Minimax D-optimal Designs for Regression Models with Heteroscedastic Errors

by

Kai Yzenbrandt
B.Sc., University of Victoria, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

Minimax D-optimal Designs for Regression Models
with Heteroscedastic Errors

by

Kai Yzenbrandt
B.Sc., University of Victoria, 2018

Supervisory Committee

Dr. Julie Zhou, Supervisor
(Department of Mathematics and Statistics)

Dr. Laura Cowen, Department Member
(Department of Mathematics and Statistics)

Supervisory Committee

Dr. Julie Zhou, Supervisor
(Department of Mathematics and Statistics)

Dr. Laura Cowen, Department Member
(Department of Mathematics and Statistics)

ABSTRACT

Minimax D-optimal designs for regression models with heteroscedastic errors are studied and constructed. These designs are robust against possible misspecification of the error variance in the model. We propose a flexible assumption for the error variance and use a minimax approach to define robust designs. As usual it is hard to find robust designs analytically, since the associated design problem is not a convex optimization problem. However, the minimax D-optimal design problem has an objective function as a difference of two convex functions. An effective algorithm is developed to compute minimax D-optimal designs under the least squares estimator and generalized least squares estimator. The algorithm can be applied to construct minimax D-optimal designs for any linear or nonlinear regression model with heteroscedastic errors. In addition, several theoretical results are obtained for the minimax D-optimal designs.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Consider regression models with heteroscedastic errors,

$$y_i = \mathbf{z}^\top(\mathbf{x}_i)\boldsymbol{\theta} + \epsilon_i, \quad i = 1, \ldots, n, \tag{1.1}$$

where $\mathbf{x} = (x_1, \ldots, x_p)^\top$ is a vector of p independent variables, y_i is the observation on a response variable y at design point $\mathbf{x}_i$ of $\mathbf{x}$, $\mathbf{z}(\mathbf{x})$ is a vector of known functions of $\mathbf{x}$, $\boldsymbol{\theta} \in R^q$ is the unknown regression parameter vector, n is the number of design points, the errors ϵ_i are assumed to be independent with mean zero and variance σ_i^2, and the σ_i^2 may depend on $\mathbf{x}_i$. Let $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^\top$ and $\mathbf{V}_\epsilon = Cov(\boldsymbol{\epsilon})$. Suppose $\hat{\boldsymbol{\theta}}$ is an estimator of $\boldsymbol{\theta}$. Define $C(\hat{\boldsymbol{\theta}}, \xi, \mathbf{V}_\epsilon) = Cov(\hat{\boldsymbol{\theta}})$, where ξ denotes the distribution of design points $\mathbf{x}_1, \cdots, \mathbf{x}_n$.

For a given model, various optimal designs have been constructed by solving the following optimization problem,

$$\min_{\xi \in \Xi} \mathcal{L}\left(C(\hat{\boldsymbol{\theta}}, \xi, \mathbf{V}_\epsilon)\right), \tag{1.2}$$

where $\mathcal{L}$ is a scalar function, such as the determinant or trace function, and Ξ is a set of distributions for $\mathbf{x}$ on a design space S. It is obvious that optimal designs depend on the regression model, function $\mathcal{L}$, the estimator $\hat{\boldsymbol{\theta}}$, the design space S and Ξ, and $\mathbf{V}_\epsilon$. See,

for example, Pukelsheim (2006), Berger and Wong (2009) and Dean et al. (2015) for a wide class of optimality criteria such as D-, A-, c-, or E-optimality and many practical applications of optimal designs. When the scalar function $\mathcal{L}$ is the determinant function then the D-optimality criterion is being used, and when $\mathcal{L}$ is the trace function then the A-optimality criterion is being used. In c-optimality, the form of the optimization is slightly different in that the variance of a linear combination of the estimator $\hat{\boldsymbol{\theta}}$ is minimized. Lastly, the E-optimality criterion is when $\mathcal{L}$ is the function denoting the largest eigenvalue.

The least squares estimator (LSE) and the generalized least squares estimator (GLSE) are often used to estimate $\boldsymbol{\theta}$ in (1.1), and they are given, respectively, by

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{LS} &= \left(\mathbf{Z}^\top \mathbf{Z}\right)^{-1} \mathbf{Z}^\top \mathbf{y}, \\ \hat{\boldsymbol{\theta}}_{G} &= \left(\mathbf{Z}^\top \mathbf{G}^{-1} \mathbf{Z}\right)^{-1} \mathbf{Z}^\top \mathbf{G}^{-1} \mathbf{y}, \end{aligned}$$

where $n \times n$ matrix $\mathbf{G}$ is a diagonal matrix that may be an estimate of the covariance matrix of the errors, and

$$\mathbf{Z} = \begin{pmatrix} \mathbf{z}^\top(\mathbf{x}_1) \\ \vdots \\ \mathbf{z}^\top(\mathbf{x}_n) \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

The covariance matrices of these estimators are, respectively,

$$C(\hat{\boldsymbol{\theta}}_{LS}, \xi, \mathbf{V}_\epsilon) = \left(\mathbf{Z}^\top \mathbf{Z}\right)^{-1} \mathbf{Z}^\top \mathbf{V}_\epsilon \mathbf{Z} \left(\mathbf{Z}^\top \mathbf{Z}\right)^{-1}, \tag{1.3}$$

$$C(\hat{\boldsymbol{\theta}}_{G}, \xi, \mathbf{V}_\epsilon) = \left(\mathbf{Z}^\top \mathbf{G}^{-1} \mathbf{Z}\right)^{-1} \mathbf{Z}^\top \mathbf{G}^{-1} \mathbf{V}_\epsilon \mathbf{G}^{-1} \mathbf{Z} \left(\mathbf{Z}^\top \mathbf{G}^{-1} \mathbf{Z}\right)^{-1}. \tag{1.4}$$

If $\mathbf{G} = \mathbf{V}_\epsilon$ is used in the GLSE, then $C(\hat{\boldsymbol{\theta}}_{G}, \xi, \mathbf{V}_\epsilon) = \left(\mathbf{Z}^\top \mathbf{V}_\epsilon^{-1} \mathbf{Z}\right)^{-1}$ and the GLSE is the best linear unbiased estimator.

