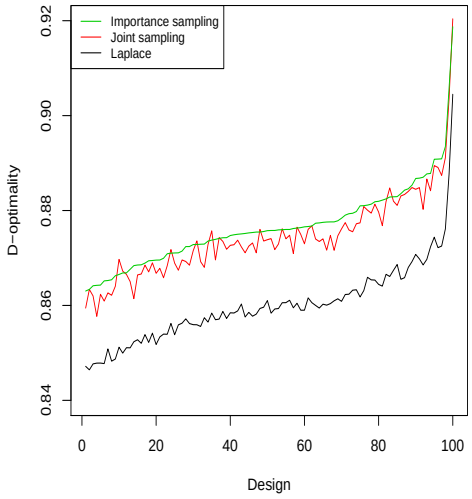


(c) 5 choice sets of size 4

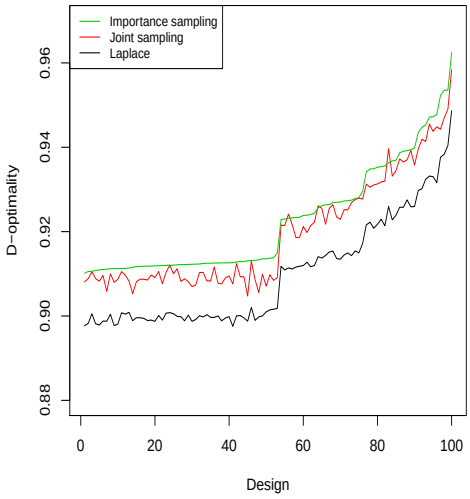


(d) 4 choice sets of size 5

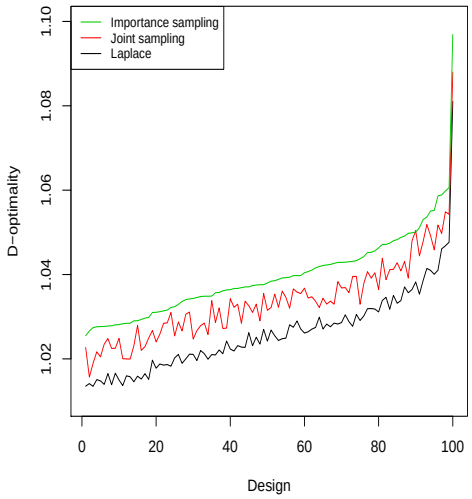
Figure 3.7: Comparisons of the three methods with A-optimality when the response accuracy is low and the respondent heterogeneity is low.



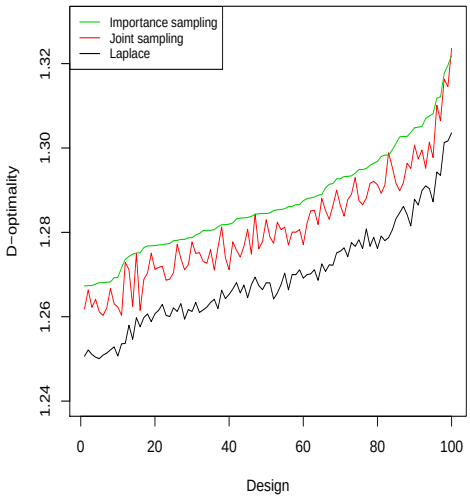
(a) 9 choice sets of size 2



(b) 6 choice sets of size 3



(c) 5 choice sets of size 4



(d) 4 choice sets of size 5

Figure 3.8: Comparisons of the three methods with D-optimality when the response accuracy is low and the respondent heterogeneity is low.

Table 3.1: Correlations between the three methods

		9 choice sets of size 2				6 choice sets of size 3			
		hh	hl	lh	ll	hh	hl	lh	ll
A-optimality	Importance -Joint	0.98	0.97	0.98	0.97	0.97	0.99	0.98	0.98
	Importance -Laplace	0.88	0.84	0.96	0.98	0.89	0.80	0.98	0.99
	Joint -Laplace	0.88	0.86	0.94	0.96	0.87	0.80	0.97	0.98
D-optimality	Importance -Joint	≈ 1	≈ 1	0.99	0.98	0.99	≈ 1	0.99	0.99
	Importance -Laplace	0.64	0.97	0.98	0.99	0.84	0.96	0.99	≈ 1
	Joint -Laplace	0.64	0.98	0.97	0.98	0.86	0.96	0.99	0.99
		5 choice sets of size 4				4 choice sets of size 5			
		hh	hl	lh	ll	hh	hl	lh	ll
A-optimality	Importance -Joint	0.99	0.99	0.97	0.96	0.98	0.99	0.97	0.95
	Importance -Laplace	0.88	0.78	0.98	0.99	0.94	0.95	0.94	0.99
	Joint -Laplace	0.88	0.80	0.97	0.95	0.94	0.95	0.93	0.94
D-optimality	Importance -Joint	≈ 1	≈ 1	0.99	0.97	≈ 1	≈ 1	0.98	0.97
	Importance -Laplace	0.86	0.94	0.99	0.99	0.86	0.96	0.97	0.99
	Joint -Laplace	0.86	0.95	0.98	0.97	0.86	0.96	0.97	0.97

Note: hh represents high accuracy and high heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$), hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$), lh represents low accuracy and high heterogeneity ($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$) and ll represents low accuracy and low heterogeneity ($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (0.5, 0.5, 0.5, 0.5)'$).

sample sizes considered are 5000, 10000, ..., 40000. For joint sampling, sample sizes considered are 50000, 100000, ..., 400000. For the Laplace approximation, sample sizes considered are 1000, 2000, ..., 9000. For each method, the relative differences between values from a small sample size and values from the largest sample size (which were also used in the previous simulation, i.e., all possible values of Y with $n_u = 10^6$ for importance sampling, $n_{yu} = 10^6$ for joint sampling and $n_y = 10^6$ for the Laplace method) are calculated. Figure 3.9 shows the relative differences of values in A-optimality and D-optimality for the three methods for 100 designs. The 100 designs are the same as those used previously for the $3^2/5/4$ case with $b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$. We conclude that it suffices to take $n_u = 20000$ for importance sampling, $n_{yu} = 250000$ for joint sampling and $n_y = 3000$ for the Laplace approximation. After these sample sizes, the improvements in the mean and variance of the relative differences become smaller as sample sizes increase. For the other cases, similar conclusions hold. Thus, we can use $n_u = 20000$ for importance sampling, $n_{yu} = 250000$ for joint sampling and $n_y = 3000$ for the Laplace approximation for all the cases considered.

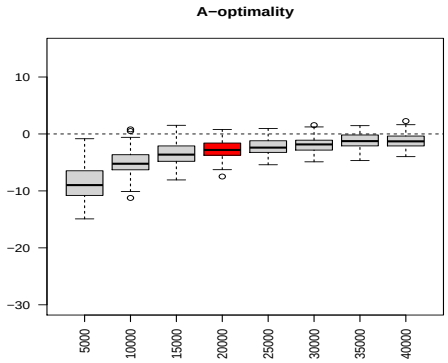
Table 3.2 shows the running time that the three methods take to approximate the information matrices for 100 designs with the reduced sample sizes. We can see that the Laplace approximation is about 3 times faster than importance sampling and 10 times faster than joint sampling. Note that here all possible values of Y are used for importance sampling. When this is not possible, we need to sample Y, making importance sampling slower, and the advantage of the Laplace approximation in running time will be larger. Another advantage of the Laplace

approximation is that only the sample size of Y needs to be decided. For importance sampling with a large number of possible Y values, sample sizes of Y and u are varied simultaneously to find the appropriate ones. For joint sampling, n_{yu} is often much larger than n_y for the Laplace approximation, so it takes more time to find the appropriate sample size. For a given choice experiment, we can see that the time of joint sampling changes with the values of the parameters. The time is shorter for the cases with high response accuracy. In these cases, the mass of Y concentrates on a small proportion of possible values of Y. The algorithm that counts the unique values of Y in the joint sample runs faster when the mass of Y concentrates on a small proportion of possible values of Y than when it is more evenly distributed over possible values of Y.

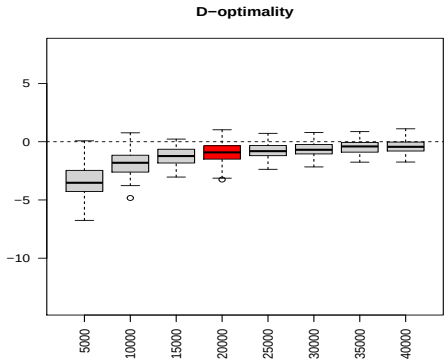
Table 3.2: Time for evaluating 100 designs using the three methods

	9 choice sets of size 2				6 choice sets of size 3			
	hh	hl	lh	ll	hh	hl	lh	ll
Importance, $n_u = 20000$	42m	43m	42m	42m	65m	60m	64m	61m
Joint, $n_{yu} = 250000$	169m	172m	298m	361m	172m	187m	417m	506m
Laplace, $n_y = 3000$	20m	20m	21m	27m	26m	26m	27m	29m
	5 choice sets of size 4				4 choice sets of size 5			
	hh	hl	lh	ll	hh	hl	lh	ll
Importance, $n_u = 20000$	90m	94m	83m	81m	56m	50m	58m	57m
Joint, $n_{yu} = 250000$	209m	198m	499m	630m	173m	166m	405m	487m
Laplace, $n_y = 3000$	30m	32m	30m	35m	33m	35m	30m	32m

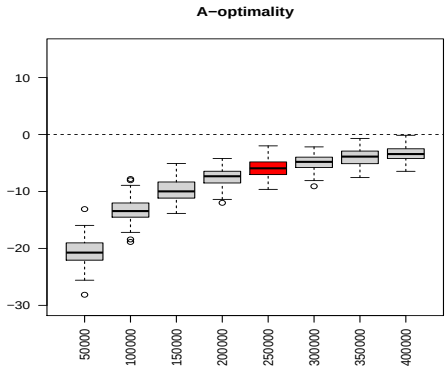
Note: hh represents high accuracy and high heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$), hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$), lh represents low accuracy and high heterogeneity ($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$) and ll represents low accuracy and low heterogeneity ($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (0.5, 0.5, 0.5, 0.5)'$).



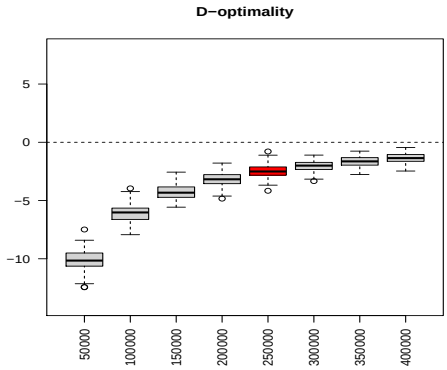
(a) Importance sampling



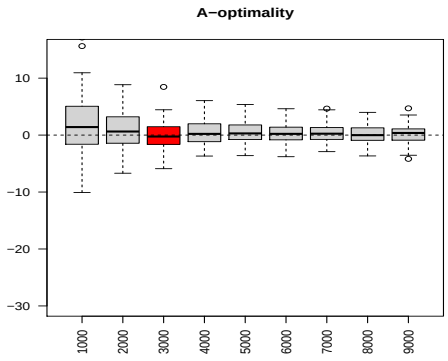
(b) Importance sampling



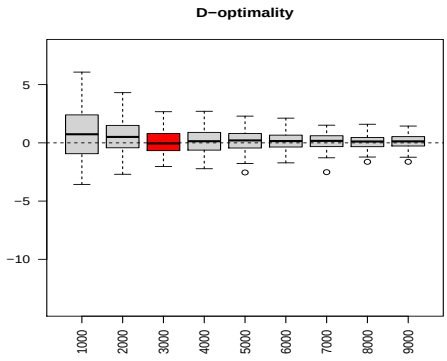
(c) Joint sampling



(d) Joint sampling



(e) The Laplace method



(f) The Laplace method

Figure 3.9: Relative difference (in %) between values from a sample size on the x-axis and the values from the largest sample size for the $3^2/5/4$ case with $b = (-3, 0, -3, 0)'$ and $\sigma = (3, 3, 3, 3)'$.

3.5 Discussion and Conclusion

For the panel mixed logit model, the information matrix has a complex form and cannot be written in a closed-form expression. We propose three methods to approximate the information matrix: importance sampling, Laplace approximation and joint sampling. For importance sampling, a sample of Y and a sample of u are taken independently, so the sample sizes of the two samples can be changed separately to adjust the precision of the approximation. When the number of possible values for Y is not large, all possible values of Y can be used, which makes the method more efficient. For joint sampling, the sample size for the joint sample is varied to adjust the accuracy of the approximation. From the simulation results, the running time for joint sampling is much longer than for the other two methods. For the Laplace approximation, although it is not as accurate as the other two methods, it ranks designs similarly and is much faster than the other two methods. For finding optimal designs, this ordering is the most important thing. Moreover, when search algorithms are used to find efficient designs, the number of information matrices to be evaluated will be much greater than 100 considered in our simulation and the search algorithm can take days, so using an efficient method to evaluate the information matrix is very important. For larger choice designs, importance sampling and joint sampling may not be practical and the Laplace approximation may be the only viable method to use. Another advantage of the Laplace approximation is that only the sample size of Y needs to be decided. It is easier and faster to get an appropriate sample size for the Laplace

approximation.

3.6 Appendix

3.6.1 Information Matrix for Panel Mixed Logit Model

We will show the validity of the expressions for $\frac{\partial \log L_n}{\partial b}$ and $\frac{\partial \log L_n}{\partial \sigma}$ in (3.1) and (3.2). First,

$$
\begin{aligned}
& \frac{\partial \log L_n}{\partial b} = \frac{1}{P_\theta(Y_n = y_n)} \frac{\partial P_\theta(Y_n = y_n)}{\partial b} \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \frac{\partial \big(\int \prod_{s=1}^{S} \prod_{j=1}^{J} p_{nsj}^{y_{nsj}} f_\sigma(u_n) \, \mathrm{d}u_n \big)}{\partial b} \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int \frac{\partial \big(\prod_{s=1}^{S} \prod_{j=1}^{J} p_{nsj}^{y_{nsj}} \big)}{\partial b} f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_{s=1}^{S} \prod_{j=1}^{J} p_{nsj}^{y_{nsj}}) (\sum_s \sum_j y_{nsj} \frac{\partial p_{nsj}}{\partial b}) f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) \Big(\sum_s \sum_j y_{nsj} (x_{nsj} - \sum_i p_{nsi} x_{nsi}) \Big) f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) \Big(\sum_s \sum_j y_{nsj} x_{nsj} - \sum_s \sum_j y_{nsj} (\sum_i p_{nsi} x_{nsi}) \Big) f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) \Big(\sum_s \sum_j y_{nsj} x_{nsj} - \sum_s \sum_j p_{nsj} x_{nsj} \Big) f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) (X_n' y_n - X_n' p_n) f_\sigma(u_n) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} X_n' \Big(P_\theta(Y_n = y_n) y_n - \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) p_n f_\sigma(u_n) \, \mathrm{d}u_n \Big) \\
= \; & X_n' \Big(y_n - \frac{1}{P_\theta(Y_n = y_n)} \int (\prod_s \prod_j p_{nsj}^{y_{nsj}}) p_n f_\sigma(u_n) \, \mathrm{d}u_n \Big) \\
= \; & X_n' \big(y_n - E_{u_n}(p_n | y_n) \big),
\end{aligned}
$$

where p_n is defined in (3.1). For the second expression that is to be evaluated,

$$
\begin{aligned}
& \frac{\partial \log P_\theta(Y_n = y_n)}{\partial \sigma} = \frac{1}{P_\theta(Y_n = y_n)} \frac{\partial P_\theta(Y_n = y_n)}{\partial \sigma} \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \frac{\partial \big(\int \prod_{s=1}^{S} \prod_{j=1}^{J} p_{nsj}^{y_{nsj}} f_\sigma(u_n) \, \mathrm{d}u_n \big)}{\partial \sigma} \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int \big(\prod_{s=1}^{S} \prod_{j=1}^{J} p_{nsj}^{y_{nsj}} \big) \frac{\partial f_\sigma(u_n)}{\sigma} \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int \big(\prod_{s} \prod_{j} p_{nsj}^{y_{nsj}} \big) \Big((2\pi)^{-k/2} (-\frac{1}{2}) |\Sigma|^{-3/2} \exp(-\frac{1}{2} u_n' \Sigma^{-1} u_n) \frac{\partial |\Sigma|}{\partial \sigma} \\
& + (2\pi)^{-k/2} |\Sigma|^{-1/2} \exp(-\frac{1}{2} u_n' \Sigma^{-1} u_n) (-\frac{1}{2}) \frac{\partial (u_n' \Sigma^{-1} u_n)}{\partial \sigma} \Big) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int \big(\prod_{s} \prod_{j} p_{nsj}^{y_{nsj}} \big) \Big(-\frac{1}{2} f_\sigma(u_n) \big[|\Sigma|^{-1} \frac{\partial |\Sigma|}{\partial \sigma} + \frac{\partial (u_n' \Sigma^{-1} u_n)}{\partial \sigma} \big] \Big) \, \mathrm{d}u_n \\
= \; & \frac{1}{P_\theta(Y_n = y_n)} \int \big(\prod_{s} \prod_{j} p_{nsj}^{y_{nsj}} \big) \Big(f_\sigma(u_n) \big[-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k})' + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3})' \big] \Big) \, \mathrm{d}u_n \\
= \; & -(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k})' + E_{u_n} \big((\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3})' | y_n \big),
\end{aligned}
$$

where u_{ni} is the ith element of u_n, $1 \le i \le k$.

Using these partial derivatives, we can now get the expressions for $E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})' \Big)$, $E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \sigma})' \Big)$ and $E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma})' \Big)$.