When $\mathbf{V}_\epsilon = \sigma^2\mathbf{I}_n$ with $\mathbf{I}_n$ being the identity matrix of order n, model (1.1) has constant error variance σ^2 and optimal designs are easier to construct than those for models with heteroscedastic errors. From (1.2), (1.3) and (1.4), it is clear that optimal designs for models with heteroscedastic errors usually depend on $\mathbf{V}_\epsilon$. Solving the optimization problem in (1.2) is hard in general, but various optimal designs have been investigated and constructed in the literature including Dette and Wong (1999), Dette et al. (1999), Wu (2007), Kitsos et al. (2006), Martínez-López et al. (2009), Papp (2012), Wiens and Li (2014), Yan et al. (2017), and Zhai and Fang (2018). Those optimal designs are obtained under the assumption that $\mathbf{V}_\epsilon$ is known up to a constant factor, and some results are for nonlinear regression models. Several results also include a sensitivity analysis on how small changes of $\mathbf{V}_\epsilon$ affect optimal designs.

In practice $\mathbf{V}_\epsilon$ is never known. Thus, there is a need to investigate robust designs against possible misspecification of $\mathbf{V}_\epsilon$. We propose a flexible assumption for $\mathbf{V}_\epsilon$ and use a minimax D-optimality criterion for constructing robust designs for linear and nonlinear regression models. In particular, we assume that $\mathbf{V}_\epsilon$ can vary in a neighbourhood of covariance matrices. This neighbourhood idea has been used for robust designs against misspecification in the regression response function and error correlation structures; see Wiens (2015) for details. However, our minimax D-optimality criterion and design problems in Chapter 2 are different from those in Wiens (2015). Recently Gao and Zhou (2020) also used a minimax approach to investigate robust designs for multi-response models, which are robust against the misspecification of the covariance matrix of the response variables. In this thesis we focus on single response models with heteroscedastic errors. We derive new theoretical results for minimax D-optimal designs, and our methods can be applied for any linear or nonlinear regression model to find minimax D-optimal designs easily. Since the minimax design problems are not convex optimization problems, we will develop a new algorithm for computing minimax designs, which is based on an algorithm for solving optimization problems with

objective function being the difference of two convex functions (Lipp and Boyd, 2016). Our algorithm is effective for finding minimax designs if we carefully select some initial values for the algorithm, and the algorithm is fairly fast with the initial values we examined. We have results for both the LSE and GLSE, and minimax D-optimal designs can be found for any discrete design space.

The rest of the thesis is organized as follows. In Chapter 2 we introduce a minimax D-optimality criterion for constructing robust designs for model (1.1) and investigate its theoretical properties. In Chapter 3 we propose an effective numerical algorithm to compute minimax D-optimal designs, and we also discuss the minimax D-optimality criterion for nonlinear regression models. Applications of minimax D-optimal designs are given in Chapter 4. Concluding remarks are in Chapter 5. Matlab code for the problem independent portion of the algorithm is presented in the Appendix.

The main contributions in this thesis are:

1. We have studied a minimax D-optimal design criterion for regression models with heteroscedastic errors and derived various theoretical properties of these designs.
2. We have proposed an effective algorithm to solve this D-optimal design problem and examined various theoretical properties of the algorithm.
3. We have implemented the algorithm in Matlab and demonstrated its use via linear and nonlinear examples.

A research paper, Yzenbrandt and Zhou (2020), based on this thesis has been submitted for publication in *MetriKa*.

Chapter 2

Minimax D-optimality criterion and its properties

The covariance matrices of the LSE and GLSE in (1.3) and (1.4) involve the true covariance matrix $\mathbf{V}_\epsilon$ of the errors. To deal with the uncertainty of $\mathbf{V}_\epsilon$, we propose a flexible error assumption in Section 2.1. In Section 2.2 we define a minimax D-optimality criterion for constructing robust regression designs. Theoretical properties of the criterion are derived and presented in Section 2.3.

2.1 Error variance assumption

From model (1.1), $\mathbf{V}_\epsilon$ is a diagonal matrix with diagonal elements σ_i^2. Let $\sigma_i^2 = \sigma^2\lambda(\mathbf{x}_i)$ for some positive function λ. For an example with a single independent variable, Dette et al. (1999) studied D-optimal designs for the following polynomial models with heteroscedastic errors,

$$y_i = \theta_0 + \theta_1 x^1 + \cdots + \theta_p x^p + \epsilon, \quad \text{with } \lambda(x) = 1/(1+x^2)^{-v}, \tag{2.1}$$

where v is a positive number, and x belongs to an interval such as $S = [-1, 1]$. In Papp (2012), model (2.1) is used to construct an A-optimal design with $p = 3$, $v = 1$ and $S = [-5, 5]$. Wu (2007) investigated D-optimal designs for segmented polynomial models with heteroscedastic errors and a single independent variable, and the error variance has $\lambda(x) = \exp(x)$. Wiens and Li (2014) constructed V-optimal designs for heteroscedastic linear regression and the error variances σ_i^2 do not follow any analytical function, but σ_i^2 are pre-specified numbers on a set of finite points of $\mathbf{x}$ ($\in S$) or σ_i^2 can be unknown.