First, for $E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big)$ we have

$$\begin{aligned}
& E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) \\
= \;& E_{Y_n}\Big(X_n'\big[Y_n - E_{u_n}(p_n|Y_n)\big]\big[Y_n' - E_{u_n}(p_n'|Y_n)\big]X_n\Big) \\
= \;& X_n' E_{Y_n}\Big(\big[Y_n - E_{u_n}(p_n|Y_n)\big]\big[Y_n' - E_{u_n}(p_n'|Y_n)\big]\Big)X_n \\
= \;& X_n'\Big(E_{Y_n}(Y_nY_n') - E_{Y_n}\big[E_{u_n}(p_n|Y_n)Y_n'\big] - E_{Y_n}\big[Y_nE_{u_n}(p_n'|Y_n)\big] \\
& \qquad + E_{Y_n}\big[E_{u_n}(p_n|Y_n)E_{u_n}(p_n'|Y_n)\big]\Big)X_n.
\end{aligned}$$

Theses expressions are now evaluated separately.

$$\begin{aligned}
E_{Y_n}(Y_nY_n') &= E_{u_n}(E_{Y_n}(Y_nY_n'|u_n)) \\
&= E_{u_n}\big[E_{Y_n}(\begin{pmatrix} Y_{n1} \\ Y_{n2} \\ \vdots \\ Y_{nS} \end{pmatrix}\begin{pmatrix} Y_{n1}' & Y_{n2}' & \dots & Y_{nS}' \end{pmatrix}|u_n)\big] \\
&= E_{u_n}\begin{pmatrix} E_{Y_n}(Y_{n1}Y_{n1}'|u_n) & E_{Y_n}(Y_{n1}Y_{n2}'|u_n) & \dots & E_{Y_n}(Y_{n1}Y_{nS}'|u_n) \\ E_{Y_n}(Y_{n2}Y_{n1}'|u_n) & E_{Y_n}(Y_{n2}Y_{n2}'|u_n) & \dots & E_{Y_n}(Y_{n2}Y_{nS}'|u_n) \\ \dots & \dots & \dots & \dots \\ E_{Y_n}(Y_{nS}Y_{n1}'|u_n) & E_{Y_n}(Y_{nS}Y_{n2}'|u_n) & \dots & E_{Y_n}(Y_{nS}Y_{nS}'|u_n) \end{pmatrix} \\
&= E_{u_n}\begin{pmatrix} \mathrm{diag}(p_{n1}) & p_{n1}p_{n2}' & \dots & p_{n1}p_{nS}' \\ p_{n2}p_{n1}' & \mathrm{diag}(p_{n2}) & \dots & p_{n2}p_{nS}' \\ \dots & \dots & \dots & \dots \\ p_{nS}p_{n1}' & p_{nS}p_{n2}' & \dots & \mathrm{diag}(p_{nS}) \end{pmatrix},
\end{aligned}$$

where p_{ns} is defined after (3.1). Next,

$$\begin{aligned}
& E_{Y_n}\big[E_{u_n}(p_n|Y_n)Y_n'\big] \\
= & \sum_{y_n}\Big[\Big(\int p_n\frac{\prod_s\prod_j p_{nsj}^{y_{nsj}}}{P_\theta(Y_n=y_n)}f_\sigma(u_n)\,\mathrm{d}u_n\Big)y_n'P_\theta(Y_n=y_n)\Big] \\
= & \int p_n\sum_{y_n}\Big[\prod_s\prod_j p_{nsj}^{y_{nsj}}y_n'\Big]f_\sigma(u_n)\,\mathrm{d}u_n \\
= & \int p_np_n'f_\sigma(u_n)\,\mathrm{d}u_n \\
= & E_{u_n}(p_np_n').
\end{aligned}$$

Let $\Delta_n=\mathrm{diag}(\Delta_{ns})$ with $\Delta_{ns}=\mathrm{diag}(p_{ns})-p_{ns}p_{ns}'$. Then

$$E_{u_n}(\Delta_n)=E_{Y_n}(y_ny_n')-E_{Y_n}\big[E_{u_n}(p_n|y_n)y_n'\big].$$

Hence, we have

$$\begin{aligned}
& E_{Y_n}\bigg(\Big(\frac{\partial logL_n}{\partial b}\Big)\Big(\frac{\partial logL_n}{\partial b}\Big)'\bigg) \\
= & X_n'\Big(E_{u_n}(\Delta_n)-E_{u_n}(p_np_n')+E_{Y_n}\big[E_{u_n}(p_n|Y_n)E_{u_n}(p_n'|Y_n)\big]\Big)X_n.
\end{aligned}$$

Second, $E_{Y_n}\Big(\Big(\frac{\partial \log L_n}{\partial b}\Big)\Big(\frac{\partial \log L_n}{\partial \sigma}\Big)'\Big)$ can be written as

$$
\begin{aligned}
& E_{Y_n}\Big(\Big(\frac{\partial \log L_n}{\partial b}\Big)\Big(\frac{\partial \log L_n}{\partial \sigma}\Big)'\Big) \\
= \;& E_{Y_n}\Big(X_n'\big[Y_n - E_{u_n}(p_n|Y_n)\big]\Big[-\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) + E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\Big]\Big) \\
= \;& -X_n' E_{Y_n}\big[Y_n - E_{u_n}(p_n|Y_n)\big]\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) \\
& \quad + X_n' E_{Y_n}\Big(\big[Y_n - E_{u_n}(p_n|Y_n)\big] E\Big[\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big]\Big) \\
= \;& X_n' E_{Y_n}\Big(Y_n E_{u_n}\Big[\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big]\Big) \\
& \quad - X_n' E_{Y_n}\Big(E_{u_n}(p_n|Y_n) E_{u_n}\Big[\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big]\Big).
\end{aligned}
$$

To evaluate the first of these, note that

$$
\begin{aligned}
& E_{Y_n}\Big(Y_n E_{u_n}\Big[\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big]\Big) \\
= \;& \sum_{y_n}\Big(y_n\Big[\int\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)\frac{\prod_s\prod_j p_{nsj}^{y_{nsj}}}{P_\theta(Y_n=y_n)} f_\sigma(u_n)\,\mathrm{d}u_n\Big] P_\theta(Y_n=y_n)\Big) \\
= \;& \int\Big[\sum_{y_n}\Big(\prod_s\prod_j p_{nsj}^{y_{nsj}}\Big) y_n\Big]\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big) f_\sigma(u_n)\,\mathrm{d}u_n \\
= \;& \int p_n\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big) f_\sigma(u_n)\,\mathrm{d}u_n \\
= \;& E_{u_n}\Big[p_n\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)\Big].
\end{aligned}
$$

Hence, we have

$$
\begin{aligned}
& E_{Y_n}\left(\Big(\frac{\partial \log L_n}{\partial b}\Big)\Big(\frac{\partial \log L_n}{\partial \sigma}\Big)'\right) \\
= \;& X_n'\left(E_{u_n}\Big[p_n\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)\Big] - E_{Y_n}\Big[E_{u_n}(p_n|Y_n)E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\Big]\right).
\end{aligned}
$$

Last, $E_{Y_n}\left(\Big(\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma}\Big)'\right)$ can be written as

$$
\begin{aligned}
& E_{Y_n}\left(\Big(\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma}\Big)'\right) \\
= \; & E_{Y_n}\Bigg(\Big[-\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)' + E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)'|Y_n\Big)\Big] \\
& \qquad \cdot \Big[-\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) + E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\Big]\Bigg) \\
= \; & \Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) - \Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)' E_{Y_n}\left[E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\right] \\
& - E_{Y_n}\left[E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)'|Y_n\Big)\right]\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) \\
& + E_{Y_n}\left[E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)'|Y_n\Big)E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\right] \\
= \; & \Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) - \Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) - \Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) \\
& + E_{Y_n}\left[E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)'|Y_n\Big)E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\right] \\
= \; & -\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k}\Big) \\
& + E_{Y_n}\left[E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)'|Y_n\Big)E_{u_n}\Big(\Big(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}\Big)|Y_n\Big)\right].
\end{aligned}
$$

Information Matrix for General Σ

For general Σ, not necessarily a diagonal matrix, a normal random vector β_n can be written as $\beta_n = b + u_n$, where $u_n \sim N_k(0, \Sigma = \Gamma\Gamma')$ with Γ a lower triangular matrix. Let $\gamma = vec(\Gamma')$, the information matrix for $\theta = (b', \gamma')'$ is

$$I(\theta|X) = \sum_{n=1}^{N} \begin{pmatrix} E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \gamma})'\Big) \\ E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \gamma})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \gamma})(\frac{\partial \log L_n}{\partial \gamma})'\Big) \end{pmatrix},$$

where $L_n = P_\theta(Y_n = y_n)$ is the likelihood function for respondent n and is given by

$$\begin{aligned} & P_\theta(Y_n = y_n) \\ = & \int P_b(Y_n = y_n|u_n) f_\gamma(u_n)\, \mathrm{d}u_n \\ = & \int \prod_{s=1}^{S}\prod_{j=1}^{J} \left(\frac{\exp(x'_{nsj}(b+u_n))}{\sum_{i=1}^{J}\exp(x'_{nsi}(b+u_n))}\right)^{y_{nsj}} (2\pi)^{-k/2}|\Sigma|^{-1/2}\exp(-\frac{1}{2}u'_n\Sigma^{-1}u_n)\, \mathrm{d}u_n. \end{aligned}$$

It can be shown that $\frac{\partial \log L_n}{\partial b}$ has the same expression as before. For $\frac{\partial \log L_n}{\partial \gamma}$, the derivation is the same as for $\frac{\partial \log L_n}{\partial \sigma}$ except that we cannot simplify the following expression further,

$$\frac{\partial \log L_n}{\partial \gamma} = \frac{1}{P_\theta(Y_n = y_n)} \int \Big(\prod_{s=1}^{S}\prod_{j=1}^{J} p_{nsj}^{y_{nsj}}\Big)\Big(-\frac{1}{2} f_\gamma(u_n)\big[|\Sigma|^{-1}\frac{\partial|\Sigma|}{\partial\gamma} + \frac{\partial(u'_n\Sigma^{-1}u_n)}{\partial\gamma}\big]\Big)\, \mathrm{d}u_n.$$

3.6.2 Laplace Approximation

In (3.6), we have

$$\begin{aligned} E_{u_1}(p_{1sj}|y_1^i) &= \frac{\int p_{1sj} P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1) \mathrm{d}u_1}{\int P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1) \mathrm{d}u_1} \\ &= \frac{\int \exp\big[\log p_{1sj} + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big] \mathrm{d}u_1}{\int \exp\big[\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big] \mathrm{d}u_1} \\ &\approx \Big(\frac{|H_{sj}|}{|H|}\Big)^{1/2} \frac{p_{1sj} P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1)|_{u_1=\hat{u}_{1_{sj}}}}{P_b(Y_1 = y_1^i|u_1) f_\sigma(u_1)|_{u_1=\hat{u}_1}}, \end{aligned}$$

where $\hat{u}_{1_{sj}}$ maximizes $\log p_{1sj} + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$, $\hat{u}_1$ maximizes $\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)$,

$$\begin{aligned} H_{sj} &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log p_{1sj} + \log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_{1_{sj}}} \\ &= -\Big(\big[\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log p_{1sj} + \frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log P_b(Y_1 = y_1^i|u_1) + \frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_{1_{sj}}} \\ &= -(-X_{1s}'\Delta_{1s}X_{1s} - X_1'\Delta_1 X_1 - \Sigma^{-1})^{-1}|_{u_1=\hat{u}_{1_{sj}}}, \end{aligned}$$

and

$$\begin{aligned} H &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log P_b(Y_1 = y_1^i|u_1) + \log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_1} \\ &= -\Big(\big[\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log P_b(Y_1 = y_1^i|u_1) + \frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_1} \\ &= -(-X_1'\Delta_1 X_1 - \Sigma^{-1})^{-1}|_{u_1=\hat{u}_1}. \end{aligned}$$

The validity of these expressions follows because

$$\begin{aligned}
\frac{\partial}{\partial u_1}(\log p_{1sj}) &= \frac{1}{p_{1sj}}(\frac{\partial p_{1sj}}{\partial u_1}) \\
&= \frac{1}{p_{1sj}}\frac{\partial}{\partial u_1}(\frac{\exp(x'_{1sj}(b+u_1))}{\sum_{i=1}^{J}\exp(x'_{1si}(b+u_1))}) \\
&= \frac{1}{p_{1sj}}(p_{1sj}x_{1sj} - p_{1sj}\sum_i p_{1si}x_{1si}) \\
&= x_{1sj} - \sum_i p_{1si}x_{1si},
\end{aligned}$$

so that

$$\begin{aligned}
\frac{\partial}{\partial u_1}\frac{\partial}{\partial u'_1}\log p_{1sj} &= \frac{\partial}{\partial u_1}(x'_{1sj} - \sum_i p_{1si}x'_{1si}) \\
&= -\sum_i(\frac{\partial}{\partial u_1}p_{1si})x'_{1si} \\
&= -\sum_i(p_{1si}x_{1si} - p_{1si}\sum_l p_{1sl}x_{1sl})x'_{1si} \\
&= -\sum_i p_{1si}x_{1si}x'_{1si} + (\sum_i p_{1si}x_{1si})(\sum_i p_{1si}x'_{1si}) \\
&= -X'_{1s}\mathrm{diag}(p_{1s})X_{1s} + X'_{1s}p_{1s}p'_{1s}X_{1s} \\
&= -X'_{1s}\Delta_{1s}X_{1s}.
\end{aligned}$$

Further,

$$
\begin{aligned}
\frac{\partial}{\partial u_1}\log P_b(Y_1 = y_1^i|u_1) &= \frac{1}{P_b(Y_1 = y_1^i|u_1)}(\frac{\partial P_b(Y_1 = y_1^i|u_1)}{\partial u_1}) \\
&= \frac{1}{P_b(Y_1 = y_1^i|u_1)}\frac{\partial}{\partial u_1}(\prod_s\prod_j p_{1sj}^{y_{1sj}^i}) \\
&= \frac{1}{P_b(Y_1 = y_1^i|u_1)}(\prod_s\prod_j p_{1sj}^{y_{1sj}^i})(\sum_s\sum_j \frac{y_{1sj}^i}{p_{1sj}}\frac{\partial p_{1sj}}{\partial u_1}) \\
&= \sum_s\sum_j \frac{y_{1sj}^i}{p_{1sj}}(p_{1sj}x_{1sj} - p_{1sj}\sum_k p_{1sk}x_{1sk}) \\
&= \sum_s\sum_j (y_{1sj}^i - p_{1sj})x_{1sj},
\end{aligned}
$$

so that

$$
\begin{aligned}
\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log P_b(Y_1 = y_1^i|u_1) &= \frac{\partial}{\partial u_1}(\sum_s\sum_j (y_{1sj}^i - p_{1sj})x_{1sj}') \\
&= -\sum_s\sum_j (\frac{\partial}{\partial u_1}p_{1sj})x_{1sj}' \\
&= -\sum_s\sum_j (p_{1sj}x_{1sj} - p_{1sj}\sum_k p_{1sk}x_{1sk})x_{1sj}' \\
&= -\sum_s (X_{1s}'diag(p_{1s})X_{1s} - X_{1s}'p_{1s}p_{1s}'X_{1s}) \\
&= -X_1'\Delta_1 X_1.
\end{aligned}
$$

In (3.7), we have

$$\begin{aligned} E\Big(\frac{u_{1j}^2}{\sigma_j^3}|y_1^i\Big) &= E\Big(\frac{u_{1j}^2}{\sigma_j^3}+c|y_1^i\Big)-c \\ &= \frac{\int \frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3} P_b(Y_1=y_1^i|u_1) f_\sigma(u_1)\mathrm{d}u_1}{\int P_b(Y_1=y_1^i|u_1) f_\sigma(u_1)\mathrm{d}u_1}-c \\ &= \frac{\int \exp\big[\log(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3})+\log P_b(Y_1=y_1^i|u_1)+\log f_\sigma(u_1)\big]\mathrm{d}u_1}{\int \exp\big[\log P_b(Y_1=y_1^i|u_1)+\log f_\sigma(u_1)\big]\mathrm{d}u_1}-c \\ &\approx \Big(\frac{|H_j|}{|H|}\Big)^{1/2}\frac{\frac{{u_{1_j}}^2+c\sigma_j^3}{\sigma_j^3}P_b(Y_1=y_1^i|u_1)\log f_\sigma(u_1)|_{u_1=\hat{u}_{1_j}}}{P_b(Y_1=y_1^i|u_1)f_\sigma(u_1)|_{u_1=\hat{u}_1}}-c, \end{aligned}$$

where $\hat{u}_{1_j}$ maximizes $\log(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3})+\log P_b(Y_1=y_1^i|u_1)+\log f_\sigma(u_1)$ and

$$\begin{aligned} H_j &= -\Big(\frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\big[\log(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3})+\log P_b(Y_1=y_1^i|u_1)+\log f_\sigma(u_1)\big]\Big)^{-1}|_{u_1=\hat{u}_{1_j}} \\ &= -\left(\frac{2(c\sigma_j^3-u_{1j}^2)}{(u_{1j}^2+c\sigma_j^3)^2}e_je_j'-X_1'\Delta_1X_1-\Sigma^{-1}\right)^{-1}|_{u_1=\hat{u}_{1_j}}. \end{aligned}$$

The validity of this expression follows because

$$\frac{\partial}{\partial u_1}\log\Big(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3}\Big) = \frac{2u_{1j}e_j}{u_{1j}^2+c\sigma_j^3},$$

and

$$\begin{aligned}
& \frac{\partial}{\partial u_1}\frac{\partial}{\partial u_1'}\log\Big(\frac{u_{1j}^2+c\sigma_j^3}{\sigma_j^3}\Big) \\
= & \frac{\partial}{\partial u_1}\Big(\frac{2u_{1j}e_j'}{u_{1j}^2+c\sigma_j^3}\Big) \\
= & \frac{2e_je_j'}{u_{1j}^2+c\sigma_j^3}-\frac{2u_{1j}e_j}{(u_{1j}^2+c\sigma_j^3)^2}(2u_{1j}e_j') \\
= & \frac{2e_je_j'(u_{1j}^2+c\sigma_j^3)-4{u_{1j}}^2e_je_j'}{(u_{1j}^2+c\sigma_j^3)^2} \\
= & \frac{2(c\sigma_j^3-u_{1j}^2)}{(u_{1j}^2+c\sigma_j^3)^2}e_je_j'.
\end{aligned}$$

3.7 References

Arora, N., and Huber, J. (2001). Improving parameter estimates and model prediction by aggregate customization in choice experiments. *Journal of Consumer Research*, **28**(2), 273−283.

Atkinson, A. C., Donev, A. N., and Tobias, R. D. (2007). *Optimum experimental designs, with SAS* . New York: Oxford University Press.

Bhat, C. (1998). Accommodating variations in responsiveness to level-of-service variables in travel mode choice models. *Transportation Research A*, **32**, 455−507.

Bhat, C. (2000). Incorporating observed and unobserved heterogeneity in urban work mode choice modeling. *Transportation Science*, **34**, 228−238.

Bliemer, M. C., and Rose, J. M. (2010). Construction of experimental designs for mixed logit models allowing for correlation across choice observations. *Transportation Research Part B: Methodological*, **44**(6), 720−734.

Booth, J. G., and Hobert, J. P. (1999). Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 265−285.

Breslow, N. E., and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**(421), 9−25.

Brownstone, D. and K. Train (1999). Forecasting new product penetration with flexible substitution patterns. *Journal of Econometrics*, **89**, 109−129.

Chernoff, H. (1953). Locally optimal designs for estimating parameters. *The Annals of Mathematical Statistics*, 586−602.

Erdem, T. (1996). A dynamic analysis of market structure based on panel data. *Marketing Science*, **15**, 359−378.

Hensher, D. A., Rose, J. M., and Greene, W. H. (2005). *Applied choice analysis: a primer*. Cambridge University Press.

McCulloch, C. E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association*, **92**(437), 162−170.

McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. *In: Zarembka P (ed), Frontiers in Econometrics*. Academic Press, New York, pp. 105−142.

McFadden, D. and K. Train (2000). Mixed MNL models of discrete response. *Journal of Applied Econometrics*, **15**, 447−470.

Moerbeek, M., and Maas, C. J. (2005). Optimal experimental designs for multilevel logistic models with two binary predictors. *Communications in Statistics− Theory and Methods*, **34**(5), 1151−1167.

Peter E. Rossi, Greg M. Allenby, and Robert McCulloch (2006). *Bayesian Statistics and Marketing*. John Wiley and Sons, Ltd.

Revelt, D. and K.Train (1998). Mixed logit with repeated choices: households' choices of appliance efficiency level. *Review of Economics and Statistics*, **80**(4), 647−657.

Sándor, Z., and Wedel, M. (2002). Profile construction in experimental choice designs for mixed logit models. *Marketing Science*, **21**(4), 455−475.

Tierney, L., and Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, **81**(393), 82−86.

Tierney, L., Kass, R. E., and Kadane, J. B. (1989). Fully exponential Laplace approximations to expectations and variances of nonpositive functions. *Journal of the American Statistical Association*, **84**(407), 710−716.

Toubia, O., Hauser, J. R., and Simester, D. I. (2004). Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, **41**(1), 116−131.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

Wand, M. P. (2007). Fisher information for generalized linear mixed models. *Journal of Multivariate Analysis*, **98**(7), 1412−1416.

Waite, T.W. and Woods, D.C. (2014) Designs for generalized linear models with random block effects via information matrix approximations. Southampton, GB, Southampton Statistical Sciences Research Institute, 21pp. (Southampton Statistical Sciences Research Institute Methodology Working Papers, M12/01).

Yu, J., Goos, P., and Vandebroek, M. (2011). Individually adapted sequential Bayesian conjoint-choice designs in the presence of consumer heterogeneity. *International Journal of Research in Marketing*, **28**(4), 378−388.

Chapter 4

Optimal Designs for the Panel Mixed Logit Model

4.1 Introduction

Discrete choice experiments are usually used in marketing, health care, transportation, etc., to understand how people make their choices among several alternatives. In a typical choice experiment, respondents are asked to choose their most preferred alternative based on characteristics of the alternatives presented to them, e.g., making a choice among laptops with different specifications. Discrete choice models can be used to explain the choices the respondents made based on influential factors such as characteristics of the alternatives and socioeconomic status of the respondents.

For example, Veldwijk et al. (2014) use a discrete choice experiment to study the barriers and facilitators of lifestyle programs for Type 2 Diabetes Mellitus patients. The five attributes, each with three levels, are menu schedule (a flexible schedule, a regular schedule or an elaborate schedule), physical activity (PA) schedule (a flexible schedule, a regular schedule or an elaborate schedule), consultation structure (individual, in groups of 5 or 10 other patients), expected outcome

(0, 5 or 10 kilograms weight loss), and out-of-pocket costs (€75, €150 or €225 per year). The design problem is to choose alternatives to present to the respondents. The characteristics of the alternatives are varied purposefully to learn about how people make their trade-offs.

Optimal design theory can be used to solve the design problem. In the literature, optimal designs have been developed for several discrete choice models. Huber and Zwerina (1996), Sándor and Wedel (2001) and Street and Burgess (2007) consider designs for the multinomial logit model. Sándor and Wedel (2002) and Yu et al. (2009) are concerned with designs for the cross-sectional mixed logit model. Rose (2010) and Yu et al. (2011) deal with designs for the panel mixed logit model.

We are interested in finding optimal designs for a panel mixed logit model, which represents choice behaviors better than the multinomial logit model or the cross-sectional mixed logit model. First, a panel mixed logit model can account for the heterogeneity in preferences by introducing random effects to the multinomial logit model, while the multinomial logit model assumes that respondents' preferences are all the same. Second, the panel mixed logit model can account for the correlation in the responses from the same respondent, while the cross-sectional mixed logit model assumes independence. In a choice experiment, each respondent usually answers several questions. Responses from a respondent are likely to be correlated, since each respondent adopts similar decision rules in different questions. However, finding efficient designs for the panel mixed logit model is difficult, since comparing designs using criteria based on the information matrix

requires a large amount of computation. For the panel mixed logit model, both the likelihood function and the information matrix can not be written in closed-form expressions and can only be evaluated numerically.

The panel mixed logit model is a special case of a generalized linear mixed model. For generalized linear mixed models, it is also difficult to find efficient designs using criteria based on the information matrix which corresponds to the maximum likelihood (ML) method. Design criteria based on other analysis methods such as penalized quasi-likelihood (PQL), marginalized quasi-likelihood (MQL) and generalized estimating equations (GEE) are considered by (Moerbeek and Mass (2005), Tekle et al. (2008), Niaparast (2009), Ogungbenro and Aarons (2011), Niaparast and Schwabe (2013), Waite et al. (2012)). The resulting design criteria are easier to compute. From the literature of generalized linear mixed models, we propose a new criterion based on the method of simulated moments (MSM) (Jiang and Zhang (2001). In the aforementioned work, except for Ogungbenro and Aarons (2011), the interest lies only in making inference about the mean of the random effects and treating variance of random effects as nuisance parameters. For the panel mixed logit model, the variance parameters are also important, since we are interested in knowing the distribution of respondents' preferences. Hence, we also consider the uncertainty of variance parameters in our new criterion.

In Sections 4.2-4.4, we discuss how to derive variance-covariance matrices of the estimators from maximum likelihood (ML), penalized quasi-likelihood (PQL) and marginalized quasi-likelihood (MQL) methods, and method of simulated moments (MSM), respectively, for the panel mixed logit model. In Section 4.5, we use a

computer search to find optimal designs with the four types of design criteria and compare the results from the searches. In Section 4.6, we consider finding designs for larger choice designs while varying the number of random attributes. In Section 4.7, we revisit the motivating example and the Chapter concludes with a discussion in Section 4.8.

4.2 Maximum Likelihood Method

In Chapter 3, we focus on the information matrix, the inverse of which is the asymptotic variance-covariance matrix of the maximum likelihood estimator. The variance-covariance matrix gives uncertainty of an estimator, so it is often used to formulate design criteria. First, we give a brief summary of how the information matrix is derived.

In a choice experiment, a respondent is often shown several alternatives and asked to choose the one they like the most. The respondent will be asked to respond to several of these hypothetical questions, each of which consists of a different set of alternatives. The set of alternatives in each question is called a choice set. Suppose an alternative can be represented by q attributes and the overall preference of alternative j in choice set s is related to the linear function $x'_{nsj}\beta_n$, where x_{nsj} is the k-vector which contains the coded levels of the q attributes of alternative j in choice set s for respondent n and β_n is the k-vector of random effects for respondent n. Given β_n, the probability of respondent n choosing alternative j in

choice set s is given by

$$P(Y_{nsj} = 1|\beta_n) = \frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)},$$

where J is the number of alternatives in the choice set. The responses for respondent n are given by $Y_n = (Y_{n11}, Y_{n12}, \ldots, Y_{nSJ})'$, where $Y_{nsj} = 1$ if respondent n chooses alternative j in choice set s and $Y_{nsj} = 0$ otherwise. The likelihood for respondent n given β_n is modeled by

$$P(Y_n = y_n|\beta_n) = \prod_{s=1}^{S}\prod_{j=1}^{J}\left(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)}\right)^{y_{nsj}}.$$

The likelihood function for the panel mixed logit model is given by

$$\begin{aligned} L(\theta|Y = y) &= \prod_{n=1}^{N}\int P(y_n|\beta_n)f_\theta(\beta_n)\,\mathrm{d}\beta_n \\ &= \prod_{n=1}^{N}\int\prod_{s=1}^{S}\prod_{j=1}^{J}\left(\frac{\exp(x'_{nsj}\beta_n)}{\sum_{i=1}^{J}\exp(x'_{nsi}\beta_n)}\right)^{y_{nsj}} f_\theta(\beta_n)\,\mathrm{d}\beta_n, \end{aligned} \quad (4.1)$$

where $y = (y'_1, y'_2, \ldots, y'_N)'$ is the vector of observed choices from the N respondents and the density function of β_n is $f_\theta(\beta_n)$ with unknown parameter θ. Usually, β_n's are assumed to be independent and follow a multivariate normal distribution $N_k(b, \Sigma)$ with $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_k^2)$. Then β_n can be written as $\beta_n = b + u_n$, where $u_n = (u_{n1}, \ldots, u_{nk})'$ is from $N_k(0, \Sigma)$. The length of u_n is the same as the length of β_n when all the attributes are random. We will assume all element in β_n are random when we derive the formula. Without loss of generality, k can be

adjusted to k' when the first k' elements of β_n are assumed to be random. Denoting $\sigma = (\sigma_1, \ldots, \sigma_k)'$, the unknown parameter vector is $\theta = (b', \sigma')'$. The information matrix of θ can be written as

$$I(\theta|X) = \sum_{n=1}^{N} \begin{pmatrix} E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \\ E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial b})'\Big) & E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \end{pmatrix},$$

where

$$L_n = \int P(y_n|\beta_n = b + u_n)(2\pi)^{-k/2}|\Sigma|^{-1/2}\exp(-\frac{1}{2}u_n'\Sigma^{-1}u_n)\,\mathrm{d}u_n.$$

It can be shown that

$$\begin{aligned}
& E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial b})'\Big) \\
= \; & X_n'\Big(E_{u_n}(\Delta_n) - E_{u_n}(p_n p_n') + E_{Y_n}\big[E_{u_n}(p_n|Y_n)E_{u_n}(p_n'|Y_n)\big]\Big)X_n, \\
& E_{Y_n}\Big((\frac{\partial \log L_n}{\partial b})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \\
= \; & X_n'\Big(E_{u_n}\big[p_n h_\sigma(u_n)\big] - E_{Y_n}\big[E_{u_n}(p_n|Y_n)E_{u_n}\big(h_\sigma(u_n)|Y_n\big)\big]\Big), \\
& E_{Y_n}\Big((\frac{\partial \log L_n}{\partial \sigma})(\frac{\partial \log L_n}{\partial \sigma})'\Big) \\
= \; & -\Big(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}\Big)'\Big(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}\Big) + E_{Y_n}\Big[E_{u_n}\big(h_\sigma(u_n)|Y_n\big)E_{u_n}\big(h_\sigma'(u_n)|Y_n\big)\Big], \qquad (4.2)
\end{aligned}$$

where $p_n = (p_{n1}', \ldots, p_{nS}')'$ with $p_{ns} = (p_{ns1}, \ldots, p_{nsJ})'$ and $p_{nsj} = \frac{\exp(x_{nsj}'\beta_n)}{\sum_{i=1}^{J}\exp(x_{nsi}'\beta_n)}$, $\Delta_n = \mathrm{diag}(\Delta_{n1}, \ldots, \Delta_{nS})$ with $\Delta_{ns} = \mathrm{diag}(p_{ns}) - p_{ns}p_{ns}'$ and $h_\sigma(u_n) = \Big(\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}\Big)'$.

The maximum likelihood estimator $\hat{\theta}$, at which $L(\theta|Y = y)$ is maximized, has asymptotic variance-covariance matrix given by $Var(\hat{\theta}) = I(\theta|X)^{-1}$.

In Chapter 3, we propose three methods to approximate the information matrix, among which Laplace approximation is the fastest method. However, it is still not fast enough if we want to use it in a coordinate exchange algorithm for finding efficient designs. For the analysis of generalized linear mixed models, there are computationally more efficient methods for estimating unknown parameters. In the next two sections, we will derive variance-covariance matrices of estimators from these methods.

4.3 PQL and MQL Applied to Panel Mixed Logit Model

For the analysis of generalized linear mixed models, Breslow and Clayton (1993) propose PQL and MQL, which are approximations to the maximum likelihood method but computationally more efficient. Similarly, for the design problem, the variance-covariance matrices of these estimators can be utilized as alternatives to the variance-covariance matrix of the maximum likelihood estimator.

First, we apply PQL to the panel mixed logit model. The likelihood function

in (4.1) can be written as

$$\begin{aligned} L &= \int \prod_{n=1}^{N} p(y_n|\beta_n = b + u_n) f_\sigma(u_n)\,\mathrm{d}u \\ &\propto |G|^{-1/2} \int \exp\left(\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{j=1}^{J} y_{nsj}\Big[x'_{nsj}(b+u_n) \right. \\ &\left. - \log\big(\sum_{i=1}^{J}\exp[x'_{nsi}(b+u_n)]\big)\Big] - \frac{1}{2}u'G^{-1}u \right)\,\mathrm{d}u, \end{aligned}$$

where $u = (u'_1, \ldots, u'_N)' \sim N(0, G)$ with $G = diag(\underbrace{\Sigma, \ldots, \Sigma}_{N})$. The log-likelihood function can be written as

$$l = c - \frac{1}{2}\log|G| + \log\int \exp(-q(u))\,\mathrm{d}u,$$

where c does not depend on the unknown parameters and $q(u)$ is given by

$$-\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{j=1}^{J} y_{nsj}\Big[x'_{nsj}(b+u_n) - \log\big(\sum_{i=1}^{J}\exp[x'_{nsi}(b+u_n)]\big)\Big] + \frac{1}{2}u'G^{-1}u.$$

Applying Laplace's method to l gives

$$l \approx c - \frac{1}{2}\log|G| - \frac{1}{2}\log|q''(\tilde{u})| - q(\tilde{u}),$$

where $\tilde{u}$ maximizes $q(u)$ and the matrix of second derivatives of $q(u)$, $q''(u)$, is a

block diagonal matrix with the nth diagonal block given by

$$q_n''(u) = \sum_{s=1}^{S} x_{ns}' \Delta_{ns} x_{ns} + \Sigma^{-1}, n = 1, \ldots, N.$$

In Breslow and Clayton (1993), with some simplification to the above approximation of l, PQL is given by

$$-\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{j=1}^{J} y_{nsj}\Big[x_{nsj}'(b+u_n) - \log\big(\sum_{i=1}^{J}\exp[x_{nsi}'(b+u_n)]\big)\Big] + \frac{1}{2}u'G^{-1}u. \quad (4.3)$$

An estimate of b and u can be obtained by jointly maximizing (4.3) with respect to b and u. In Breslow and Clayton (1993), the approximate variance-covariance matrix for $\hat{b}^{PQL}$ is given by

$$Var(\hat{b}^{PQL}) = (\sum_{i=1}^{N} X_n' V_n^{-1} X_n)^{-1},$$

where $V_n = \Delta_n^{-1}|_{u_n=\tilde{u}_n} + X_n \Sigma X_n'$. An estimate of θ can be obtained by maximizing REML of the corresponding linear mixed effects model for y. The (j,k) element of the information matrix for σ from the REML version of approximation is given by

$$\frac{1}{2} tr(P\frac{\partial V}{\partial \sigma_j} P \frac{\partial V}{\partial \sigma_k}),$$

where $P = V^{-1} - V^{-1}X(X'V^{-1}X)^{-1}X'V^{-1}$ and $V = \text{diag}(V_1, \ldots, V_N)$. When N is large, the results from REML and ML are similar, but REML requires a lot more computation. Hence, we use the ML version of the information matrix of σ

and denote it as $I_{\sigma\sigma}^{PQL}$, of which the (j,k) element is given by

$$\frac{1}{2}tr(V^{-1}\frac{\partial V}{\partial \sigma_j}V^{-1}\frac{\partial V}{\partial \sigma_k}).$$

Finally, the variance-covariance of $\hat{\theta}^{PQL}$ is given by

$$Var(\hat{\theta}^{PQL}) = \begin{pmatrix} Var(\hat{b}^{PQL}) & 0 \\ 0 & (I_{\sigma\sigma}^{PQL})^{-1} \end{pmatrix}. \tag{4.4}$$

In PQL, $\tilde{u}$, which maximizes $q(u)$, needs to be found numerically. MQL can simplify PQL further by using $\tilde{u} = 0$ and the corresponding variance-covariance matrix for $\hat{\theta}^{MQL}$ can be obtained by evaluating (4.4) at $\tilde{u} = 0$.

PQL and MQL aim to get estimates of unknown parameters faster, but they are known to produce biased estimates in some cases. Our goal is to use the variance covariance matrices from PQL and MQL as alternatives to the variance-covariance matrix from ML in the design criteria. The design criteria based on these approximations have not been compared to the design criteria based on ML for finding efficient designs before. We will compare these design criteria in their abilities for finding efficient designs in Section 4.6.

4.4 Method of Simulated Moments (MSM) Applied to Panel Mixed Logit Model

For generalized linear mixed models, Jiang and Zhang (2001) propose to use the method of simulated moments for analysis. Instead of using the likelihood function, they use the method of moments to construct estimating equations.

If β_n is written as $b + L\nu_n$ where $L = diag(\sigma)$ is the Cholesky decomposition of $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_k^2)$ and $\nu_n \sim N_k(0, I_k)$, then the likelihood function for respondent n can be written as

$$\begin{aligned} L_n &= \int P(y_n|\beta_n = b + L\nu_n)\frac{1}{(2\pi)^{k/2}}\exp(-\frac{\nu_n'\nu_n}{2})\mathrm{d}\nu_n \\ &= \int \exp\Big(\sum_{s=1}^{S}\sum_{j=1}^{J}\sum_{l=1}^{k} y_{nsj}x_{nsjl}(b_l + \sigma_l\nu_{nl}) \\ &\qquad -\sum_{s=1}^{S}\log\big(\sum_{j=1}^{J}\exp(x_{nsi}'(b + L\nu_n))\big) - \frac{k}{2}\log(2\pi) - \frac{\nu_n'\nu_n}{2}\Big)\mathrm{d}\nu_n. \end{aligned}$$

A set of sufficient statistics for θ is given by

$$\begin{aligned} &\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsj1}y_{nsj}, 1 \le n \le N, \\ &\vdots \\ &\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjk}y_{nsj}, 1 \le n \le N. \end{aligned}$$

A set of estimating equations using method of moments is given by

$$\begin{aligned}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj} &= \sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}E(y_{nsj}), 1 \le l \le k, \\ \sum_{n=1}^{N}(\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj})^2 &= \sum_{n=1}^{N} E(\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj})^2, 1 \le l \le k.\end{aligned}$$

Let the lth column of X_n be $x_n^l = (x_{n11l}, x_{n12l}, \ldots, x_{nSJl})'$ and $s_{nl} = x_n^{l\,\prime} y_n$. The above equations can be written as

$$S = u(\theta),$$

where $S = (S_1', S_2')'$ and $u(\theta) = E(S_1', S_2')'$ with $S_1 = (\sum_{n=1}^{N} s_{n1}, \ldots, \sum_{n=1}^{N} s_{nk})'$ and $S_2 = (\sum_{n=1}^{N} s_{n1}^2, \ldots, \sum_{n=1}^{N} s_{nk}^2)'$.

Let the estimate from the above estimating equations be $\tilde{\theta}$. A second step estimator $\hat{\theta}^{MSM}$ can be obtained by solving $\tilde{B}S = \tilde{B}u(\theta)$, where $\tilde{B} = U'V^{-1}|_{\theta=\tilde{\theta}}$ with $U = \partial u(\theta)/\partial\theta'$ and $V = Var(S)$. Jiang and Zhang (2001) show that, under suitable conditions, $\hat{\theta}^{MSM}$ has the same asymptotic variance-covariance matrix as the estimate from solving $BS = Bu(\theta)$, which is

$$\begin{aligned} & Var(\hat{\theta}^{MSM}) \\ = \; & (U'V^{-1}U)^{-1} \\ = \; & \left[\begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial (S_2)}{\partial \sigma'} \end{pmatrix}' \begin{pmatrix} V(S_1) & Cov(S_1, S_2) \\ Cov(S_2, S_1) & V(S_2) \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial E(S_2)}{\partial \sigma'} \end{pmatrix} \right]^{-1}, \end{aligned} \tag{4.5}$$

where $U = \frac{\partial u(\theta)}{\partial \theta'}$.

Next, we will derive the expression for U. The details are given in the appendix. $\frac{\partial E(s_{nl})}{\partial \theta'}$ is given by

$$\begin{aligned}
\sum_{n=1}^{N} x_n^{l\,\prime} \frac{\partial E(y_n)}{\partial b'} &= \sum_{n=1}^{N} x_n^{l\,\prime} E(\Delta_n) X_n, \\
\sum_{n=1}^{N} x_n^{l\,\prime} \frac{\partial E(y_n)}{\partial \sigma'} &= \sum_{n=1}^{N} x_n^{l\,\prime} E[p_n(-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}))].
\end{aligned}$$

Further, we can write out $\sum_{n=1}^{N} \frac{\partial E(s_{nl}^2)}{\partial \theta'} = \sum_{n=1}^{N} (x_n^{l\,\prime} \otimes x_n^{l\,\prime}) \frac{\partial E(y_n \otimes y_n)}{\partial \theta'}$, and show that the $(((s-1)J + i - 1)(SJ) + (s'-1)J + j)$th row of $\frac{\partial E(y_n \otimes y_n)}{\partial \theta'}$, given by $(\frac{\partial E(y_{nsi} y_{ns'j})}{\partial b'}, \frac{\partial E(y_{nsi} y_{ns'j})}{\partial \sigma'})$, is

$$\frac{\partial E(y_{nsi} y_{ns'j})}{\partial b'} = \begin{cases} \Big(-E(p_{ns1}p_{nsj}), \ldots, E(p_{nsj} - p_{nsj}^2), \ldots, -E(p_{nsJ}p_{nsj}) \Big) X_{ns} & \text{if } s = s',\ i = j \\ 0 & \text{if } s = s',\ i \neq j \\ E\Big(p_{nsi} p_{ns'j} \big[(x'_{nsi} - \sum_{k=1}^{J} p_{nsk} x'_{nsk}) + (x'_{ns'j} - \sum_{k=1}^{J} p_{ns'k} x'_{ns'k}) \big] \Big) & \text{if } s \neq s' \end{cases}.$$

and

$$\frac{\partial E(y_{nsi} y_{ns'j})}{\partial \sigma'} = \begin{cases} E\Big(p_{nsj} \big[-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}) \big] \Big)\, \mathrm{d}u_n & \text{if } s = s',\ i = j \\ 0 & \text{if } s = s',\ i \neq j \\ E\Big(p_{nsj} p_{ns'j} \big[-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}) \big] \Big) & \text{if } s \neq s' \end{cases}.$$

To evaluate $Var(\hat{\theta}^{MSM})$ in (4.5), we can use Monte Carlo methods on U and V independently, because U only involves expectations with respect to u and V only involves moments of Y. Therefore, the evaluations should be much faster than those of the information matrix in (4.2), where expectations with respect to u and Y are nested.