For practical applications, we do not know $\lambda(\mathbf{x})$ or σ_i^2 exactly, but we may have some information about them from subject matter knowledge or from a pilot study. To deal with the uncertainty of the error variances, we model them with a flexible assumption as follows,

$$\begin{cases} \sigma_i^2 = \sigma^2 \lambda(\mathbf{x}_i), \quad i = 1, \ldots, n, \\ |\lambda(\mathbf{x}_i) - g(\mathbf{x}_i)| \leq \alpha, \end{cases} \tag{2.2}$$

where $g(\mathbf{x})$ is a prespecifed function, and positive parameter α reflects the magnitude of the uncertainty. Function $g(\mathbf{x})$ can be viewed as an estimate (prior information) of the error variance, and α is an error bound of the estimation. If $\alpha = 0$, $\lambda(\mathbf{x}) = g(\mathbf{x})$ is known exactly. If α is large, it indicates a lot of uncertainty in $\lambda(\mathbf{x})$.

2.2 Minimax D-optimality criterion

In order to develop a general procedure for finding minimax designs, we consider a design space S_N with N discrete design points, and let $S_N = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\} \subset S \subset R^p$. Here N can be a huge number, such as $N = 10,000$ in Example 1 in Chapter 4. This discrete design space S_N is a very reasonable choice for many practical experiments. A distribution ξ on S_N

is denoted by

$$\xi(\mathbf{w}) = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_N \\ w_1 & w_2 & \cdots & w_N \end{pmatrix},$$

where $w_1, \ldots, w_N$ are weights with $w_i \geq 0$ and $\sum_{i=1}^N w_i = 1$, and weight vector $\mathbf{w} = (w_1, \ldots, w_N)^\top$. If weight $w_j > 0$, then $\mathbf{u}_j$ is a support point of ξ. Design points $\mathbf{x}_1, \cdots, \mathbf{x}_n$ are selected from ξ. Define matrices

$$\mathbf{A}_i = \mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i), \quad \mathbf{B}_i = \lambda(\mathbf{u}_i)\mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i),$$

$$\mathbf{D}_i = \frac{1}{g(\mathbf{u}_i)} \cdot \mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i), \quad \mathbf{H}_i = \frac{\lambda(\mathbf{u}_i)}{g^2(\mathbf{u}_i)}\mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i),$$

and

$$\mathbf{C}_1(\mathbf{w}, \lambda) = \left(\sum_{i=1}^N w_i \mathbf{A}_i\right)^{-1} \left(\sum_{i=1}^N w_i \mathbf{B}_i\right) \left(\sum_{i=1}^N w_i \mathbf{A}_i\right)^{-1}, \tag{2.3}$$

$$\mathbf{C}_2(\mathbf{w}, \lambda) = \left(\sum_{i=1}^N w_i \mathbf{D}_i\right)^{-1} \left(\sum_{i=1}^N w_i \mathbf{H}_i\right) \left(\sum_{i=1}^N w_i \mathbf{D}_i\right)^{-1}. \tag{2.4}$$

From (1.3) and (1.4), the covariance matrix of the LSE is proportional to $\mathbf{C}_1(\mathbf{w}, \lambda)$, and that of the GLSE with $\mathbf{G} = \text{diag}(g(\mathbf{x}_1), \ldots, g(\mathbf{x}_n))$ is proportional to $\mathbf{C}_2(\mathbf{w}, \lambda)$. Since $g(\mathbf{x}_i)$ is an estimate of $\lambda(\mathbf{x}_i)$, it is natural to use it in $\mathbf{G}$ to compute the GLSE. In the following we focus on $\mathbf{C}_1(\mathbf{w}, \lambda)$ and $\mathbf{C}_2(\mathbf{w}, \lambda)$ to define a minimax D-optimality criterion.

Definition 1. Design $\xi(\mathbf{w}^*)$ is called a minimax D-optimal design under the LSE if $\mathbf{w}^*$ is a solution to the following optimization problem

$$\begin{cases} \min_{\mathbf{w}} \ \max_{\lambda \in \Lambda(\alpha)} \log(\det(\mathbf{C}_1(\mathbf{w}, \lambda))) \\ \text{subject to: } w_i \geq 0, \ \sum_{i=1}^N w_i = 1, \end{cases} \tag{2.5}$$

where $\Lambda(\alpha) = \{\lambda : \ |\lambda(\mathbf{u}_i) - g(\mathbf{u}_i)| \leq \alpha, \ \text{for all } i = 1, \ldots, N\}$ from the assumption in (2.2).

Similarly, a design $\xi(\mathbf{w}^*)$ is called a minimax D-optimal design under the GLSE if $\mathbf{w}^*$ is a solution to problem (2.5) with $\mathbf{C}_1(\mathbf{w}, \lambda)$ being replaced by $\mathbf{C}_2(\mathbf{w}, \lambda)$ in the objective function.