4.5 Searching for Optimal Designs

In the previous sections, we derive variance-covariance matrices from the penalized quasi-likelihood method (PQL), the marginal quasi-likelihood method (MQL), and the method of simulated moments (MSM). In this section, we will find optimal designs based on design criteria using these variance-covariance matrices.

The coordinate exchange algorithm is usually used to search for optimal designs for discrete choice models. The algorithm starts with a random design and uses exchanges to find better designs under a given design criterion. The exchange is done for one attribute of one alternative at a time by replacing the current level of the attribute with the other possible levels of the attribute. The exchange will be kept if there is an improvement in the design criterion after the exchange. The algorithm will stop if no change is kept after going through the entire design−from the first attribute of the first alternative in the first choice set to the last attribute of the last alternative in the last choice set. This is a run of the coordinate exchange algorithm, which yields a design that cannot be improved anymore by exchanges. Usually, many runs are used to avoid local optima. The final result is the best

design from all runs. The designs from the coordinate exchange algorithm are not guaranteed to be optimal designs, so they are often referred to as efficient designs.

First, we consider a case where there are 4 attributes each with 3 levels. We assume that the coefficients for the first two attributes are random and those for the other two attributes are fixed. Effects type coding is used for the attribute levels. The random coefficients are assumed to be independent and distributed as $N(b_i, \sigma_i^2),\ i = 1, \ldots, 4$. Let the mean of the whole coefficient vector be $b = (b_1, b_2, \ldots, b_8)'$ and the variance vector be $\sigma = (\sigma_1, \ldots, \sigma_4)'$. The values of parameters are varied in terms of the response accuracy and the respondent heterogeneity. The parameter vector b is given by $a \cdot (1, 0, 1, 0, 1, 0, 1, 0)'$ and the parameter vector σ is given by $\sqrt{c \cdot a} \cdot (1, 1, 1, 1)'$, where $a = 3$ in the case of high response accuracy and $a = 1/2$ in the case of low response accuracy; $c = 3$ in the case of high respondent heterogeneity and $c = 1/2$ in the case of low respondent heterogeneity. The 4 combinations of a and c values are all used in the computer search.

We will first consider an experiment with 9 choice sets of size 2, which is denoted as $3^4/2/9$.

For given parameter values and an approximation method, the coordinate exchange algorithm is used to find a locally A-optimal design and a locally D-optimal design respectively. A-optimality and D-optimality are used with the Laplace approximation of the information matrix and variance-covariance matrices from PQL, MQL and MSM. Hence, for each combination of parameter values, the coordinate

exchange algorithm is implemented 8 times− for A- and D-optimality each with the 4 methods. After searches are finished, we want to compare efficient designs obtained from different methods. Since the panel mixed logit model is usually analyzed with the maximum likelihood method, we are interested in comparing the designs for this method. Hence, the efficient designs are evaluated again with the criterion based on the variance-covariance matrix of the maximum likelihood method. To get an accurate comparison of designs, the information matrix is approximated using the importance sampling method in Chapter 3 with all possible values of Y and 10^6 as the sample size for u. The results are reported in Table 4.1, where the given time is for one run of the coordinate exchange algorithm.

The results in Table 4.1 are used to compare the 4 methods. First, a good method should enable the coordinate exchange algorithm to find a good design. Second, the time to complete a search is determined mainly by the time needed to evaluate the design criterion here. For the two cases in the upper half of Table 4.1, the best design for A-optimality is found by MSM and the best design for D-optimality is found by the Laplace approximation. While the A- or D-optimality values of the designs found by the two methods are similar, the time for MSM is about 1/10 of the time for the Laplace approximation. Although the searches with PQL and MQL run faster, the designs found are considerably less efficient than the best designs except with D-optimality in the second case, where all methods yield similar results. For the two cases in the lower half of the table, the A- or D-optimality values of designs found by the four criteria are all similar. The search with MQL is the fastest, while the time for PQL might also be acceptable in

Table 4.1: Results for $3^4/2/9$

	hh				hl			
	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	18.18	7.77	115m24s	157m29s	5.87	2.18	144m	163m32s
MSM	16.93	7.86	11m47s	14m27s	5.83	2.21	11m52s	13m41s
PQL	26.30	8.79	1m56s	2m6s	7.47	2.25	3m9s	2m29s
MQL	37.39	10.31	3.4s	3.6s	7.27	2.33	3.4s	3.8s
	lh				ll			
	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	2.37	1.65	115m26s	203m47s	0.85	0.69	161m38s	190m30s
MSM	2.38	1.66	11m1s	10m58s	0.87	0.69	14m59s	15m3s
PQL	2.36	1.68	2m45s	1m43s	0.84	0.69	4m48s	3m28s
MQL	2.45	1.68	3.8s	3.6s	0.86	0.70	3.5s	3.5s

Note: hh represents high accuracy and high heterogeneity($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$), hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$), lh represents low accuracy and high heterogeneity($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$) and ll represents low accuracy and low heterogeneity ($b = (0.5, 0, 0.5, 0)'$ and $\sigma = (0.5, 0.5, 0.5, 0.5)'$).

practice and PQL yields better designs than MQL in almost all cases. For the second case in the upper half and the first case in the lower half, the values for b are different while the values for σ are the same, we can see that the performance of the methods not only depends on the magnitude of the variance but also on the magnitude of the mean.

Next, we consider an experiment with 5 choice sets of size 4 and denote it as $3^4/4/5$. We repeat the procedures in the previous experiment with only the first 2 cases of the parameter values, since results from the 4 methods are similar in the other two cases. The results are given in Table 4.2. The results are similar

Table 4.2: Results for $3^4/4/5$

	hh				hl			
	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	15.60	6.71	190m35s	234m34s	4.63	2.30	266m7s	241m46s
MSM	14.67	6.68	24m17s	29m18s	4.49	2.27	26m48s	21m53s
PQL	20.61	7.49	2m50s	2m42s	4.70	2.35	2m58s	3m47s
MQL	25.83	∞	4.3s	3.8s	16.70	2.34	3.1s	3.1s

Note: hh represents high accuracy and high heterogeneity($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$) and hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$).

to those in Table 4.1, except that PQL performs better for A-optimality in the case of high response accuracy and low respondent heterogeneity and PQL is now comparable with the Laplace approximation and MSM. The improvement in PQL might be due to the increase in the size of the design, i.e., there are $(4-1)\cdot 5 = 15$ independent rows in the design matrix for $3^5/4/5$ compared to $(2-1)\cdot 9 = 9$ for $3^5/2/9$.

With MSM, the coordinate exchange algorithm still takes too long to run, e.g., it takes approximately 1000×10 mins in $3^5/2/10$. Next, we try to reduce the number of runs in the coordinate exchange algorithm. With m (< 1000) runs, what is the probability of getting a design that is at least 90% efficient as the best design from 1000 runs, when the designs are compared using the maximum likelihood method. One way is to assess this is by running the coordinate exchange algorithm with m starting designs repeatedly and calculating the proportion of times of getting a design that is at least 90% efficient, but it will be computationally difficult.

Another way is by repeatedly sampling m runs from the 1000 runs we have already completed and calculating the proportion of times of getting a design at least 90% efficient. The simulation is carried out as follows. From running a computer search with 1000 runs, we obtain 1000 designs from the coordinate exchange algorithm and the corresponding A- or D-optimality values for a given method. If m runs are taken randomly from the 1000 runs, the m designs can be ordered according to the m A- or D-optimality values obtained from the computer search. The best design is evaluated using the criterion based on the maximum likelihood method. This process is repeated many times, so the probability of getting a design that is above 90% efficient can be calculated. For the first case of the parameter values, the best A-optimality value we have is 16.77 and the best D-optimality value is 7.68. A design that is at least 90% efficient in A-optimality will have a A-optimality smaller than $16.77/0.9 = 18.65$ and a design that is at least 90% efficient in D-optimality will have a D-optimality smaller than $7.68/0.9 = 8.53$. Table 4.3 reports the the values of m with which the probability of getting a 90% or 80% efficient design is over 0.9 for the first two cases of the parameter values. The m values in some of the cells are much smaller than 1000. Hence, it is possible to reduce the number of the runs in these cases. For some cases with PQL and MQL, the results with 1000 starting designs can not pass the 90% or 80% efficiency standard, so NA values are entered for these cases. From the m values for different methods, we conclude MSM is the best method to use, which is consistent with the previous result.

Table 4.3: Reduced number of runs for $3^4/2/9$

	hh				hl			
	A		D		A		D	
	90%	80%	90%	80%	90%	80%	90%	80%
Laplace	895	9	3	1	595	5	9	1
MSM	182	12	6	1	90	3	8	1
PQL	NA	NA	NA	190	NA	NA	55	3
MQL	NA	NA	NA	NA	NA	NA	900	30

Note: hh represents high accuracy and high heterogeneity($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$) and hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$). NA are put in cases where even with 1000 runs the results are not at least 90% or 80% efficient.

4.6 Second Look at the Search for Optimal Designs

First, we will study the robustness of locally optimal designs against misspecification of parameter values. The rows of Table 4.4 give the assumed parameter values for finding the locally optimal designs. The columns give the true values of the parameters. There are 8 designs in the table−one locally A-optimal design and one locally D-optimal design in each row for the assumed parameter values. The designs are from previous results−in the first two rows, we use the designs from MSM; in the last two rows, we use the designs from PQL. The designs are compared with the best designs found previously with the true values, so the efficiencies (shown in parentheses) are always less than 1. As expected, we can see that the further the true parameter values move away from the assumed ones,

Table 4.4: Robustness

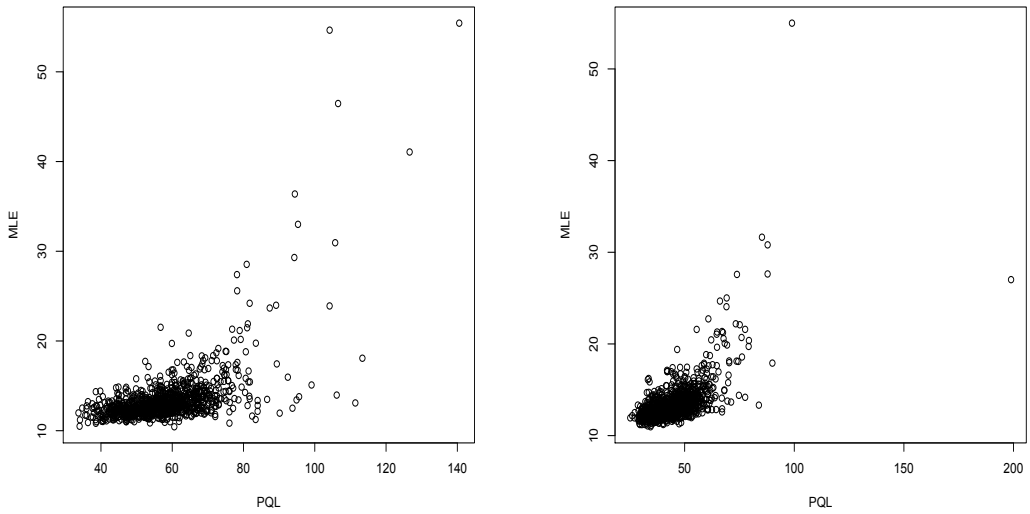
	hh		hl		lh		ll	
design	A	D	A	D	A	D	A	D
hh	16.91	7.86	7.33 (.80)	2.87 (.77)	3.74 (.63)	2.14 (.79)	2.27 (.37)	1.22 (.56)
hl	19.50 (.87)	9.16 (.86)	5.86	2.21	3.50 (.68)	1.98 (.85)	1.70 (.5)	0.81 (.84)
lh	75.11 (.23)	9.52 (.83)	25.62 (.23)	3.90 (.57)	2.37	1.69	0.88 (.97)	0.74 (.92)
ll	59.96 (.28)	9.92 (.79)	13.69 (.43)	3.05 (.72)	2.42 (.98)	1.75 (.97)	0.85	0.68

Note: hh represents high accuracy and high heterogeneity($b = (3, 0, 3, 0)'$ and $\sigma = (3, 3, 3, 3)'$) and hl represents high accuracy and low heterogeneity ($b = (3, 0, 3, 0)'$ and $\sigma = (\sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5}, \sqrt{1.5})'$). In the parentheses are the efficiency of the design compared to the best design.

the locally optimal designs become less efficient; however, the loss of efficiency is not more than 20%, except for a few cases within the range of parameter values studied.

Second, we conduct the search for larger experiments. We consider a case with 5 attributes with 3 levels and vary the number of random attributes from 2 to 5. The two sets of parameter values with high accuracy are used, because the differences between the 4 methods for the other values are small in the previous results. The design problems for $3^5/2/10$ and $3^5/4/5$ are discussed.

Figure 4.1 gives A-optimalities of the 1000 good designs from the coordinate exchange algorithm using criterion based on PQL. The x-axis gives A-optimality using ML and the y-axis gives A-optimality using PQL. The best design for ML can not be found if PQL, i.e. values from the y-axis, is used to compare designs. Based on the observation, we propose a new algorithm to improve on PQL. After each run of the coordinate exchange algorithm using PQL, the design found in the



(a) 10 choice sets of size 2

(b) 5 choice sets of size 4

Figure 4.1: The 1000 designs from the coordinate exchange algorithm using criterion based on PQL and A-optimality for design problem with 5 attributes with 3 levels and out of which 3 attributes are random.

run is evaluated with the criterion based on ML. After the 1000 runs, the design criterion values from ML are used to find the best design. With this change to the exchange algorithm, good designs could be found with one more evaluation of the design criterion based on ML at the end. The new algorithm is denoted as new PQL in Table 4.5 to Table 4.8.

If two attributes are random, the results in Table 4.5 are similar to the previous ones. MQL gives the worst results in all cases considered except one. In all cases considered for A-optimality, the designs from criteria based on the Laplace approx-

Table 4.5: 3^5 with 2 random attributes

	hh				hl			
$3^5/2/10$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	19.76	5.64	106m52s	92m26s	5.91	1.84	100m37s	110m33s
MSM	17.75	5.91	7m6s	16m17s	6.06	1.99	5m26s	7m21s
PQL	20.53	7.00	40s	38s	6.70	1.96	1m9s	49s
New PQL	18.47	5.73	6m41s	6m34s	5.74	1.95	7m14s	6m14s
MQL	28.84	7.83	6s	5s	6.56	2.22	6s	6s
$3^5/4/5$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	14.30	5.56	98m10s	88m21s	4.92	2.07	98m00s	113m5s
MSM	15.40	5.89	5m39s	5m11s	5.05	2.14	6m5s	5m55s
PQL	20.77	5.81	53s	49s	5.45	2.18	1m14s	52s
New PQL	16.26	5.81	6m55s	6m30s	4.73	2.18	7m29s	7m30s
MQL	46.97	9.72	7s	6s	39.27	4.00	6s	5s

imation and MSM are similar and they are better than designs from the criterion based on PQL by a small margin. Considering design efficiency and time, the criteria based on MSM and PQL are competitive when A-optimality is used−MSM is better in terms of design efficiency and PQL is better in terms of time. For D-optimality, the designs from criteria based on the Laplace approximation, MSM and PQL are similar. For $3^5/2/10$ with high accuracy and high heterogeneity, there is some efficiency loss associated with the criterion based on PQL. PQL is the best method for D-optimality except for smaller designs with parameter values at high accuracy and high heterogeneity. In this case, MSM might be a competitive option because it yields better designs in acceptable time.

The results for 3 random attributes are listed in Table 4.6. The results are similar to those in Table 4.5 except for the results for A-optimality with high accuracy and

high heterogeneity in the smaller design $3^4/2/10$, where criteria based on Laplace approximation and PQL does not work well. Table 4.7 contains the results for 4 random attributes. The results are similar to those in Table 4.5 and Table 4.6 except for high accuracy and high heterogeneity and A-optimality, where MSM works best while the Laplace approximation and PQL do not work in both cases. Table 4.8 contains results for 5 random attributes. The results are similar to the previous ones, except the Laplace approximation and PQL fail in one more case, i.e., the case $2^{10}/2/10$ with high accuracy and low heterogeneity and A-optimality.

To summarize, MSM works well for both A- and D-optimality in all the cases and PQL works well with D-optimality in almost all the cases. The results show their performance under different specifications of the experiments, such as the size of the design, the values of the parameters and the number of random attributes. PQL runs about 10 times faster than MSM. When they yield similar results, PQL is preferred. The Laplace approximation is much slower than the other methods and does not work for A-optimality in some situations.

In new PQL, we use all possible Y and 10^5 as the sample size for U, which adds about 6 minutes to the time for PQL. The time for MSM and the time for the new algorithm are similar. Also, we can control the speed by using appropriate sample sizes for Y and U in the evaluation of criterion based on ML. The results are given in Table 4.5 to Table 4.8. We see improvements in almost all cases, especially in cases PQL preform worse than MSM. The results of MSM and the new algorithm are similar in almost all cases except for high accuracy and high heterogeneity with A-optimality in Table 4.8, where results for the new algorithm

Table 4.6: 3^5 with 3 random attributes

	hh				hl			
$3^5/2/10$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	88.86	10.61	118m16s	157m10s	9.32	2.71	129m46s	158m40s
MSM	31.67	11.01	5m48s	7m5s	8.94	2.62	5m26s	6m22s
PQL	60.58	11.89	58s	53s	11.15	2.84	1m31s	1m4s
New PQL	33.68	10.71	7m55s	7m33s	8.79	2.77	7m38s	7m30s
MQL	75.47	13.84	6s	4s	11.17	2.97	6s	5s
$3^5/4/5$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	26.13	11.77	141m55s	170m16s	6.92	3.02	174m47s	143m18s
MSM	23.75	11.66	5m25s	5m40s	6.85	3.23	6m35s	5m25s
PQL	34.45	13.27	49s	56s	8.12	3.12	1m28s	1m14s
New PQL	25.19	11.17	7m40s	7m14s	6.56	3.12	8m14s	7m16s
MQL	46.11	16.48	7s	4s	21.50	5.62	6s	4s

Table 4.7: 3^5 with 4 random attributes

	hh				hl			
$3^5/2/10$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	76.76	15.56	185m12s	164m34s	13.25	3.60	160m42s	189m43s
MSM	67.01	16.74	6m50s	5m56s	12.06	3.61	12m7s	11m20s
PQL	195.47	16.64	1m11s	1m10s	20.52	3.72	2m8s	2m3s
New PQL	50.53	16.38	7m49s	7m48s	13.00	3.72	8m59s	8m40s
MQL	104.14	19.37	7s	5s	20.07	4.05	6m	4m
$3^5/4/5$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	84.27	16.99	239m3s	224m17s	9.08	3.97	242m31s	285m3s
MSM	41.41	18.26	6m11s	4m30s	8.91	4.09	8m32s	5m24s
PQL	49.87	19.01	1m19s	59s	10.71	4.00	1m21s	1m35s
New PQL	36.63	17.84	7m52s	7m44s	9.03	4.00	8m9s	8m28s
MQL	67.77	20.41	6s	6s	15.16	5.79	5s	4s

Table 4.8: 3^5 with 5 random attributes

	hh				hl			
$3^5/2/10$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	384.41	22.27	288m25s	264m31s	149.45	4.56	345m50s	363m2s
MSM	135.08	22.67	17m2s	19m43s	16.33	4.78	18m38s	20m4s
PQL	362.85	23.00	1m36s	1m33s	40.99	4.88	3m11s	2m40s
New PQL	211.5	23.00	76m46s	82m54s	20.61	4.81	84m11s	78m51s
MQL	484.10	25.89	5s	8s	44.82	5.06	6s	7s
$3^5/4/5$	A	D	time (A)	time (D)	A	D	time (A)	time (D)
Laplace	305	23.89	306m4s	373m11s	13.23	5.16	366m47s	332m58s
MSM	71.66	25.42	17m31s	14m19s	12.46	5.12	17m41s	17m
PQL	160.11	27.94	1m27s	1m41s	17.62	5.41	2m34s	2m6s
New PQL	118.45	24.51	78m41s	78m53s	13.24	5.25	80m9s	107m29s
MQL	181.52	∞	6s	5s	20.72	8.45	5s	4s

are much worse than MSM.