2.3 Properties

We derive several theoretical properties of minimax D-optimal designs, which are helpful for developing an effective algorithm for finding those designs. Notice that (2.5) is a minimax optimization problem. Firstly we work on the maximization step and let

$$\phi_1(\mathbf{w}) = \max_{\lambda \in \Lambda(\alpha)} \log(\det(\mathbf{C}_1(\mathbf{w}, \lambda))), \quad \phi_2(\mathbf{w}) = \max_{\lambda \in \Lambda(\alpha)} \log(\det(\mathbf{C}_2(\mathbf{w}, \lambda))).$$

Theorem 1. *For $\mathbf{C}_1(\mathbf{w}, \lambda)$ and $\mathbf{C}_2(\mathbf{w}, \lambda)$ defined in (2.3) and (2.4), we obtain*

$$\phi_1(\mathbf{w}) = -2\log\left(\det\left(\sum_{i=1}^{N} w_i \mathbf{A}_i\right)\right) + \log\left(\det\left(\sum_{i=1}^{N} w_i (g(\mathbf{u}_i) + \alpha)\mathbf{A}_i\right)\right), \tag{2.6}$$

$$\phi_2(\mathbf{w}) = -2\log\left(\det\left(\sum_{i=1}^{N} w_i \mathbf{D}_i\right)\right) + \log\left(\det\left(\sum_{i=1}^{N} w_i \frac{(g(\mathbf{u}_i) + \alpha)}{g(\mathbf{u}_i)}\mathbf{D}_i\right)\right). \tag{2.7}$$

Furthermore, $\phi_1(\mathbf{w})$ is a difference of two convex functions of $\mathbf{w}$, as is $\phi_2(\mathbf{w})$.

Proof of Theorem 1: For any $\lambda \in \Lambda(\alpha)$, we have

$$\lambda(\mathbf{u}_i) \leq g(\mathbf{u}_i) + \alpha, \quad \text{for all } i = 1, \ldots, N.$$

Notice that $\mathbf{B}_i = \lambda(\mathbf{u}_i)\mathbf{A}_i$ and $\mathbf{A}_i$ are positive semidefinite for all $i = 1, \ldots, N$. In the following notation $\succeq$ denotes Loewner order for positive semidefinite matrices, where $\mathbf{A} \succeq \mathbf{B}$

indicates that $\mathbf{A}-\mathbf{B}$ is a positive semidefinite matrix. From (2.3) we get

$$\begin{aligned}\mathbf{C}_1(\mathbf{w},\lambda) &= \left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right)^{-1}\left(\sum_{i=1}^{N} w_i\mathbf{B}_i\right)\left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right)^{-1} \\ &\preceq \left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right)^{-1}\left(\sum_{i=1}^{N} w_i(g(\mathbf{u}_i)+\alpha)\mathbf{A}_i\right)\left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right)^{-1}, \quad \text{for all } \lambda\in\Lambda(\alpha),\end{aligned}$$

this implies using

$$\lambda(\mathbf{u}_i) = g(\mathbf{u}_i)+\alpha, \quad \text{for all } i=1,\ldots,N,$$

dominates all other choices of $\lambda(\mathbf{u}_i)$ which leads to the result in (2.6), i.e.,

$$\begin{aligned}\phi_1(\mathbf{w}) &= \max_{\lambda\in\Lambda(\alpha)} \log(\det(\mathbf{C}_1(\mathbf{w},\lambda))) \\ &= -2\log(\det\left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right)) + \log(\det\left(\sum_{i=1}^{N} w_i(g(\mathbf{u}_i)+\alpha)\mathbf{A}_i\right)).\end{aligned}$$

From Boyd and Vandenberghe (2004, page 387), both $-2\log(\det\left(\sum_{i=1}^{N} w_i\mathbf{A}_i\right))$ and $-\log(\det\left(\sum_{i=1}^{N} w_i(g(\mathbf{u}_i)+\alpha)\mathbf{A}_i\right))$ are convex functions of $\mathbf{w}$, so $\phi_1(\mathbf{w})$ is a difference of two convex functions of $\mathbf{w}$. The result for $\phi_2(\mathbf{w})$ in (2.7) can be proved similarly. □

The analytical results in Theorem 1 are very useful for solving problem (2.5), since we can transform the minimax optimization problem into a minimization problem below,

$$\begin{cases} \min_{\mathbf{w}} \phi(\mathbf{w}) \\ \text{subject to: } w_i \geq 0, \ \sum_{i=1}^{N} w_i = 1, \end{cases} \tag{2.8}$$

where $\phi(\mathbf{w})$ is equal to $\phi_1(\mathbf{w})$ or $\phi_2(\mathbf{w})$. Problem (2.8) looks like a classical D-, A-, c-, or E-optimal design problem on a discrete design space (Ye et al., 2017; Wong et al., 2019). However, there is a major difference in the objective functions. In the classical optimal design problem the objective function is a convex function of $\mathbf{w}$, while the objective function

in problem (2.8) is a difference of two convex functions of $\mathbf{w}$. Thus, it is more difficult to solve problem (2.8) for finding minimax D-optimal designs.

Next we will find a necessary condition for $\mathbf{w}^*$ to be a solution to problem (2.8). When $\alpha = 0$, from Theorem 1 $\phi_2(\mathbf{w})$ becomes a convex function of $\mathbf{w}$ and the minimax D-optimal design under the GLSE is the classical D-optimal design, but $\phi_1(\mathbf{w})$ is still a difference of two convex functions and not a convex function of $\mathbf{w}$. In the classical optimal design theory we have a sufficient and necessary condition to check for optimal designs. Since $\phi(\mathbf{w})$ is not convex here, we can only obtain a necessary condition for $\mathbf{w}^*$ to be a solution to problem (2.8). Define

$$\begin{aligned}
\mathbf{A}(\mathbf{w}) &= \sum_{i=1}^{N} w_i \mathbf{A}_i, \\
\mathbf{A}_g(\mathbf{w}) &= \sum_{i=1}^{N} w_i (g(\mathbf{u}_i) + \alpha) \mathbf{A}_i, \\
\mathbf{D}(\mathbf{w}) &= \sum_{i=1}^{N} w_i \mathbf{D}_i, \\
\mathbf{D}_g(\mathbf{w}) &= \sum_{i=1}^{N} w_i \frac{(g(\mathbf{u}_i) + \alpha)}{g(\mathbf{u}_i)} \mathbf{D}_i, \\
d_1(\mathbf{u}_i) &= \text{trace}\left(\left[2\mathbf{A}^{-1}(\mathbf{w}^*) - (g(\mathbf{u}_i) + \alpha) \mathbf{A}_g^{-1}(\mathbf{w}^*) \right] \mathbf{A}_i \right) - q, \qquad (2.9) \\
d_2(\mathbf{u}_i) &= \text{trace}\left(\left[2\mathbf{D}^{-1}(\mathbf{w}^*) - \frac{(g(\mathbf{u}_i) + \alpha)}{g(\mathbf{u}_i)} \mathbf{D}_g^{-1}(\mathbf{w}^*) \right] \mathbf{D}_i \right) - q, \qquad (2.10)
\end{aligned}$$

where $q \geq 0$.

Theorem 2. *If $\mathbf{w}^*$ is a solution to problem (2.8) with $\phi(\mathbf{w}) = \phi_1(\mathbf{w})$, then it satisfies*

$$d_1(\mathbf{u}_i) \leq 0, \quad \textit{for all} \quad i = 1, \ldots, N.$$

If $\mathbf{w}^*$ is a solution to problem (2.8) with $\phi(\mathbf{w}) = \phi_2(\mathbf{w})$, then it satisfies

$$d_2(\mathbf{u}_i) \leq 0, \quad \textit{for all} \quad i = 1, \ldots, N.$$

Proof of Theorem 2: Suppose $\mathbf{w}^*$ is a solution to problem (2.8) with $\phi(\mathbf{w}) = \phi_1(\mathbf{w})$. Let $\mathbf{w}$ be any weight vector and let $\mathbf{w}_\delta = (1-\delta)\mathbf{w}^* + \delta\mathbf{w}$ for $\delta \in [0,1]$, so $\mathbf{w}_\delta$ is also a weight vector. Since $\phi_1(\mathbf{w})$ is minimized at $\mathbf{w}^*$, we must have

$$\frac{\partial \phi_1(\mathbf{w}_\delta)}{\partial \delta} \mid_{\delta=0} \geq 0, \text{ for any } \mathbf{w}.$$

Straightforward computation of the derivative leads to

$$\sum_{i=1}^{N} w_i \cdot \text{trace}\left(\left[2\mathbf{A}^{-1}(\mathbf{w}^*) - (g(\mathbf{u}_i)+\alpha)\mathbf{A}_g^{-1}(\mathbf{w}^*)\right]\mathbf{A}_i\right) \leq q, \quad \text{for any } \mathbf{w},$$

which gives the necessary condition,

$$d_1(\mathbf{u}_i) = \text{trace}\left(\left[2\mathbf{A}^{-1}(\mathbf{w}^*) - (g(\mathbf{u}_i)+\alpha)\mathbf{A}_g^{-1}(\mathbf{w}^*)\right]\mathbf{A}_i\right) - q \leq 0, \quad \text{for all } i = 1, \ldots, N.$$

Similarly we can prove the result when $\phi(\mathbf{w}) = \phi_2(\mathbf{w})$. □

The result in Theorem 2 is not a sufficient condition, since a locally optimal solution can also satisfy the condition.

Since it is hard (impossible) to solve the minimization problem (2.8) analytically, we develop a numerical algorithm to compute $\mathbf{w}^*$ in the next chapter.

Chapter 3

Computing minimax D-optimal designs

We present an effective algorithm in Section 3.1, and some theoretical results of the algorithm are also derived. In Section 3.2 we discuss the computation of minimax D-optimal designs for nonlinear regression models.

3.1 Numerical algorithm

To present a general algorithm for solving problem (2.8) with $\phi(\mathbf{w}) = \phi_1(\mathbf{w})$ or $\phi_2(\mathbf{w})$, we write $\phi(\mathbf{w}) = v_1(\mathbf{w}) - v_2(\mathbf{w})$, where $v_1(\mathbf{w})$ and $v_2(\mathbf{w})$ are two convex functions. Using the first-order approximation for $v_2(\mathbf{w})$ around a weight vector $\mathbf{w}^{(0)}$, we define

$$\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)}) = v_1(\mathbf{w}) - \left[v_2(\mathbf{w}^{(0)}) + \frac{\partial v_2(\mathbf{w})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^{(0)}} \left(\mathbf{w} - \mathbf{w}^{(0)} \right) \right], \tag{3.1}$$

so $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)})$ is an approximation of $\phi(\mathbf{w})$. For fixed $\mathbf{w}^{(0)}$, $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)})$ is the sum of a convex and a linear function of $\mathbf{w}$, so $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)})$ is a convex function of $\mathbf{w}$. The derivative $\frac{\partial v_2(\mathbf{w})}{\partial \mathbf{w}^\top}$ in (3.1) is given below.