In new PQL, we use all possible Y and 10^5 as the sample size for U, which adds about 6 minutes to the time for PQL. The time for MSM and the time for the new algorithm are similar. Also, we can control the speed by using appropriate sample sizes for Y and U in the evaluation of the criterion based on ML. The results are given in Table 4.5 to Table 4.8. We see improvements in almost all cases, especially when PQL preform worse than MSM. The results of MSM and the new algorithm are similar in almost all cases except for high accuracy and high heterogeneity with A-optimality in Table 4.8, where results for the new algorithm are much worse than MSM.

4.7 Revisiting the Example

We will use the lifestyle program example to show how an efficient design from the computer search looks like. Suppose we want to find a design for $3^5/2/10$ with 4 random attributes and the parameters are given by $b = (3, 0, 3, 0, 3, 0, 3, 0, 3, 0)'$ and $\sigma = 3 \cdot 1_8$, Table 4.9 gives an efficient design from the coordinate exchange algorithm using A-optimality with MSM.

4.8 Discussion and Conclusion

In this Chapter, we propose three alternatives to the design criterion based on the maximum likelihood (ML) method for the panel mixed logit model, which is difficult to use in practice because of the amount of computation required. The three alternatives are derived from penalized quasi-likelihood (PQL), marginal quasi-likelihood (MQL) and method of simulated moments (MSM). The alternatives based on PQL and MQL have closed form expressions, while the alternative from MSM simplifies the criterion based on ML to contain expectations with respect to the response and the random effects independently. However, PQL and MQL are approximate analysis methods which produce biased estimates, while the estimate from MSM are consistent but could be less efficient than from ML. We use a computer search to see whether an efficient design can be found in a computer search using the alternatives. From the results, all three methods show significant improvement in time with MQL being the fastest followed by PQL and MSM. The designs from MSM are as good as those from the Laplace approximation and

Table 4.9: $3^5/2/10$ with 4 random attributes when $b = (3, 0, 3, 0, 3, 0, 3, 0, 3, 0)'$ and $\sigma = 3 \cdot 1_8$.

Choice Set	Measure Schedule	Physical Activity (PA) Schedule	Construction Structure	Expected Weight Loss	Out-of-pocket Costs
1	Regular	Elaborate	10 other patients	10 kilograms	€225
	Regular	Elaborate	5 other patients	10 kilograms	€225
2	Elaborate	Regular	Individual	5 kilograms	€75
	Elaborate	Elaborate	5 other patients	10 kilograms	€75
3	Flexible	Flexible	Individual	0 kilograms	€150
	Flexible	Flexible	5 other patients	0 kilograms	€225
4	Flexible	Regular	5 other patients	10 kilograms	€75
	Elaborate	Regular	5 other patients	0 kilograms	€75
5	Regular	Regular	5 other patients	5 kilograms	€225
	Regular	Flexible	5 other patients	5 kilograms	€225
6	Regular	Regular	5 other patients	5 kilograms	€150
	Flexible	Regular	Individual	0 kilograms	€225
7	Elaborate	Regular	10 other patients	0 kilograms	€75
	Flexible	Regular	10 other patients	10 kilograms	€225
8	Elaborate	Elaborate	Individual	0 kilograms	€150
	Flexible	Elaborate	Individual	10 kilograms	€225
9	Elaborate	Regular	5 other patients	10 kilograms	€225
	Regular	Elaborate	Individual	10 kilograms	€75
10	Regular	Regular	5 other patients	10 kilograms	€225
	Elaborate	Elaborate	5 other patients	5 kilograms	€225

even outperform those from Laplace approximation for the case with more random effects. Designs from MSM are the best for both A- and D-optimality and all parameter values considered. The designs from PQL are the best for D-optimality and cases where heterogeneity is not too large. One run of the coordinate exchange algorithm using MSM takes 5-20 minutes in all the cases considered, so 1000 runs still take a significant amount of time. However, the 1000 runs can be executed in parallel. One run of the coordinate exchange algorithm using PQL takes less than 4 minutes in all cases considered, so PQL is viable option to use in practice for D-optimality and cases where heterogeneity is not too large.

4.9 Appendix

4.9.1 Method of Moments Applied to Panel Mixed Logit Model

If β_n is written as $b + Lu_n$ where $L = diag(\sigma)$ is the Cholesky decomposition of $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_k^2)$ and $u_n \sim N_k(0, I_k)$, the likelihood function for respondent n

can be written as

$$
\begin{aligned}
L_n &= \int \prod_{s=1}^{S}\prod_{j=1}^{J}\left(\frac{\exp\left(x'_{nsj}(b+Lu_n)\right)}{\sum_{i=1}^{J}\exp\left(x'_{nsi}(b+Lu_n)\right)}\right)^{y_{nsj}} \frac{1}{(2\pi)^{k/2}}\exp(-\frac{u'_n u_n}{2})\mathrm{d}u_n \\
&= \int \exp\left(\sum_{s=1}^{S}\sum_{j=1}^{J} y_{nsj}x'_{nsj}(b+Lu_n) - \sum_{s=1}^{S}\log\Big(\sum_{j=1}^{J}\exp\left(x'_{nsi}(b+Lu_n)\right)\Big)\right. \\
&\qquad \left. -\frac{k}{2}\log(2\pi) - \frac{u'_n u_n}{2}\right)\mathrm{d}u_n \\
&= \int \exp\left(\sum_{s=1}^{S}\sum_{j=1}^{J}\sum_{l=1}^{k} y_{nsj}x_{nsjl}(b_l+\sigma_l u_{nl}) - \sum_{s=1}^{S}\log\left(\sum_{j=1}^{J}\exp\left(x'_{nsi}(b+Lu_n)\right)\right)\right. \\
&\qquad \left. -\frac{k}{2}\log(2\pi) - \frac{u'_n u_n}{2}\right)\mathrm{d}u_n.
\end{aligned}
$$

A set of sufficient statistics for θ is given by $\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj}, 1 \le l \le k$. Then, a set of estimating equations using method of moments can be formulated as

$$
\begin{aligned}
\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj} &= \sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}E(y_{nsj}), 1 \le l \le k, \\
(\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj})^2 &= E(\sum_{s=1}^{S}\sum_{j=1}^{J} x_{nsjl}y_{nsj})^2, 1 \le l \le k. \qquad (4.6)
\end{aligned}
$$

Let the lth column of X_n be $x_n^l = (x_{n11l}, x_{n12l}, \ldots, x_{nSJl})'$ and $s_l = x_n^{l\prime} y$. Equations (4.6) can be written as

$$
\begin{aligned}
x_n^{l\prime} y_n &= x_n^{l\prime} E(y_n) = x_n^{l\prime} E(p_n) \quad , \ 1 \le l \le k \\
(x_n^{l\prime} y_n)^2 &= E(x_n^{l\prime} y_n)^2 = (x_n^{l\prime} \otimes x_n^{l\prime}) E(y_n \otimes y_n) \quad , \ 1 \le l \le k
\end{aligned}
.
$$

Let $S = (S_1', S_2')'$ with $S_1 = (s_1, \ldots, s_k)'$ and $S_2 = (s_1^2, \ldots, s_k^2)'$ and $u(\theta) = E(S_1', S_2')'$, equations 4.6 can be written as $S = u(\theta)$. For estimating equations of the form $BS = Bu(\theta)$, the optimal B is $U'V^{-1}$, where $U = \partial u/\partial\theta'$ and $V = Var(S)$ is the variance-covariance matrix of S. The variance-covariance matrix of the resulting estimator is

$$
\begin{aligned}
Var(\hat{\theta}) &= (U'V^{-1}U)^{-1} \\
&= \left[\begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial (S_2)}{\partial \sigma'} \end{pmatrix}' \begin{pmatrix} V(S_1) & Cov(S_1, S_2) \\ Cov(S_2, S_1) & V(S_2) \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial E(S_2)}{\partial \sigma;} \end{pmatrix} \right]^{-1},
\end{aligned}
$$

where $U = \frac{\partial (E(S_1', S_2')')}{\partial \theta'}$.

We will first derive $\frac{\partial E(s_l)}{\partial \theta'} = x_n^{l\,\prime} \frac{\partial E(p_n)}{\partial \theta'}$, $1 \leq l \leq k$. The $(sJ+j)$th row of $\partial E(p_n)/\partial\theta'$ is given by $(\partial E(p_{nsj})/\partial b', \partial E(p_{nsj})/\partial \sigma')$, which can be written as

$$
\begin{aligned}
\frac{\partial E(p_{nsj})}{\partial b'} &= \int \frac{\partial p_{nsj}}{\partial b'} f_\sigma(u_n)\, \mathrm{d}u_n = E\Big(p_{nsj}(x'_{nsj} - \sum_{i=1}^{J} p_{nsi} x'_{nsi})\Big) \\
&= \Big(- E(p_{ns1}p_{nsj}), \ldots, E(p_{nsj} - p_{nsj}^2), \ldots, -E(p_{nsJ}p_{nsj})\Big) X_{ns}, \\
\frac{\partial E(p_{nsj})}{\partial \sigma'} &= \int p_{nsj} f_\sigma(u_n)(-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3})) \\
&= E(p_{nsj}(-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}))) \, \mathrm{d}u_n.
\end{aligned}
$$

So $\frac{\partial E(s_l)}{\partial \theta'}$ is given by

$$\begin{aligned} {x_n^l}' \frac{\partial E(y_n)}{\partial b'} &= {x_n^l}' E(\Delta_n) X_n, \\ {x_n^l}' \frac{\partial E(y_n)}{\partial \sigma'} &= {x_n^l}' E[p_n(-(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}) + (\frac{u_{n1}^2}{\sigma_1^3}, \ldots, \frac{u_{nk}^2}{\sigma_k^3}))]. \end{aligned}$$

Next, we derive $\frac{\partial E(s_l^2)}{\partial \theta'}$. The $((s-1)*J+i-1)*(SJ)+(s'-1)*J+j)$th element of $E(y_n \otimes y_n)$ is given by

$$E(y_{nsi} y_{ns'j}) = \begin{cases} E(p_{nsj}) & , \quad s = s', \ i = j \\ 0 & , \quad s = s', \ i \neq j \\ E(p_{nsi} p_{ns'j}) & , \quad s \neq s' \end{cases} .$$

It can be shown that

$$\begin{aligned} \frac{\partial E(p_{nsi} p_{ns'j})}{\partial b'} &= \int \frac{\partial (p_{nsi} p_{ns'j})}{\partial b'} f_\sigma(u_n) \, \mathrm{d}u_n \\ &= \int (\frac{\partial p_{nsi}}{\partial b'} p_{ns'j} + p_{nsi} \frac{\partial p_{ns'j}}{\partial b'}) f_\sigma(u_n) \, \mathrm{d}u_n \\ &= E(p_{nsi} p_{ns'j} (x'_{nsi} - \sum_{k=1}^{J} p_{nsk} x'_{nsk}) + p_{nsi} p_{ns'j} (x'_{ns'j} - \sum_{k=1}^{J} p_{ns'k} x'_{ns'k})) \\ &= E(p_{nsi} p_{ns'j} ((x'_{nsi} - \sum_{k=1}^{J} p_{nsk} x'_{nsk}) + (x'_{ns'j} - \sum_{k=1}^{J} p_{ns'k} x'_{ns'k}))) \end{aligned}$$

and

$$\begin{aligned}\frac{\partial E(p_{nsi}p_{ns'j})}{\partial \sigma'} &= \int p_{nsj}p_{ns'j}f_{\sigma}(u_n)(-(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k})+(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}))\,\mathrm{d}u_n \\ &= E(p_{nsj}p_{ns'j}(-(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k})+(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}))).\end{aligned}$$

So we can write out $\frac{\partial E(s_l^2)}{\partial \theta'} = ({x_n^l}' \otimes {x_n^l}')\frac{\partial E(y_n \otimes y_n)}{\partial \theta'}$, where the $((s-1)*J+i-1)*(SJ)+(s'-1)*J+j)$th row of $\frac{\partial E(y_n \otimes y_n)}{\partial \theta'}$, given by $(\frac{\partial E(y_{nsi}y_{ns'j})}{\partial b'}, \frac{\partial E(y_{nsi}y_{ns'j})}{\partial \sigma'})$, is

$$\frac{\partial E(y_{nsi}y_{ns'j})}{\partial b'}$$
$$= \begin{cases} \Big(-E(p_{ns1}p_{nsj}),\ldots,E(p_{nsj}-p_{nsj}^2),\ldots,-E(p_{nsJ}p_{nsj})\Big)X_{ns} & , \quad s=s',\ i=j \\ 0 & , \quad s=s',\ i\neq j \\ E(p_{nsi}p_{ns'j}((x'_{nsi}-\sum_{k=1}^{J}p_{nsk}x'_{nsk})+(x'_{ns'j}-\sum_{k=1}^{J}p_{ns'k}x'_{ns'k}))) & , \quad s\neq s' \end{cases};$$

and

$$\frac{\partial E(y_{nsi}y_{ns'j})}{\partial \sigma'}$$
$$= \begin{cases} E(p_{nsj}(-(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k})+(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3})))\,\mathrm{d}u_n & , \quad s=s',\ i=j \\ 0 & , \quad s=s',\ i\neq j \\ E(p_{nsj}p_{ns'j}(-(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_k})+(\frac{u_{n1}^2}{\sigma_1^3},\ldots,\frac{u_{nk}^2}{\sigma_k^3}))) & , \quad s\neq s' \end{cases}.$$

Hence, the variance-covariance matrix of $\hat{\theta}$ is

$$Var(\hat{\theta}) = (U'V^{-1}U)^{-1}$$

$$= \left[\begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial (S_2)}{\partial \sigma'} \end{pmatrix}' \begin{pmatrix} V(S_1) & Cov(S_1, S_2) \\ Cov(S_2, S_1) & V(S_2) \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial E(S_1)}{\partial b'} & \frac{\partial E(S_1)}{\partial \sigma'} \\ \frac{\partial E(S_2)}{\partial b'} & \frac{\partial E(S_2)}{\partial \sigma;} \end{pmatrix} \right]^{-1},$$

where $U = \frac{\partial (E(S_1', S_2')')}{\partial \theta'}$, $S_1 = (s_1, \ldots, s_k)'$, $S_2 = (s_1^2, \ldots, s_k^2)'$ and $S = (S_1', S_2')'$.

4.10 References

Bliemer, M. C., and Rose, J. M. (2010). "Construction of experimental designs for mixed logit models allowing for correlation across choice observations", *Transportation Research Part B: Methodological*, 44(6), 720−734.

Breslow, N. E., & Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**(421), 9−25.

Huber, J., and Zwerina, K. (1996). "The importance of utility balance in efficient choice designs", *Journal of Marketing research*, 307−317.

Jiang, J., & Zhang, W. (2001). Robust estimation in generalised linear mixed models. *Biometrika*, **88**(3), 753−765.

Moerbeek, M., & Maas, C. J. (2005). Optimal experimental designs for multilevel logistic models with two binary predictors. *Communications in Statistics− Theory and Methods*, **34**(5), 1151−1167.

Niaparast, M. (2009). On optimal design for a Poisson regression model with random intercept. *Statistics & Probability Letters*, **79**(6), 741−747.

Niaparast, M., & Schwabe, R. (2013). Optimal design for quasi-likelihood estimation in Poisson regression with random coefficients. *Journal of Statistical Planning and Inference*, **143**(2), 296−306.

Ogungbenro, K., & Aarons, L. (2011). Population Fisher information matrix and optimal design of discrete data responses in population pharmacodynamic experiments. *Journal of pharmacokinetics and pharmacodynamics*, **38**(4), 449−469.

Sándor, Z., and Wedel, M. (2001). "Designing conjoint choice experiments using managers' prior beliefs", *Journal of Marketing Research*, 430−444.

Sándor, Z., and Wedel, M. (2002). "Profile construction in experimental choice designs for mixed logit models", *Marketing Science*, 21(4), 455−475.

Street, D. J., and Burgess, L. (2007). *The construction of optimal stated choice experiments: theory and methods* , Wiley-Interscience.

Tekle, F. B., Tan, F. E., & Berger, M. P. (2008). Maximin D-optimal designs for binary longitudinal responses. *Computational Statistics & Data Analysis*, **52**(12), 5253−5262.

Veldwijk, J., Lambooij, M. S., de Bekker-Grob, E. W., Smit, H. A., & De Wit, G. A., The effect of including an opt-out option in discrete choice experiments. *PloS one*, 9(11), (2014).

Waite, T. W., Woods, D. C., & Waterhouse, T. H. (2012). Designs for generalized linear models with random block effects.

Yu, J., Goos, P., and Vandebroek, M. (2009). "Efficient conjoint choice designs in the presence of respondent heterogeneity", *Marketing Science*, 28(1), 122−135.

Yu, J., Goos, P., and Vandebroek, M. (2011). "Individually adapted sequential Bayesian conjoint-choice designs in the presence of consumer heterogeneity", *International Journal of Research in Marketing*, 28(4), 378−388.

Chapter 5

Conclusion

In this thesis, we are interested in optimal designs for discrete choice models. In discrete choice experiments, respondents are recruited to compare products or services which are assumed to be represented by a few characteristics of the product or services. The design problem is to decide which combinations should the respondents see and compare. A good design could give a good understanding of how people make trade offs between the characteristics and make their choices giving the information they have under the constraint of the number of respondents and the time people have. The panel mixed logit model can account for the heterogeneity in the respondents and correlation between choices made by the same respondents, compared to multinomial logit model which estimate of the average of the respondents and cross-sectional mixed logit model which assumes independence of the choices from the same respondent. We give a review of papers on optimal designs for multinomial logit model and cross-sectional mixed logit in chapter 1. For the panel mixed logit model, the information matrix, which is often used in design criteria, does not have a closed form expression. In chapter 3, we propose three approximations of the information matrix which make use of the formula derived in the chapter. In chapter 4, we propose three alternative to the information matrix based on approximate analysis methods for the generalized

linear mixed models−the panel mixed logit is a special case of the generalized linear mixed models. The approximations are used in the computer search for finding optimal designs in chapter 4, which we show that the design criteria based on the approximations of the information matrix in chapter 3 and 4 are much faster to evaluate than the information matrix.

In Chapter 4, we give the time needed for one run of the coordinate exchange algorithm for each method. More runs are needed to avoid local optima. For the coordinate exchange algorithm, the runs can be carried out separately. If running in parallel at the same time, the time needed to finish all the runs will be similar to the time for one run of the coordinate exchange algorithm. If the runs are carried out sequentially, the time needed to finish all runs will be the number of runs times the time for one run of the coordinate exchange algorithm, which will grow significantly. In chapter 4, we consider a few design scenarios. The time needed may be longer than the time reported in Chapter 4 for a scenario with more choice sets, more alternatives in a choice set, more random effects or more extreme parameter values.

For the panel mixed logit model, the information matrix depends on the unknown parameters. We consider locally optimal designs, where optimal designs are found for given parameter values. The optimal designs can be less efficient if the true parameter values are very different from the assumed ones. There are more robust design criteria, e.g. Bayesian design criteria and minimax design criteria, which take into account the variation of the parameter values into account, but they will require more computation than the locally optimal design criteria and

are not considered in this Chapter.

Chapter 6

Appendix: Code for Chapter 3 and 4

6.1 Code for Laplace Approximation

The following code is for implementing coordinate exchange algorithm with design criteria based on Laplace approximation in Section 3.3.2. The Fortran code needs to be compiled first before running the R code. The coordinate exchange algorithm in the R code uses the Laplace approximation in the Fortran code to evaluate each design.

R code

```
library(DoE.base)
library(MASS)
library(Matrix)

##############################
#          main     141013   #
##############################
S = 10               # number of choice sets
J = 2               # number of alternatives per choice sets
# cs = c(,,,,)        # input vector, the number of alternatives in each choice set, S, J can be read from cs
cs=rep(J,S)           # a vector for numbers of alternatives in choice sets, used
ns=length(cs)         # number of choice sets reading from the length of cs
nl=c(3,3,3,3,3)#,3,3)         # a vector for the number of levels for attributes
#b = c(0,0,0,0)
#sigma = c(1,1,1,1)
b = 3*c(1,0,1,0,1,0,1,0,1,0)
sigma = 9*c(rep(1,8),rep(0,2))
#b=c(.4,.6,1,2)#,1,0,1,0)              # value of mean vector
```

```
#sigma=c(.5,.5,1,1)#,1,1,1,1)     # value of the parameters in covariance matrix
nb2 = 200                          # number of draws for random effects
ny = 1000                          # sample size of y if we are not using ally

indexr = (1:length(b))[sigma!=0]
indexf = (1:length(b))[sigma==0]
br = b[indexr]
sigmar = sigma[indexr]
bf = b[indexf]



# b = .5*c(1,0,1,0)
# sigma = 1.5*rep(1,4)
# b = .5*c(1,0,1,0)
# sigma = .25*rep(1,4)
nattr<-length(nl)                # number of attributes
dimb<-sum(nl)-nattr              # number of parameters for the main effects

dimbr <- length(br)
qes<- rep(factor(1:ns),times=cs) # a vector of ids of the alternatives in each choice set.
ind <- c(0,cumsum(cs))

if( dimb > dimbr)
{
                dyn.load('laplaceybm.so')
        laplace = function(modmat,N=100)
        {   I = matrix(1,dimb+dimbr,dimb+dimbr)
                sigmai = solve(diag(sigmar))
                res=.Fortran('laplaceybm',as.integer(ny),as.integer(modmat),as.integer(c), as.integer(S),as.
                     integer(J),as.double(b),as.double(sigma),as.double(sigmai),as.integer(dimb),as.integer(
                     dimbr),as.integer(N),I=as.double(I))
                return(matrix(res$I,dimb+dimbr,dimb+dimbr))
        }
}else{
        dyn.load('laplaceyb.so')
        I = matrix(1,2*dimb,2*dimb)
        laplace = function(modmat,N=100)
        {
                sigmai=solve(diag(sigma))
                # laplace(y,ny,uy,modmat,s,j,b,sigma,dimb,n,infm)
                res=.Fortran('laplaceyb',as.integer(ny),as.integer(modmat),as.integer(c), as.integer(S),as.
                     integer(J),as.double(b),as.double(sigma),as.double(sigmai),as.integer(dimb),as.integer(N
                     ),I=as.double(I))
                return(matrix(res$I,2*dimb,2*dimb))
        }
}
```

```
c=100000

#############################################################
#  ordinate exchange alg for generating designs             #
#############################################################

do<-Inf
contr=NULL
R = 10                                  # number of starting designs for coordinate exchange alg
design<-array(0,c(R,sum(cs),nattr)) # optimal designs from R starting designs
designc<-array(0,c(R,sum(cs),dimb)) # coded values of design
derr <- rep(0,R)                    # the determinant of the R optimal designs
count=0
for(r in 1:R)
  {
         st=NULL
         fullfac=fac.design(nlevels=nl,random=F)
         for(i in 1:S){st=c(st,sample(prod(nl),J,replace=T))}
         designm=(fullfac)[st,]
     if(is.null(contr))
             {
                  contr=rep('contr.sum',nattr)
                  names(contr)=names(designm)
                  contr=as.list(contr)
              }
     modmat=model.matrix(~.,designm,contrasts = contr)[,-1]  #contr is used to get effects type coding,
          current coded design
     Ic=laplace(modmat)
     #Ic=round(Ic,digits=10)
         #if(rankMatrix(Infm[,,j])<(2*dimb))
          if((rankMatrix(Ic)<(dimb+dimbr)))
         #if(rankMatrix(Ijt[,,j])<(2*dimb))
        {
                dc = Inf
        }else{
                if(det(Ic)<0)
                {
                        dc=Inf
                }else{
                # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)/(2*dimb)
                # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
       #dc=abs(det(Ic))^{-1/(2*dimb)} # determinant of the current design
       dc=sum(eigen(solve(Ic))$values)/(dimb+dimbr)
                # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)/(2*dimb)
                # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
                if(dc < 0)
                  {
```

```
                    dc = Inf
              }
            }}
new=matrix()               # new design
##n,n,i,j,k,l
m=1
while(m!=0)                # if no exchange is made, then m=0
  {
     n=0                                  # number of exchange
     for(i in 1:(sum(cs)*nattr))          # i goes through all elements in the uncoded design matrix
             {
                 j=(i%%nattr)                       # column in uncoded design matrix, i.e., jth
                      attribute
                 if(j==0) {j=nattr}
                 k=(i-j)/nattr+1                    # row in uncoded design matrix, i.e., kth row
                 ch=ceiling(k/J)                    # the 'ch'th choice set
                 diff=setdiff(1:nl[j],designm[k,j]) # possible levels for exchange
                 for(l in diff)
                         {
                              new=designm
                              new[k,j]=l                                        # uncoded design
                                   matrix after exchange
                              modmatnew=model.matrix(~.,new,contrasts=contr)[,-1] # coded matrix of new
                                          result1 = fi(modmat)
                                              I1=laplace(modmatnew)
                                              #c=round(Ic,digits=10)
                                                    #if(rankMatrix(Infm[,,j])<(2*dimb))
                                                     if((rankMatrix(I1)<(dimb+dimbr)))
                                                    #if(rankMatrix(Ijt[,,j])<(2*dimb))
                                                   {
                                                           d1 = Inf
                                                   }else{
                                                           if(det(I1)<0)
                                                           {
                                                                   d1=Inf
                                                           }else{
                                                           # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)
                                                                /(2*dimb)
                                                           # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
                                                      #d1 = abs(det(I1))^{-1/(2*dimb)} # determinant of the
                                                           current design
                                                      d1=sum(eigen(solve(I1))$values)/(dimb+dimbr)
                                                           # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)
                                                                /(2*dimb)
                                                           # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
                                                           if(d1 < 0)
                                          {
```

```
                                        d1 = Inf
                                       }
                                                    }
                                        }
                                if (d1<dc)
                                   {
                                     designm=new
                                     modmat=modmatnew
                                     Ic=I1
                                     dc=d1
                                     n=n+1       # exchange is kept, add 1 to number of change
                                    }
                                   if(d1 == dc)
                                   {
                                     u = runif(1,0,1)
                                     if(u < 0.5)
                                     {
                                     designm=new
                                     modmat=modmatnew
                                     Ic=I1
                                     dc=d1
                                     n=n+1       # exchange is kept, add 1 to number of change
                                     }
                                   }
                                        count=count+1
                             } #l
                          print(dc)
                      }# i
                  m=n
           }#end of while
         design[r,,]=as.matrix(designm)
         designc[r,,]=modmat
         derr[r]=dc
     if(dc<do)
       {
         Io=Ic        # information matrix of the optimal design
         opt=designm  # optimal design
         mo=modmat    # coded optimal design
         do=dc        # determinant of optimal design
        }
         save.image('rcode.RData')
   }# r

design.R = matrix(0,S*J,nattr*R)
for(i in 1:R)
{
        design.R[,((i-1)*nattr+1):(i*nattr)]=design[i,,]
```

```
}
write.table(design.R,'design.txt',row.names=F,col.names=F)
write.table(opt,'opt.txt',row.names=F,col.names=F)
write.table(do,'do.txt',col.names=F,row.names=F)
write.table(derr,'derr.txt',col.names=F,row.names=F)
save.image('rcode.RData')
```

Fortran Code

```
      subroutine laplaceybm(ny,modmat,c,s,j,b,sigma,sigmami,dimb,dimbr,n,infm)
      implicit none
      integer, intent(in)                              :: ny,s,j,n,dimb,c,dimbr          ! stop after n runs
           of newton's alg
      integer, dimension(S*J,ny)                       :: y
      double precision, dimension(dimbr,ny)             :: uy
      double precision, intent(in), dimension(dimb,1)  :: b,sigma
      integer, intent(in),dimension(S*J,dimb)          :: modmat
      double precision, intent(in), dimension(dimbr,dimbr)  :: sigmami             ! inverse of diag(sigma)
      integer                                          :: i, i1,j1,j2,nyd,jj,l1,l2,ind1,ind2,i2,i3,ind
      integer, dimension(j,dimb)                       :: modmatj                ! modmat for a choice set
      integer, dimension(j,dimbr)                       :: modmatjr               ! modmat for a choice set
      integer, dimension(s*j,dimbr)                        :: modmatr                ! modmat for a choice set
      double precision,dimension(j,j)                  :: deltaj                 ! delta for a choice set
      double precision                                 :: sumy,denom,hddet,u,stepsize
      double precision, dimension(s*j,1)               :: p,pp,uty
      double precision, dimension(j,1)                 :: pj,postptemp,ptemp,postptemp1
      double precision, dimension(s*j,1)               :: epy,num1,eps,v
      double precision, dimension(dimbr,1)             :: eu2,num2,gr,mleu0,mleu1,mstart,sigmar   ! e(u^2/
           sigma^3|y)
      double precision, dimension(dimb,1)              :: bb
      integer, dimension(dimbr,1)                       :: ej
      integer, dimension(1,2)                          :: pos
      double precision, dimension(dimb,1)              :: score1,beta
      double precision, dimension(dimbr,1)             :: score2
      double precision, dimension(dimb,dimb)           :: i11
      double precision, dimension(dimb,dimbr)          :: i12
      double precision, dimension(dimbr,dimbr)           :: i22,he,hd,hei
      double precision, dimension(dimb+dimbr,dimb+dimbr) :: infm
      double precision, dimension(s*j,s*j)             :: delta
      double precision, parameter                      :: tol = 1.d-12
      integer                                          :: con
      double precision, PARAMETER                      :: Pi = 3.14159265359
      uy = 0
      u = 0
      sigmar = 0
      modmatr = 0
      modmatjr = 0
```

```
do i1 = 1,dimbr
  sigmar(i1,1) = sigma(i1,1)
end do
Do I1 = 1,ny        ! number of y
  DO I2=1,(S*J)       !
      CALL RANDOM_NUMBER(u)
      eps(I2,1)=-log(-log(U))
  end do !I2
  do I2 = 1,dimbr
      call random_normal(uy(i2,i1))
      uy(i2,i1) = uy(i2,i1)*sqrt(sigma(i2,1))
      beta(i2,1) = b(i2,1) + uy(i2,i1)
  end do
  do i2 = (dimbr+1),dimb
     beta(i2,1) = b(i2,1)
  end do
  v(:,1)=matmul(modmat,beta(:,1))+eps(:,1) ! utility of one y vector
  y(:,i1)=0
  do I2=1,S         ! I2 choice set
      ind= (I2-1)*J+1 ! 1st alt in I2, index for the chosen alt
      y(IND,I1)=1     ! current chosen alternative, 1st alt in choice set
      do I3=2,J      ! 2 to Jth alternative
          if(v((I2-1)*J+I3,1)>v(ind,1)) then ! if the alt larger than the current chosen one
          y((I2-1)*J+I3,I1)=1              ! chosen
          y(ind,I1)=0                      ! not chose
          ind=(I2-1)*J+I3                  ! update the index for the chosen one
          end if
      end do !I3,alt
  end do    !I2,choice set
end do      !I1,y

hei = 0
!delta = 0
!p = 0
hd = 0
hddet = 0
i11 = 0
i12 = 0
i22 = 0
nyd = ny
do i1 = 1,dimbr
   modmatr(:,i1) = modmat(:,i1)
end do
do i=1,ny
  ! get E(p|y),E(u^2/simga^3|y) from laplace approximation
  ! epy
  !mleu0(:,1) = uy(:,i)         !starting value for u
```

```
mleu0(:,1) = uy(:,i)          !starting value for u
con = 0                       ! 1 is newton's alg for denominator converges
j1 = 1                        ! find maximizer u for denominator
do while (j1<=n)
  do l1 = 1,dimbr
   bb(l1,1) = b(l1,1) + mleu0(l1,1)
  end do
  do l1 = (dimbr+1),dimb
    bb(l1,1) = b(l1,1)
  end do
  UTY = exp(MATMUL(modmat,bb))
  delta = 0
  DO l1 = 1,S ! the probability of y,
    ind1 = (l1-1)*j+1
    ind2 =  l1*j
    SUMY =  sum(UTY(ind1:ind2,1))
    p(ind1:ind2,1)=UTY(ind1:ind2,1)/SUMY
    ptemp(:,1) = p(ind1:ind2,1)
    delta(ind1:ind2,ind1:ind2) = - matmul(ptemp,transpose(ptemp))
    do l2 = 1,j
        delta(ind1-1+l2,ind1-1+l2) = delta(ind1-1+l2,ind1-1+l2)+p(ind1-1+l2,1)
    end do
  end do
  gr(:,1) = matmul(transpose(modmatr),y(:,i)-p(:,1))- matmul(sigmami,mleu0(:,1))
  he = -matmul(matmul(transpose(modmatr),delta),modmatr)-sigmami
  stepsize=1/(1+10*sqrt(sum(gr(:,1)**2)))
  call solve(dimbr,he,hei)
  mleu1 = mleu0 - stepsize*matmul(hei,gr)
  if( sqrt(sum((mleu1-mleu0)**2))<tol) then
    j1 = n+1
    con = 1
  else
    mleu0 = mleu1
    j1 = j1+1
  end if
end do ! while
if(con==1) then   ! the newton algrithm in the denominator converges
     call solve(dimbr, he, hd)
     hd = -hd
     call det(dimbr, hd, hddet)
     denom = sqrt(hddet)*product(p(:,1)**y(:,i))*product((1/(sqrt(2*pi*SIGMAr)))*exp(-(mleu0**2)/(2*
          sigmar)))

     mstart = mleu0
     do j1 = 1,S*J                              ! element in the numerator vector
        mleu0 = mstart                          ! starting value for the jth element
        i1 = floor((j1-1.0)/J)+1                ! ith choice set
```

```
 modmatj = modmat(((i1-1)*j+1):(i1*j),:) ! modmat for the ith choice set
 modmatjr = modmatj(:,1:dimbr)
 jj = 1
do while(jj <= n)
     do l1 = 1,dimbr
         bb(l1,1) = b(l1,1) + mleu0(l1,1)
     end do
     do l1 = (dimbr+1),dimb
         bb(l1,1) = b(l1,1)
     end do
     UTY = exp(MATMUL(modmat,bb))
     delta = 0
     DO l1 = 1,S ! the probability of y,
         ind1 = (l1-1)*j+1
         ind2 =  l1*j
         SUMY =  sum(UTY(ind1:ind2,1))
         p(ind1:ind2,1)=UTY(ind1:ind2,1)/SUMY
         ptemp(:,1) = p(ind1:ind2,1)
         delta(ind1:ind2,ind1:ind2) = - matmul(ptemp,transpose(ptemp))
         do l2 = 1,j
             delta(ind1-1+l2,ind1-1+l2) = delta(ind1-1+l2,ind1-1+l2)+p(ind1-1+l2,1)
         end do
     end do
     pp = p
     pp(j1,1) = -1+p(j1,1)
     pj(:,1) = -pp(((i1-1)*j+1):(i1*j),1)
     gr(:,1) = matmul(transpose(modmatjr),pj(:,1))&
               +matmul(transpose(modmatr),y(:,i)-p(:,1))&
              -matmul(sigmami,mleu0(:,1))
     deltaj = delta(((i1-1)*j+1):(i1*j),((i1-1)*j+1):(i1*j))
     he = - matmul(matmul(transpose(modmatjr),deltaj),modmatjr)&
          -matmul(matmul(transpose(modmatr),delta),modmatr)-sigmami
     call solve(dimbr,he,hei)
     stepsize=1/(1+10*sqrt(sum(gr(:,1)**2)))
     mleu1 =  mleu0 - stepsize*matmul(hei,gr)
     if( sqrt(sum((mleu1-mleu0)**2))<tol) then
         jj = n+1
     else
         mleu0 = mleu1
         jj = jj+1
     end if
 end do ! while
 call solve(dimbr, he, hd)
 hd = -hd
 call det(dimbr, hd, hddet)
 num1(j1,1) = sqrt(hddet)*p(j1,1)*product(p(:,1)**y(:,i))*product((1/(sqrt(2*pi*SIGMAr)))&
           *exp(-(mleu0**2)/(2*sigmar)))
```

```
end do !j1
do j1 = 1,dimbr                                  ! element in the numerator vector
    mleu0 = mstart                               ! starting value for the jth element
    jj = 1
    ej = 0
    ej(j1,1) =1
    do while(jj <= n)
      do l1 = 1,dimbr
          bb(l1,1) = b(l1,1) + mleu0(l1,1)
      end do
      do l1 = (dimbr+1),dimb
          bb(l1,1) = b(l1,1)
      end do
        UTY = exp(MATMUL(modmat,bb))
        delta = 0
        DO l1 = 1,S ! the probability of y,
          ind1 = (l1-1)*j+1
          ind2 =  l1*j
          SUMY =  sum(UTY(ind1:ind2,1))
          p(ind1:ind2,1)=UTY(ind1:ind2,1)/SUMY
          ptemp(:,1) = p(ind1:ind2,1)
          delta(ind1:ind2,ind1:ind2) = - matmul(ptemp,transpose(ptemp))
          do l2 = 1,j
              delta(ind1-1+l2,ind1-1+l2) = delta(ind1-1+l2,ind1-1+l2)+p(ind1-1+l2,1)
          end do
        end do
        gr(:,1) =(2*mleu0(j1,1)/(mleu0(j1,1)**2+c*sqrt(sigmar(j1,1))**3))*ej(:,1)&
          +matmul(transpose(modmatr),y(:,i)-p(:,1))&
          -matmul(sigmami,mleu0(:,1))
        he = (2*(c*sqrt(sigmar(j1,1))**3-mleu0(j1,1)**2)/(mleu0(j1,1)**2+c*sqrt(sigmar(j1,1))
             **3)**2)&
          *matmul(ej,transpose(ej))&
          -matmul(matmul(transpose(modmatr),delta),modmatr)-sigmami
        call solve(dimbr,he,hei)
        stepsize=1/(1+10*sqrt(sum(gr(:,1)**2)))
        mleu1 =  mleu0 - stepsize*matmul(hei,gr)
        if( sqrt(sum((mleu1-mleu0)**2))<tol) then
            jj = n+1
        else
            mleu0 = mleu1
            jj = jj+1
        end if
    end do ! while
    call solve(dimbr, he, hd)
    hd = -hd
    call det(dimbr, hd, hddet)
   ! num2(j1,1) = sqrt(hddet)*((mleu0(j1,1)**2+100)/(sqrt(sigma(j1,1))**3))&
```

```
              !               *product(p(:,1)**y(:,i))*product((1/(sqrt(2*pi*SIGMA)))*exp(-(mleu0**2)/(2*
                   sigma)))
               num2(j1,1) = sqrt(hddet)*((mleu0(j1,1)**2+c*sqrt(sigmar(j1,1))**3)/(sqrt(sigmar(j1,1))**3))
                    &
                 *product(p(:,1)**y(:,i))*product((1/(sqrt(2*pi*SIGMAr)))*exp(-(mleu0**2)/(2*sigmar)))
          end do !j1
          epy = num1/denom
          do j2 = 1,s! redundancies
            IND1=(J2-1)*j+1
            ind2 = j2*j
            epy(ind1:ind2,1) = epy(ind1:ind2,1) / sum(epy(ind1:ind2,1))
            !postptemp(:,1)=epy(ind1:ind2,1)
            !pos(1,:) = minloc(postptemp)
            !postptemp1(:,1) = postptemp(:,1)
            !postptemp1(:,pos(1,1)) = -1
            !pos(1,:) = maxloc(postptemp1)
            !postptemp(pos(1,1),1)=1-sum(postptemp)+postptemp(pos(1,1),1)
            !epy(ind1:ind2,1)=postptemp(:,1)
          end do
          eu2 = num2/denom-c
! end of epy, eu2
          score1(:,1) = matmul(transpose(modmat),y(:,i)-epy(:,1))
          score2 =-1/sqrt(sigmar) + eu2
          do j2 = 1, dimbr
            if(eu2(j2,1)<0) then
              score1(:,1)=0
              score2(:,1)=0
            end if
          end do
          if(score2(1,1)==0) then
            nyd = nyd-1
          end if
          i11 = i11 + matmul(score1,transpose(score1))
          i12 = i12 + matmul(score1,transpose(score2))
          i22 = i22 + matmul(score2,transpose(score2))
    else
          nyd = nyd -1
    end if
  end do ! i
  infm(1:dimb,1:dimb) = I11
  infm(1:dimb,(dimb+1):(dimb+dimbr)) = I12
  infm((dimb+1):(dimb+dimbr),1:dimb) = TRANSPOSE(I12)
  infm((dimb+1):(dimb+dimbr),(dimb+1):(dimb+dimbr)) = I22
  infm=infm/nyd
  return
  end subroutine laplaceybm
```

6.2 Code for MSM

The following code is for implementing coordinate exchange algorithm with design criteria based on method of simulated moments in Section 4.4. The Fortran code needs to be compiled first before running the R code. The coordinate exchange algorithm in the R code uses the MSM approximation in the Fortran code to evaluate each design.