The derivative $\frac{\partial v_2(\mathbf{w})}{\partial \mathbf{w}^\top}$: For $\phi(\mathbf{w}) = \phi_1(\mathbf{w})$, we have

$$v_2(\mathbf{w}) = -\log\left[\det\left(\sum_{i=1}^{N} w_i[g(\mathbf{u}_i)+\alpha]\mathbf{A}_i\right)\right],$$

and for $i = 1, \ldots, N$,

$$\frac{\partial v_2(\mathbf{w})}{\partial w_i} = -\operatorname{trace}\left([g(\mathbf{u}_i)+\alpha]\mathbf{A}_g^{-1}(\mathbf{w})\mathbf{A}_i\right).$$

Similarly, for $\phi(\mathbf{w}) = \phi_2(\mathbf{w})$ we get

$$\frac{\partial v_2(\mathbf{w})}{\partial w_i} = -\operatorname{trace}\left(\frac{[g(\mathbf{u}_i)+\alpha]}{g(\mathbf{u}_i)}\mathbf{D}_g^{-1}(\mathbf{w})\mathbf{D}_i\right), \quad i = 1, \ldots, N.$$

□

The key idea of our algorithm is from Lipp and Boyd (2016), and we consider and solve the following convex optimization problem,

$$\begin{cases} \min_{\mathbf{w}} \tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)}) \\ \text{subject to: } w_i \geq 0, \ \sum_{i=1}^{N} w_i = 1. \end{cases} \tag{3.2}$$

We use the limit of a sequence of solutions to problem (3.2) to approximate the solution to problem (2.8). The main steps of our algorithm are given below.

Algorithm 1: Find a solution to problem (2.8).

Step (i): For a given model and S_N, compute $\mathbf{A}_i$ (or $\mathbf{D}_i$) and $g(\mathbf{u}_i)$ for $i = 1, \ldots, N$.

Step (ii): Let $\mathbf{w}^{(0)}$ be an initial weight vector and let η be a small positive number used in the stopping criterion in Step (iii), say $\eta = 10^{-4}$.

Step (iii): For $j = 1, 2, \cdots$, do the following iterations,

(iii.1) find the solution to problem (3.2) with objective function being replaced by $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(j-1)})$ for each j and denote the solution as $\mathbf{w}^{(j)}$,

(iii.2) stop the iteration if $\max_{1\le i\le N} |w_i^{(j)} - w_i^{(j-1)}| \le \eta$.

Step (iv): Let $\mathbf{w}^* = \mathbf{w}^{(j)}$ at the last iteration in Step (iii), and $\mathbf{w}^*$ is a solution to problem (2.8).

We have several remarks for Algorithm 1. The convergence of the algorithm depends on the regression model, design space S_N, the value of α, and the initial weight vector $\mathbf{w}^{(0)}$. For each practical application, we may not be able to control the regression model and design space S_N, but we can choose $\mathbf{w}^{(0)}$ carefully. If $\mathbf{w}^{(0)}$ is close to the solution of problem (2.8), then $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(0)})$ provides a good approximation to $\phi(\mathbf{w})$ in the neighbourhood of the solution. One choice is to let $\mathbf{w}^{(0)}$ be the miminizer of $v_1(\mathbf{w})$, and it works well for most applications. Additional comments about $\mathbf{w}^{(0)}$ are given in Chapter 4.

How do we find the miminizer of $v_1(\mathbf{w})$? We need to solve problem (3.2) with the objective function being replaced by $v_1(\mathbf{w})$. Notice that $v_1(\mathbf{w})$ is a convex function of $\mathbf{w}$. There are several fast algorithms for solving constrained convex optimization problems in Boyd and Vandenberghe (2004), and we use a MATLAB solver called CVX (Grant and Boyd, 2013) to get $\mathbf{w}^{(0)}$ and the solutions $\mathbf{w}^{(j)}$ to problem (3.2) in Step (iii.1) of Algorithm 1. Since the CVX program is fairly fast and can find solutions for large N, Algorithm 1 is also fairly fast to compute the solution to problem (2.8).

If $\lim_{j\to\infty} \mathbf{w}^{(j)} = \mathbf{w}^*$ in Step (iii) of Algorithm 1, then we use Theorem 2 to see if $\mathbf{w}^*$ satisfies the necessary condition. Alternatively we can use the necessary condition as a stopping criterion in Step (iii.2) of Algorithm 1. For numerical results, the necessary condition is implemented as follows, for all $i = 1, \ldots, N$,

$$d_1(\mathbf{u}_i) \le \eta_0, \tag{3.3}$$

or

$$d_2(\mathbf{u}_i) \leq \eta_0, \tag{3.4}$$

for a small positive η_0, where $d_1(\mathbf{u}_i)$ and $d_2(\mathbf{u}_i)$ are defined in (2.9) and (2.10), respectively.

The connection between the solutions of (2.8) and (3.2) is obtained in the following theorem, and its proof is shown below.