R code

```
library(DoE.base)
library(MASS)
library(Matrix)

##############################
#          main     141013   #
##############################
S = 10               # number of choice sets
J = 2               # number of alternatives per choice sets
# cs = c(,,,,)        # input vector, the number of alternatives in each choice set, S, J can be read from cs
cs=rep(J,S)           # a vector for numbers of alternatives in choice sets, used
ns=length(cs)         # number of choice sets reading from the length of cs
nl=c(3,3,3,3,3)#,3,3)         # a vector for the number of levels for attributes
#b = c(0,0,0,0)
#sigma = c(1,1,1,1)
b = 3*c(1,0,1,0,1,0,1,0,1,0)
sigma = 9*c(rep(1,8),rep(0,2))
#b=c(.4,.6,1,2)#,1,0,1,0)               # value of mean vector
#sigma=c(.5,.5,1,1)#,1,1,1,1)     # value of the parameters in covariance matrix
nb2 = 200                          # number of draws for random effects
ny = 1000                          # sample size of y if we are not using ally
nu = 1000

indexr = (1:length(b))[sigma!=0]
indexf = (1:length(b))[sigma==0]
br = b[indexr]
sigmar = sigma[indexr]
bf = b[indexf]
```

```
# b = .5*c(1,0,1,0)
# sigma = 1.5*rep(1,4)
# b = .5*c(1,0,1,0)
# sigma = .25*rep(1,4)
nattr<-length(nl)                # number of attributes
dimb<-sum(nl)-nattr              # number of parameters for the main effects

dimbr <- length(br)
qes<- rep(factor(1:ns),times=cs) # a vector of ids of the alternatives in each choice set.
ind <- c(0,cumsum(cs))



dyn.load('momm.so')
vm = matrix(1,dimb+dimbr,dimb+dimbr)
du = matrix(1,dimb+dimbr,dimb+dimbr)
mom = function(modmat,ny,nu)
{
        # laplace(y,ny,uy,modmat,s,j,b,sigma,dimb,n,infm)
        res=.Fortran('mom',as.integer(ny),as.integer(nu),as.integer(modmat),as.integer(S),as.integer(J),as.
             double(b),as.double(sigma),as.integer(dimb),as.integer(dimbr),vm=as.double(vm),du=as.double(du))
        vm=matrix(res$vm,dimb+dimbr,dimb+dimbr)
        du=matrix(res$du,dimb+dimbr,dimb+dimbr)
        return(I=t(du)%*%ginv(vm)%*%du)
}

############################################################
#  ordinate exchange alg for generating designs            #
############################################################

do<-Inf
contr=NULL
R=10                                # number of starting designs for coordinate exchange alg
design<-array(0,c(R,sum(cs),nattr)) # optimal designs from R starting designs
designc<-array(0,c(R,sum(cs),dimb)) # coded values of design
derr <- rep(0,R)                    # the determinant of the R optimal designs
count=0
for(r in 1:R)
  {
         st=NULL
         fullfac=fac.design(nlevels=nl,random=F)
         for(i in 1:S){st=c(st,sample(prod(nl),J,replace=T))}
         designm=(fullfac)[st,]
     if(is.null(contr))
             {
                  contr=rep('contr.sum',nattr)
                  names(contr)=names(designm)
```

```
              contr=as.list(contr)
          }
 modmat=model.matrix(~.,designm,contrasts = contr)[,-1]  #contr is used to get effects type coding,
       current coded design
 Ic=mom(modmat,ny,nu)
 Ic=round(Ic,digits=10)
     #if(rankMatrix(Infm[,,j])<(2*dimb))

             if(all(is.na(Ic))==T)
    {
            dc = Inf
    }else{
                    if(rankMatrix(Ic)<(dimbr+dimb)||isSymmetric(Ic)==F)
                    { dc = Inf
                    }      else{
            # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)/(2*dimb)
            # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
        # a.lap[j]=sum(eigen(solve(Ilap[,,j]))$values)/(2*dimb)
    # d.lap[j]=(det(Ilap[,,j]))^(-1/(2*dimb))
            dc=sum(eigen(solve(Ic))$values)/(dimb+dimbr)
            #dc=(det(Ic))^(-1/(2*dimb))
    if(dc < 0)
              {
                    dc = Inf
              }
            }
}
 new=matrix()              # new design
 ##n,n,i,j,k,l
 m=1
 while(m!=0)               # if no exchange is made, then m=0
   {
      n=0                                  # number of exchange
      for(i in 1:(sum(cs)*nattr))          # i goes through all elements in the uncoded design matrix
              {
                  j=(i%%nattr)                      # column in uncoded design matrix, i.e., jth
                       attribute
                  if(j==0) {j=nattr}
                  k=(i-j)/nattr+1                   # row in uncoded design matrix, i.e., kth row
                  ch=ceiling(k/J)                   # the 'ch'th choice set
                  diff=setdiff(1:nl[j],designm[k,j]) # possible levels for exchange
                  for(l in diff)
                          {
                               new=designm
                               new[k,j]=l                                        # uncoded design
                                    matrix after exchange
```

```
    modmatnew=model.matrix(~.,new,contrasts=contr)[,-1] # coded matrix of new
                result1 = fi(modmat)
                    I1=mom(modmatnew,ny,nu)
                    I1=round(I1,digits=10)
                          #if(rankMatrix(Infm[,,j])<(2*dimb))
               if(all(is.na(I1))==T)
                  {
                           d1 = Inf
                  }else{
                                    if(rankMatrix(I1)<(dimb+dimbr)||isSymmetric(
                                         I1)==F)
                                    { d1 = Inf
                                    }      else{
                                # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)/(2*dimb)
                                # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
                            # a.lap[j]=sum(eigen(solve(Ilap[,,j]))$values)/(2*dimb)
                  # d.lap[j]=(det(Ilap[,,j]))^(-1/(2*dimb))
                           d1=sum(eigen(solve(I1))$values)/(dimb+dimbr)
                           #d1=(det(I1))^(-1/(2*dimb))
                                   if(d1 < 0)
                {
                  d1 = Inf
                }
                           }
     }
    if (d1<dc)
       {
         designm=new
         modmat=modmatnew
         Ic=I1
         dc=d1
         n=n+1       # exchange is kept, add 1 to number of change
       }
      if(d1 == dc)
      {
         u = runif(1,0,1)
         if(u < 0.5)
         {
         designm=new
         modmat=modmatnew
         Ic=I1
         dc=d1
         n=n+1       # exchange is kept, add 1 to number of change
         }
      }
                 count=count+1
} #1
```

```
                        print(dc)
                    }# i
                  m=n
            }#end of while
          design[r,,]=as.matrix(designm)
          designc[r,,]=modmat
          derr[r]=dc
      if(dc<do)
        {
          Io=Ic        # information matrix of the optimal design
          opt=designm  # optimal design
          mo=modmat    # coded optimal design
          do=dc        # determinant of optimal design
         }
          save.image('rcode.RData')
    }# r
design.R = matrix(0,S*J,nattr*R)
for(i in 1:R)
{
        design.R[,((i-1)*nattr+1):(i*nattr)]=design[i,,]
}
write.table(design.R,'design.txt',row.names=F,col.names=F)
write.table(opt,'opt.txt',row.names=F,col.names=F)
write.table(do,'do.txt',col.names=F,row.names=F)
write.table(derr,'derr.txt',col.names=F,row.names=F)

save.image('rcode.RData')
```

Fortran Code

```
      subroutine mom(ny,nu,modmat,s,j,b,sigma,dimb,dimbr,vm1,du1)
      implicit none
      integer, dimension(S*J,ny)                      :: y
      integer, intent(in)                             :: ny,nu,s,j,dimb,dimbr          ! stop after n runs
           of newton's alg
      double precision, dimension(dimb,ny)            :: uy
      double precision, dimension(dimb,nu)            :: un
      double precision, intent(in), dimension(dimb,1) :: b,sigma
      double precision, dimension(s*j,nu)             :: p
      double precision, dimension(s*j,1)              :: ep
      double precision, dimension(s*j,dimb)           :: epu, u11, u12
      double precision, dimension((s*j)**2,1)         :: epp
      double precision, dimension((s*j)**2,dimb)      :: eppu, u21,u22, xx
      double precision, dimension((s*j)**3,1)         :: eppp
      double precision, dimension((s*j)**3,dimb)      :: epppu
      integer, intent(in),dimension(S*J,dimb)         :: modmat
      double precision, dimension(dimb,dimb)          :: du11,du12,du21,du22
```

```
double precision,dimension(2*dimb,2*dimb)        :: du, vm
double precision, dimension(2*dimb,1)            :: em
double precision,dimension(2*dimb,ny)            :: mme
integer                                          :: i1,j1,j2,ind1,ind2,i2,i3,ind,i4,n1,n2,sttemp1,
     entemp1,s1,s2
integer                                          :: sttemp2,entemp2,index
double precision                                 :: sumy,u,ppi,pppi,ss1,ss2
double precision, dimension(s*j,1)               :: uty
double precision, dimension(s*j,1)               :: epy,eps,v,pi
double precision, dimension(dimb,1)              :: beta, ui
integer, dimension(s,1)                          :: st, en
double precision, dimension(1,dimb)              :: u12temp,u21temp, u22temp,tui,u11temp
double precision, dimension(dimb+dimbr,dimb+dimbr) :: vm1,vmi,du1,imom,vmom
uy = 0
u = 0
Do I1 = 1,ny        ! number of y
  DO I2=1,(S*J)       !
      CALL RANDOM_NUMBER(u)
      eps(I2,1)=-log(-log(U))
  end do !I2
  do I2 = 1,dimb
      call random_normal(uy(i2,i1))
      uy(i2,i1) = uy(i2,i1)*sqrt(sigma(i2,1))
  end do
  beta(:,1) = b(:,1) + uy(:,i1)
  v(:,1)=matmul(modmat,beta(:,1))+eps(:,1) ! utility of one y vector
  y(:,i1)=0
  do I2=1,S         ! I2 choice set
      ind= (I2-1)*J+1 ! 1st alt in I2, index for the chosen alt
      y(IND,I1)=1     ! current chosen alternative, 1st alt in choice set
      do I3=2,J      ! 2 to Jth alternative
          if(v((I2-1)*J+I3,1)>v(ind,1)) then ! if the alt larger than the current chosen one
          y((I2-1)*J+I3,I1)=1              ! chosen
          y(ind,I1)=0                      ! not chose
          ind=(I2-1)*J+I3                  ! update the index for the chosen one
          end if
      end do !I3,alt
  end do    !I2,choice set
end do      !I1,y

Do I1 = 1,nu        ! number of y
  do I2 = 1,dimb
      call random_normal(un(i2,i1))
      beta(i2,1) = b(i2,1) + un(i2,i1)*sqrt(sigma(i2,1))
  end do
  UTY = exp(MATMUL(modmat,beta))
  DO I2 = 1,S ! the probability of y,
```

```
      ind1 = (I2-1)*j+1
      ind2 =  I2*j
      SUMY =  sum(UTY(ind1:ind2,1))
      p(ind1:ind2,i1)=UTY(ind1:ind2,1)/SUMY
  end do
end do      !I1,y
ep = 0
epu = 0
epp = 0
eppu = 0
eppp = 0
epppu = 0
do i1 = 1,nu
  ep(:,1) = ep(:,1) + p(:,i1)
  pi(:,1) = p(:,i1)
  ui(:,1) = un(:,i1)
  tui = transpose(ui)
  epu = epu + matmul(pi,tui)
  do i2 = 1,(s*j)
     do i3 = 1,(s*j)
        ppi = p(i2,i1)*p(i3,i1)
        index = (i2-1)*s*j+i3
        epp(index,1) = epp(index,1) + ppi
        eppu(index,:) = eppu(index,:) + ppi*tui(1,:)
        do i4 = 1,(s*j)
           pppi = ppi*p(i4,i1)
           index = ((i2-1)*s*j+i3-1)*s*j +i4
           eppp(index,1) = eppp(index,1) + pppi
           epppu(index,:) = epppu(index,:) + pppi*tui(1,:)
        end do !i4
     end do !i3
  end do !i2
end do !i1
ep = ep/nu
epu = epu /nu
epp = epp / nu
eppu = eppu/nu
eppp = eppp/nu
epppu = epppu/nu

do i1 = 1, s
  st(i1,1) = (i1-1)*j+1
  en(i1,1) = i1*J
end do

s1 = 1
do n1 = 1,(s*j)
```

```
if(n1==(s1*j+1)) then
    s1 = s1+1
end if
j1 = n1-(s1-1)*j
sttemp1 = st(s1,1)
entemp1 = en(s1,1)
u11temp = 0
u12temp = 0
do i2 =1, j
    u11temp(1,:) = u11temp(1,:)+epp((n1-1)*(s*j)+sttemp1+i2-1,1)*modmat(sttemp1+i2-1,:)
    do i3 = 1, dimb
        u12temp(1,i3) = u12temp(1,i3)+modmat(sttemp1+i2-1,i3)*eppu((n1-1)*(s*j)+sttemp1+i2-1,i3)
    end do !i3
end do !i2
u11(n1,:) = ep(n1,1)*modmat(n1,:)-u11temp(1,:)
do i2 = 1,dimb
    u12(n1,i2) = modmat(n1,i2)*epu(n1,i2)-u12temp(1,i2)
end do !i2
s2 = 1
do n2 = 1,(s*j)
   if(n2==(s2*j+1)) then
     s2 = s2+1
   end if
   j2 = n2-(s2-1)*j
   sttemp2 = st(s2,1)
   entemp2 = en(s2,1)
   index = (n1-1)*s*j+n2
   if(s1 == s2) then
       if(j1==j2) then
            u21(index,:) = u11(n1,:)
            u22(index,:) = u12(n1,:)
       else
            u21(index,:) = 0
            u22(index,:) = 0
       end if
  else
      u21temp = 0
      u22temp = 0
      do i2 = 1,J
         ind1 = ((n1-1)*s*j+sttemp1+i2-1-1)*s*j+n2
         ind2 = ((n2-1)*s*j+sttemp2+i2-1-1)*s*j+n1
         u21temp(1,:) = u21temp(1,:) + eppp(ind1,1)*modmat(sttemp1+i2-1,:)+eppp(ind2,1)*modmat(
              sttemp2+i2-1,:)
         do i3 = 1,dimb
             u22temp(1,i3) = u22temp(1,i3) + modmat(sttemp1+i2-1,i3)*epppu(ind1,i3)+modmat(sttemp2+i2
                  -1,i3)*epppu(ind2,i3)
         end do !i3
```

```
        end do !i2
        u21(index,:) = epp(index,1)*(modmat(n1,:)+modmat(n2,:))-u21temp(1,:)
        do i2 = 1,dimb
           u22(index,i2) = (modmat(n1,i2)+modmat(n2,i2))*eppu(index,i2)-u22temp(1,i2)
        end do !i2
     end if
   end do !n2
end do !n1

do i1 = 1,dimb
   do i2 = 1,(s*j)
      do i3 = 1,(s*j)
         index = (i2-1)*s*j+i3
         xx(index,i1) = modmat(i2,i1)*modmat(i3,i1)
      end do !i3
   end do !i2
end do !i1

du11 = matmul(transpose(modmat),u11)
du12 = matmul(transpose(modmat),u12)
du21 = matmul(transpose(xx),u21)
du22 = matmul(transpose(xx),u22)
mme(1:dimb,:) = matmul(transpose(modmat),y)
mme((dimb+1):(2*dimb),:) = matmul(transpose(modmat),y)**2

vmi = 0
vmom = 0
em(:,1) = sum(mme,dim=2)/ny
vm = matmul(mme, transpose(mme))/ny-matmul(em,transpose(em))
vm1 = vm(1:(dimb+dimbr),1:(dimb+dimbr))
!call solve(dimb+dimbr,vm1,vmi)
du(1:dimb,1:dimb) = du11
du(1:dimb,(dimb+1):(2*dimb))=du12
du((dimb+1):(2*dimb),1:dimb) = du21
du((dimb+1):(2*dimb),(dimb+1):(2*dimb))=du22
du1 = du(1:(dimb+dimbr),1:(dimb+dimbr))
!imom = matmul(transpose(du1),matmul(vmi,du1))
!call solve((dimb+dimbr),imom,vmom)

return
end subroutine mom
```

6.3 Code for PQL

The following code is for implementing coordinate exchange algorithm with design criteria based on PQL in Section 4.3. The Fortran code needs to be compiled first before running the R code. The coordinate exchange algorithm in the R code uses the PQL approximation in the Fortran code to evaluate each design.