Theorem 3. *If* $\lim_{j\to\infty} \mathbf{w}^{(j)} = \mathbf{w}^*$ *in Step (iii) of Algorithm 1, then we have*

$$\lim_{j\to\infty} \tilde{\phi}(\mathbf{w}^{(j)}, \mathbf{w}^{(j-1)}) = \phi(\mathbf{w}^*), \tag{3.5}$$

$$\lim_{j\to\infty} \frac{\partial \tilde{\phi}(\mathbf{w}, \mathbf{w}^{(j-1)})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^{(j)}} = \frac{\partial \phi(\mathbf{w})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^*} . \tag{3.6}$$

In addition, $\mathbf{w}^*$ *has the same necessary condition to be an optimum as in Theorem 2.*

Proof of Theorem 3: From (3.1) and $\lim_{j\to\infty} \mathbf{w}^{(j)} = \mathbf{w}^*$, we get

$$\lim_{j\to\infty} \tilde{\phi}(\mathbf{w}^{(j)}, \mathbf{w}^{(j-1)}) = v_1(\mathbf{w}^*) - v_2(\mathbf{w}^*) = \phi(\mathbf{w}^*),$$

and

$$\lim_{j\to\infty} \frac{\partial \tilde{\phi}(\mathbf{w}, \mathbf{w}^{(j-1)})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^{(j)}} = \lim_{j\to\infty} \left(\frac{\partial v_1(\mathbf{w})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^{(j)}} - \frac{\partial v_2(\mathbf{w})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^{(j-1)}} \right) = \frac{\partial \phi(\mathbf{w})}{\partial \mathbf{w}^\top} \mid_{\mathbf{w}=\mathbf{w}^*},$$

which are the results in (3.5) and (3.6). To show that $\mathbf{w}^*$ requires the necessary condition from Theorem 2, we use the method in the proof of Theorem 2 and let $\mathbf{w}_\delta = (1-\delta)\mathbf{w}^{(j)} + \delta\mathbf{w}$ for $\delta \in [0, 1]$. Then we have

$$\frac{\partial \tilde{\phi}(\mathbf{w}_\delta, \mathbf{w}^{(j-1)})}{\partial \delta} \mid_{\delta=0} \geq 0, \text{ for any } \mathbf{w},$$

and computing the derivatives gives:

$$\frac{\partial\tilde{\phi}(\mathbf{w}_\delta, \mathbf{w}^{(j-1)})}{\partial\delta} \mid_{\delta=0} = \sum_{i=1}^{N} \Big({w_i}^{(j-1)} \cdot \text{trace}\Big\{ -2\mathbf{A}^{-1}(\mathbf{w}^{(j-1)}) + [g(\mathbf{u}_i) + \alpha]\mathbf{A}_g^{-1}(\mathbf{w}^{(j-1)})\mathbf{A}_i \Big\}\Big) + q$$

which leads to the necessary condition in Theorem 2 as $j \to \infty$. □

Now we explore the symmetry of minimax D-optimal designs. To focus on the idea, let $p = 1$ and let S_N be symmetric about 0.

Theorem 4. *Consider a regression model with a symmetric S_N, and let $\lim_{j\to\infty} \mathbf{w}^{(j)} = \mathbf{w}^*$ in Step (iii) of Algorithm 1. If the model satisfies the following conditions:*

(i) $g(-\mathbf{u}_i) = g(\mathbf{u}_i)$ for all $\mathbf{u}_i \in S_N$,

(ii) there exists a constant matrix $\mathbf{Q}$ with $\det(\mathbf{Q}) = 1$ such that $\mathbf{z}(-\mathbf{u_i}) = \mathbf{Q}\mathbf{z}(\mathbf{u_i})$ for all $\mathbf{u}_i \in S_N$,

then $\xi(\mathbf{w}^*)$ is symmetric on the S_N.

Proof of Theorem 4: First we show that for each j, $\xi(\mathbf{w}^{(j)})$ is symmetric on S_N. For any distribution $\xi(\mathbf{w})$ on the S_N, we define $\xi(\tilde{\mathbf{w}})$ to be the following distribution,

$$\begin{pmatrix} -\mathbf{u}_1 & -\mathbf{u}_2 & \cdots & -\mathbf{u}_N \\ w_1 & w_2 & \cdots & w_N \end{pmatrix}.$$

If $\xi(\mathbf{w}) = \xi(\tilde{\mathbf{w}})$, then $\xi(\mathbf{w})$ is symmetric on the S_N. Otherwise, the convex combination $0.5\xi(\mathbf{w}) + 0.5\xi(\tilde{\mathbf{w}})$ is symmetric on the S_N. Under the two conditions in Theorem 4, it is easy to verify that $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(j-1)}) = \tilde{\phi}(\tilde{\mathbf{w}}, \tilde{\mathbf{w}}^{(j-1)})$, if $\xi(\mathbf{w}^{(0)})$ is symmetric. Note that the choice of $\mathbf{w}^{(0)}$ in the algorithm always makes $\xi(\mathbf{w}^{(0)})$ symmetric. Since $\tilde{\phi}(\mathbf{w}, \mathbf{w}^{(j-1)})$ is a convex function of $\mathbf{w}$, the solution $\mathbf{w}^{(j)}$ makes $\xi(\mathbf{w}^{(j)})$ symmetric on the S_N. Since $\xi(\lim_{j\to\infty} \mathbf{w}^{(j)}) = \xi(\mathbf{w}^*)$, $\xi(\mathbf{w}^*)$ is also symmetric on the S_N. □

For $p > 1$, we can look at the symmetry property of $\xi(\mathbf{w}^*)$ with respect to a particular (or each) design variable in $\mathbf{x}$, and the conditions can be derived similarly to those in Theorem 4 to check for the symmetry assuming that S_N is symmetric with respect to that (or each) design variable. For example, if we consider the symmetry property with respect to variable x_1, in the two conditions of Theorem 4 we replace $-\mathbf{u}_i$ by $T_1\mathbf{u}_i$, where T_1 is a diagonal matrix and its diagonal elements are $-1, 1, \ldots, 1$. $T_1\mathbf{u}_i$ can be viewed as the reflection transformation with respect to variable x_1. In applications with multiple design variables, $\xi(\mathbf{w}^*)$ can be symmetric with respect to several variables.