R Code

```
library(DoE.base)
library(MASS)
library(Matrix)

##############################
#          main     141013   #
##############################
S = 10                # number of choice sets
J = 2                # number of alternatives per choice sets
# cs = c(,,,,)         # input vector, the number of alternatives in each choice set, S, J can be read from cs
cs=rep(J,S)           # a vector for numbers of alternatives in choice sets, used
ns=length(cs)         # number of choice sets reading from the length of cs
nl=c(3,3,3,3,3)#,3,3)         # a vector for the number of levels for attributes
#b = c(0,0,0,0)
#sigma = c(1,1,1,1)
b = 3*c(1,0,1,0,1,0,1,0,1,0)
sigma = 9*c(rep(1,8),rep(0,2))
ny = 100

indexr = (1:length(b))[sigma!=0]
indexf = (1:length(b))[sigma==0]
br = b[indexr]
sigmar = sigma[indexr]
bf = b[indexf]

# b = .5*c(1,0,1,0)
# sigma = 1.5*rep(1,4)
# b = .5*c(1,0,1,0)
# sigma = .25*rep(1,4)
nattr<-length(nl)               # number of attributes
dimb<-sum(nl)-nattr             # number of parameters for the main effects
```

```
dimbr <- length(br)
qes<- rep(factor(1:ns),times=cs) # a vector of ids of the alternatives in each choice set.
ind <- c(0,cumsum(cs))


dyn.load('pql.so')

pql = function(modmat,N=100)
{   I = matrix(1,dimb+dimbr,dimb+dimbr)
        sigmai = solve(diag(sigmar))
res=.Fortran('pql',as.integer(ny),as.integer(modmat),as.integer(S),as.integer(J),as.double(b),as.double(sigma
     ),as.double(sigmai),as.integer(dimb),as.integer(dimbr),as.integer(N),I=as.double(I))
        return(matrix(res$I,dimb+dimbr,dimb+dimbr))
}



############################################################
#  ordinate exchange alg for generating designs             #
############################################################

do<-Inf
contr=NULL
R=10                                 # number of starting designs for coordinate exchange alg
design<-array(0,c(R,sum(cs),nattr)) # optimal designs from R starting designs
designc<-array(0,c(R,sum(cs),dimb)) # coded values of design
derr <- rep(0,R)                    # the determinant of the R optimal designs
count=0
for(r in 1:R)
  {
         st=NULL
         fullfac=fac.design(nlevels=nl,random=F)
         for(i in 1:S){st=c(st,sample(prod(nl),J,replace=T))}
         designm=(fullfac)[st,]
     if(is.null(contr))
             {
                  contr=rep('contr.sum',nattr)
                  names(contr)=names(designm)
                  contr=as.list(contr)
              }
     modmat=model.matrix(~.,designm,contrasts = contr)[,-1]  #contr is used to get effects type coding,
          current coded design
     Ic=pql(modmat)
     #Ic=round(Ic,digits=10)
         #if(rankMatrix(Infm[,,j])<(2*dimb))
                         if(any(is.na(Ic))==T)
        {
                dc = Inf
```

```
   }else{
     if((rankMatrix(Ic)<(dimb+dimbr))||isSymmetric(Ic)==F)
    #if(rankMatrix(Ijt[,,j])<(2*dimb))
   {
           dc = Inf
   }else{
           # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)/(2*dimb)
           # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
  #dc=abs(det(Ic))^{-1/(2*dimb)} # determinant of the current design
  dc=sum(eigen(solve(Ic))$values)/(dimb+dimbr)
           # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)/(2*dimb)
           # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
           if(dc < 0)
             {
                   dc = Inf
             }
           }}
new=matrix()              # new design
##n,n,i,j,k,l
m=1
while(m!=0)               # if no exchange is made, then m=0
  {
     n=0                                  # number of exchange
     for(i in 1:(sum(cs)*nattr))          # i goes through all elements in the uncoded design matrix
             {
                 j=(i%%nattr)                      # column in uncoded design matrix, i.e., jth
                      attribute
                 if(j==0) {j=nattr}
                 k=(i-j)/nattr+1                   # row in uncoded design matrix, i.e., kth row
                 ch=ceiling(k/J)                   # the 'ch'th choice set
                 diff=setdiff(1:nl[j],designm[k,j]) # possible levels for exchange
                 for(l in diff)
                         {
                              new=designm
                              new[k,j]=l                                        # uncoded design
                                   matrix after exchange
                              modmatnew=model.matrix(~.,new,contrasts=contr)[,-1] # coded matrix of new
                                          result1 = fi(modmat)
                                              I1=pql(modmatnew)
                                              #c=round(Ic,digits=10)
                                                    #if(rankMatrix(Infm[,,j])<(2*dimb))
                                                    if(any(is.na(I1))==T)
                                                   {
                                                           d1 = Inf
                                                   }else{
                                                     if((rankMatrix(I1)<(dimb+dimbr))||isSymmetric(I1)==
                                                          F)
```

```
                                        #if(rankMatrix(Ijt[,,j])<(2*dimb))
                                        {
                                                d1 = Inf
                                        }else{
                                                # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)
                                                      /(2*dimb)
                                                # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
                                       #d1 = abs(det(I1))^{-1/(2*dimb)} # determinant of the
                                             current design
                                       d1=sum(eigen(solve(I1))$values)/(dimb+dimbr)
                                                # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)
                                                      /(2*dimb)
                                                # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
                                                if(d1 < 0)
                                {
                                 d1 = Inf
                                }
                                 }}
                        if (d1<dc)
                           {
                             designm=new
                             modmat=modmatnew
                             Ic=I1
                             dc=d1
                             n=n+1       # exchange is kept, add 1 to number of change
                           }
                          if(d1 == dc)
                          {
                             u = runif(1,0,1)
                             if(u < 0.5)
                             {
                             designm=new
                             modmat=modmatnew
                             Ic=I1
                             dc=d1
                             n=n+1       # exchange is kept, add 1 to number of change
                             }
                          }
                                    count=count+1
                    } #l
                print(dc)
            }# i
          m=n
  }#end of while
design[r,,]=as.matrix(designm)
designc[r,,]=modmat
derr[r]=dc
```

```
    if(dc<do)
      {
        Io=Ic        # information matrix of the optimal design
        opt=designm  # optimal design
        mo=modmat    # coded optimal design
        do=dc        # determinant of optimal design
       }
        save.image('rcode.RData')
  }# r

design.R = matrix(0,S*J,nattr*R)
for(i in 1:R)
{
        design.R[,((i-1)*nattr+1):(i*nattr)]=design[i,,]
}
write.table(design.R,'design.txt',row.names=F,col.names=F)
write.table(opt,'opt.txt',row.names=F,col.names=F)
write.table(do,'do.txt',col.names=F,row.names=F)
write.table(derr,'derr.txt',col.names=F,row.names=F)

save.image('rcode.RData')
```

Fortran Code

```
      subroutine pql(ny,modmat,s,j,b,sigma,sigmami,dimb,dimbr,n,infm)
      implicit none
      integer, intent(in)                              :: ny,s,j,n,dimb,dimbr          ! stop after n runs of
            newton's alg
      integer, dimension(S*J,ny)                       :: y
      double precision, dimension(dimbr,ny)            :: uy
      double precision, intent(in), dimension(dimb,1)  :: b,sigma
      integer, intent(in),dimension(S*J,dimb)          :: modmat
      integer,dimension(S*(J-1),dimb)                  :: modmatd
      double precision, intent(in), dimension(dimbr,dimbr)  :: sigmami             ! inverse of diag(sigma)
      double precision,dimension(s*(j-1),s*(j-1))       :: deltad,deltadi,vn,vni
      double precision, dimension(dimb, dimb)          :: sigmam
      integer                                          :: i, i1,j1,nyd,l1,l2,ind1,ind2,i2,i3,ind
      integer, dimension(s*j,dimbr)                    :: modmatr                ! modmat for a choice set
      integer, dimension(s*(j-1),dimbr)                :: modmatrd
      double precision                                 :: sumy,denom,u,stepsize
      double precision, dimension(s*j,1)               :: p,uty
      double precision, dimension(j,1)                 :: pj,ptemp,postptemp1
      double precision, dimension(s*j,1)               :: epy,num1,eps,v
      double precision, dimension(dimbr,1)             :: eu2,num2,gr,mleu0,mleu1,mstart,sigmar   ! e(u^2/
           sigma^3|y)
      double precision, dimension(dimb,1)              :: bb
      integer, dimension(dimbr,1)                       :: ej
```

```
double precision, dimension(dimb,1)              :: beta
double precision, dimension(dimb,dimb)           :: i11,i11temp,i22
double precision, dimension(dimb,dimbr)          :: i12
double precision, dimension(dimbr,dimbr)           :: he,hd,hei
double precision, dimension(dimb+dimbr,dimb+dimbr) :: infm
double precision, dimension(s*j,s*j)             :: delta
double precision, parameter                      :: tol = 1.d-12
integer                                          :: con
double precision, PARAMETER                      :: Pi = 3.14159265359

uy = 0
u = 0
sigmar = 0   ! part of sigmar which is corresponding to the random effects
modmatr = 0  ! part of modmat ...
sigmam = 0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! non zeor part of sigma
do i1 = 1,dimbr
  sigmar(i1,1) = sigma(i1,1)
end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! sigmam is diagonal matrix of sigma
do i1 = 1,dimb
   sigmam(i1,i1) = sigma(i1,1)
end do

beta = 0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! generate y
Do I1 = 1,ny        ! number of y
  DO I2=1,(S*J)       !
      CALL RANDOM_NUMBER(u)
      eps(I2,1)=-log(-log(U))
  end do !I2
  do I2 = 1,dimbr
      call random_normal(uy(i2,i1))
      uy(i2,i1) = uy(i2,i1)*sqrt(sigma(i2,1))
      beta(i2,1) = b(i2,1) + uy(i2,i1)
  end do
  if( dimb > dimbr) then
      do i2 = (dimbr+1),dimb
         beta(i2,1) = b(i2,1)
      end do
  end if
  v(:,1)=matmul(modmat,beta(:,1))+eps(:,1) ! utility of one y vector
  y(:,i1)=0
```

```
  do I2=1,S         ! I2 choice set
      ind= (I2-1)*J+1 ! 1st alt in I2, index for the chosen alt
      y(IND,I1)=1     ! current chosen alternative, 1st alt in choice set
      do I3=2,J      ! 2 to Jth alternative
          if(v((I2-1)*J+I3,1)>v(ind,1)) then ! if the alt larger than the current chosen one
          y((I2-1)*J+I3,I1)=1              ! chosen
          y(ind,I1)=0                      ! not chose
          ind=(I2-1)*J+I3                  ! update the index for the chosen one
          end if
      end do !I3,alt
  end do    !I2,choice set
end do      !I1,y

hei = 0
hd = 0
i11 = 0
i11temp = 0
i12 = 0
i22 = 0
nyd = ny
vn = 0
vni = 0
deltad = 0
deltadi = 0
bb = 0

do i1 = 1,dimbr
   modmatr(:,i1) = modmat(:,i1)
end do
! the differenced coded the design matrix madmatd and matmatrd(for random effects)
do i1 =1,s
   do i2 = 1,j-1
       ind1 = (i1-1)*(j-1)+i2
       ind2 = (i1-1)*j+i2
       modmatd(ind1,:) = modmat(ind2,:)-modmat(i1*j,:)
       modmatrd(ind1,:) = modmatr(ind2,:)-modmatr(i1*j,:)
   end do
end do

do i=1,ny
  ! get E(p|y),E(u^2/simga^3|y) from laplace approximation
  ! epy
  !mleu0(:,1) = uy(:,i)         !starting value for u
  mleu0(:,1) = uy(:,i)         !starting value for u
  con = 0                      ! 1 is newton's alg for denominator converges
  j1 = 1                       ! find maximizer u for denominator
  do while (j1<=n)
```

```
  do l1 = 1,dimbr
   bb(l1,1) = b(l1,1) + mleu0(l1,1)
  end do
  if(dimb > dimbr) then
      do l1 = (dimbr+1),dimb
        bb(l1,1) = b(l1,1)
      end do
  end if
  UTY = exp(MATMUL(modmat,bb))
  delta = 0
  DO l1 = 1,S ! the probability of y,
    ind1 = (l1-1)*j+1
    ind2 =  l1*j
    SUMY =  sum(UTY(ind1:ind2,1))
    p(ind1:ind2,1)=UTY(ind1:ind2,1)/SUMY
    ptemp(:,1) = p(ind1:ind2,1)
    delta(ind1:ind2,ind1:ind2) = - matmul(ptemp,transpose(ptemp))
    do l2 = 1,j
        delta(ind1-1+l2,ind1-1+l2) = delta(ind1-1+l2,ind1-1+l2)+p(ind1-1+l2,1)
    end do
  end do
  gr(:,1) = matmul(transpose(modmatr),y(:,i)-p(:,1))- matmul(sigmami,mleu0(:,1))
  he = -matmul(matmul(transpose(modmatr),delta),modmatr)-sigmami
  stepsize=1/(1+10*sqrt(sum(gr(:,1)**2)))
  call solve(dimbr,he,hei)
  mleu1 = mleu0 - stepsize*matmul(hei,gr)
  if( sqrt(sum((mleu1-mleu0)**2))<tol) then
    j1 = n+1
    con = 1
  else
    mleu0 = mleu1
    j1 = j1+1
  end if
end do ! while
if(con==1) then   ! the newton algrithm in the denominator converges
    do i1 =1,s
        ind1 = (i1-1)*(j-1)
        ind2 = (i1-1)*j
        deltad((ind1+1):(ind1+j-1),(ind1+1):(ind1+j-1)) = delta((ind2+1):(ind2+j-1),(ind2+1):(ind2+j
             -1))
    end do
    call solve(s*(j-1),deltad,deltadi)
    vn = deltadi + matmul(matmul(modmatd,sigmam),transpose(modmatd))
    call solve(s*(j-1),vn,vni)
    i11temp = matmul(matmul(transpose(modmatd),vni),modmatd)
    i11 = i11 + i11temp
    i22 = i22 + 2*matmul(matmul((sqrt(sigmam)),i11temp**2),sqrt(sigmam))
```

```
  else
        nyd = nyd -1
  end if
end do ! i
infm(1:dimb,1:dimb) = I11
infm(1:dimb,(dimb+1):(dimb+dimbr)) = I12
infm((dimb+1):(dimb+dimbr),1:dimb) = TRANSPOSE(I12)
infm((dimb+1):(dimb+dimbr),(dimb+1):(dimb+dimbr)) = I22(1:dimbr,1:dimbr)
infm=infm/nyd
return
end subroutine pql
```

6.4 Code for MQL

The following code is for implementing coordinate exchange algorithm with design criteria based on MQL in Section 4.3.

R Code

```
library(DoE.base)
library(MASS)
library(Matrix)

##############################
#          main     141013   #
##############################
S = 10               # number of choice sets
J = 2               # number of alternatives per choice sets
# cs = c(,,,,)        # input vector, the number of alternatives in each choice set, S, J can be read from cs
cs=rep(J,S)           # a vector for numbers of alternatives in choice sets, used
ns=length(cs)         # number of choice sets reading from the length of cs
nl=c(3,3,3,3,3)#,3,3)         # a vector for the number of levels for attributes
#b = c(0,0,0,0)
#sigma = c(1,1,1,1)
b = 3*c(1,0,1,0,1,0,1,0,1,0)
sigma = 9*c(rep(1,8),rep(0,2))

indexr = (1:length(b))[sigma!=0]
indexf = (1:length(b))[sigma==0]
br = b[indexr]
sigmar = sigma[indexr]
```

```
bf = b[indexf]

# b = .5*c(1,0,1,0)
# sigma = 1.5*rep(1,4)
# b = .5*c(1,0,1,0)
# sigma = .25*rep(1,4)
nattr<-length(nl)                # number of attributes
dimb<-sum(nl)-nattr              # number of parameters for the main effects

dimbr <- length(br)
qes<- rep(factor(1:ns),times=cs) # a vector of ids of the alternatives in each choice set.
ind <- c(0,cumsum(cs))


cprob<-function(b,modmat,...)                        # each column of b is a vector of parameter values(nrow
     = dimb), modmat is the coded design matrix (sum(cs)*dimb)
        {
          rep=exp(modmat%*%b)                                # each column of rep is a vector of
               representative utilities of alternatives, nrow = sum(cs)
          sumrep=apply(rep,2,function(x) ave(x,qes,FUN=sum)) # each column of sumrep is a vector of sum of
               utilities in each choice set, nrow = ns
          p=rep/sumrep                                       # each column of p is a vector of choice
               probabilities of alternatives, nrow = ns
          return(p)
         }

delta = matrix(0,S*J,S*J)
ind2 = seq(J,by=J,length=S)

mql = function(modmat)
{
        modmatd = matrix(0,S*(J-1),dimb)
        for(i in 1:S)
                {
                        modmatd[((i-1)*(J-1)+1):(i*(J-1)),] = modmat[((i-1)*J+1):((i-1)*J+J-1),]-modmat[i*J,]
                }
        I = matrix(0,dimb+dimbr,dimb+dimbr)
        I11 = 0
    I22 = 0
        u = 0
        p = cprob(b,modmat) # p evaluated at tilde(u)
        for( j in 1:S)
        {
                st = (j-1)*J+1
                en = j*J
                pos = st:en
                delta[pos,pos]=diag(p[pos,1])-p[pos,1]%*%t(p[pos,1])
```

```
        }
        v = solve(delta[setdiff(1:(S*J),ind2),setdiff(1:(S*J),ind2)])+modmatd%*%diag(sigma)%*%t(modmatd)  #
             vn in the formula
        vi = solve(v)
        vix = vi%*%modmatd
        I11 = I11 + t(modmatd)%*%vix
        I22 =   I22 + 2*diag(sqrt(sigma))%*%(t(modmatd)%*%vi%*%modmatd)^2%*%diag(sqrt(sigma))
        I[1:dimb,1:dimb] = I11
        I[(dimb+1):(dimb+dimbr),(dimb+1):(dimb+dimbr)] = I22[1:dimbr,1:dimbr]
        return(I)
        }

############################################################
#  ordinate exchange alg for generating designs             #
############################################################

do<-Inf
contr=NULL
R=10                                 # number of starting designs for coordinate exchange alg
design<-array(0,c(R,sum(cs),nattr)) # optimal designs from R starting designs
designc<-array(0,c(R,sum(cs),dimb)) # coded values of design
derr <- rep(0,R)                    # the determinant of the R optimal designs
count=0
for(r in 1:R)
  {
         st=NULL
         fullfac=fac.design(nlevels=nl,random=F)
         for(i in 1:S){st=c(st,sample(prod(nl),J,replace=T))}
         designm=(fullfac)[st,]
    if(is.null(contr))
            {
                 contr=rep('contr.sum',nattr)
                 names(contr)=names(designm)
                 contr=as.list(contr)
             }
    modmat=model.matrix(~.,designm,contrasts = contr)[,-1]  #contr is used to get effects type coding,
         current coded design
    Ic=mql(modmat)
    #Ic=round(Ic,digits=10)
        #if(rankMatrix(Infm[,,j])<(2*dimb))
         if((rankMatrix(Ic)<(dimb+dimbr)))
        #if(rankMatrix(Ijt[,,j])<(2*dimb))
        {
                dc = Inf
        }else{
                # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)/(2*dimb)
                # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
```

```
  #dc=abs(det(Ic))^{-1/(2*dimb)} # determinant of the current design
  dc=sum(eigen(solve(Ic))$values)/(dimb+dimbr)
          # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)/(2*dimb)
          # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
          if(dc < 0)
            {
                  dc = Inf
            }
          }
new=matrix()              # new design
##n,n,i,j,k,l
m=1
while(m!=0)               # if no exchange is made, then m=0
  {
     n=0                                  # number of exchange
     for(i in 1:(sum(cs)*nattr))          # i goes through all elements in the uncoded design matrix
             {
                 j=(i%%nattr)                      # column in uncoded design matrix, i.e., jth
                      attribute
                 if(j==0) {j=nattr}
                 k=(i-j)/nattr+1                   # row in uncoded design matrix, i.e., kth row
                 ch=ceiling(k/J)                   # the 'ch'th choice set
                 diff=setdiff(1:nl[j],designm[k,j]) # possible levels for exchange
                 for(l in diff)
                         {
                              new=designm
                              new[k,j]=l                                    # uncoded design
                                   matrix after exchange
                              modmatnew=model.matrix(~.,new,contrasts=contr)[,-1] # coded matrix of new
                                          result1 = fi(modmat)
                                              I1=mql(modmatnew)
                                              #c=round(Ic,digits=10)
                                                    #if(rankMatrix(Infm[,,j])<(2*dimb))
                                                     if((rankMatrix(I1)<(dimb+dimbr)))
                                                    #if(rankMatrix(Ijt[,,j])<(2*dimb))
                                                   {
                                                           d1 = Inf
                                                    }else{
                                                           # a.i[j]=sum(eigen(solve(Infm[,,j]))$values)
                                                                /(2*dimb)
                                                           # d.i[j]=(det(Infm[,,j]))^(-1/(2*dimb))
                                                  #d1 = abs(det(I1))^{-1/(2*dimb)} # determinant of the
                                                       current design
                                                  d1=sum(eigen(solve(I1))$values)/(dimb+dimbr)
                                                           # a.jt[j]=sum(eigen(solve(Ijt[,,j]))$values)
                                                                /(2*dimb)
                                                           # d.jt[j]=(det(Ijt[,,j]))^(-1/(2*dimb))
```

```
                                                    if(d1 < 0)
                                              {
                                               d1 = Inf
                                              }

                                               }
                                    if (d1<dc)
                                       {
                                         designm=new
                                         modmat=modmatnew
                                         Ic=I1
                                         dc=d1
                                         n=n+1       # exchange is kept, add 1 to number of change
                                       }
                                      if(d1 == dc)
                                      {
                                         u = runif(1,0,1)
                                         if(u < 0.5)
                                         {
                                         designm=new
                                         modmat=modmatnew
                                         Ic=I1
                                         dc=d1
                                         n=n+1       # exchange is kept, add 1 to number of change
                                         }
                                      }
                                               count=count+1
                                } #l
                            print(dc)
                        }# i
                    m=n
            }#end of while
          design[r,,]=as.matrix(designm)
          designc[r,,]=modmat
          derr[r]=dc
      if(dc<do)
        {
          Io=Ic        # information matrix of the optimal design
          opt=designm  # optimal design
          mo=modmat    # coded optimal design
          do=dc        # determinant of optimal design
         }
          save.image('rcode.RData')
    }# r

design.R = matrix(0,S*J,nattr*R)
for(i in 1:R)
```

```
{
        design.R[,((i-1)*nattr+1):(i*nattr)]=design[i,,]
}
write.table(design.R,'design.txt',row.names=F,col.names=F)
write.table(opt,'opt.txt',row.names=F,col.names=F)
write.table(do,'do.txt',col.names=F,row.names=F)
write.table(derr,'derr.txt',col.names=F,row.names=F)

save.image('rcode.RData')
```