3.2 Extension to nonlinear models

So far we have studied minimax D-optimal designs for linear models, but the methodology can be applied for nonlinear regression models. Here we present the procedure for finding minimax D-optimal designs for a nonlinear regression model,

$$y = F(\mathbf{x}, \boldsymbol{\theta}) + \epsilon, \tag{3.7}$$

where $V(\epsilon) = \sigma^2\lambda(\mathbf{x})$, and $\mathbf{x} \in S_N$. Let $\mathbf{z}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*)$ with

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) = \frac{\partial F(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$$

and $\boldsymbol{\theta}^*$ is the true parameter value of $\boldsymbol{\theta}$. Define matrices, for $i = 1, \ldots, N$,

$$\begin{aligned} \mathbf{A}_i &= \mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i) = \mathbf{f}(\mathbf{u}_i, \boldsymbol{\theta}^*)\mathbf{f}^\top(\mathbf{u}_i, \boldsymbol{\theta}^*), \\ \mathbf{D}_i &= \frac{1}{g(\mathbf{u}_i)}\mathbf{z}(\mathbf{u}_i)\mathbf{z}^\top(\mathbf{u}_i) = \frac{1}{g(\mathbf{u}_i)}\mathbf{f}(\mathbf{u}_i, \boldsymbol{\theta}^*)\mathbf{f}^\top(\mathbf{u}_i, \boldsymbol{\theta}^*). \end{aligned}$$

Putting those matrices in $\phi_1(\mathbf{w})$ and $\phi_2(\mathbf{w})$ defined in (2.6) and (2.7), we can use Algorithm 1 to solve problem (2.8) for finding minimax D-optimal designs for model (3.7).

From the above procedure, it is clear that the minimax D-optimal designs for (3.7) often depend on the true parameter value $\boldsymbol{\theta}^*$, so they are called locally minimax D-optimal designs. In Chapter 4 we give one example to illustrate the procedure and present some locally minimax D-optimal designs.

Chapter 4

Applications

We give representative examples to show minimax D-optimal designs, computation times and sensitivity analysis of minimax D-optimal designs, and address a few numerical issues.

In Section 4.1 we present two examples for linear models. In Example 1 we use a linear model to show the computation times for various values of N, the reflection symmetry, and the changes in minimax D-optimal designs as α changes. In Example 2 we have a mixture experiment with several independent variables, and we compare the minimax D-optimal designs under both the LSE and GLSE. In Section 4.2 we give an example for a nonlinear regression model. All the compuation is done on a laptop equipped with an AMD Ryzen 7 2700U Four core 8 thread CPU with 2.20 GHz base clock, and 16GB 2400 MHz RAM.

4.1 Linear regression models

Example 1. Consider a linear model in Papp (2012) given by (2.1) with $p = 3,\ v = 1$ and $S = [-5, 5]$, where Papp (2012) studied A-optimal designs for this model using semi-definite programming in convex optimization. The model then becomes

$$y_i = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \epsilon, \quad \text{with } \lambda(x) = (1 + x^2).$$

Using Algorithm 1 we compute minimax D-optimal designs with $g(x) = (1+x^2)$ on S_N, which contains N equally spaced points in $[-5, 5]$, i.e., $u_i = -5 + 10(i-1)/(N-1)$, $i = 1, \ldots, N$. The general Matlab functions for Algorithm 1 are given in the Appendix. By calling those functions using the following code we can find minimax D-optimal designs under the GLSE:

```
%beginning end and size of search space
a = -5; b = 5; N = 1001;
u = zeros(1,N);  %Points in the search space
for i = 1:N
    u(1,i) = a+(i-1)*(b-a)/(N-1);
end
%describe size of information matrix, alpha and stopping criteria
q = 4; alpha = 5;
deltaw = 1e-4; delta_equiv = 1e-6; maxiter = 50;
%setup information matrices
infomat = zeros(q,q,N);
inforob = zeros(q,q,N);
for i=1:N
            V = (1+(u(1,i)^2));
            F = [1 u(1,i) u(1,i)^2 u(1,i)^3];
            infomat(:,:,i) = F'*F/V;
            inforob(:,:,i) = F'*F*((V+alpha)/V^2);
end
%obtain initial and iterative weights, see code in appendix for details
w0=initialw(infomat);
[iter,determ,equiv,time,wn] = RobDopt_Diagnostic(w0,infomat,inforob,
   maxiter,deltaw,delta_equiv);
%use a minimum weight (0.0009 here) to get design from weights
design = [u(wn(iter,:)>0.0009)' wn(iter,wn(iter,:)>0.0009)']';
```

Figure 4.1 shows the computation times of minimax D-optimal designs under the LSE and GLSE for $\alpha = 0.5$ with various values of N. For the GLSE, the initial weight vector

is the one suggested in Section 3.1. For the LSE, two initial weight vectors are used. LSE Design 1 in Figure 4.1 shows the computation time using the initial weight vector suggested in Section 3.1, while LSE Design 2 shows the result using the initial weight vector for the GLSE. It is clear that Algorithm 1 is very fast and can find minimax D-optimal designs easily for $N \leq 10,000$ for this model. It also indicates that finding minimax D-optimal designs under the GLSE is faster than that under the LSE for small α. However, for large α values, finding minimax D-optimal designs under the LSE Design 1 is faster than with either method using the GLSE initial weight vector.

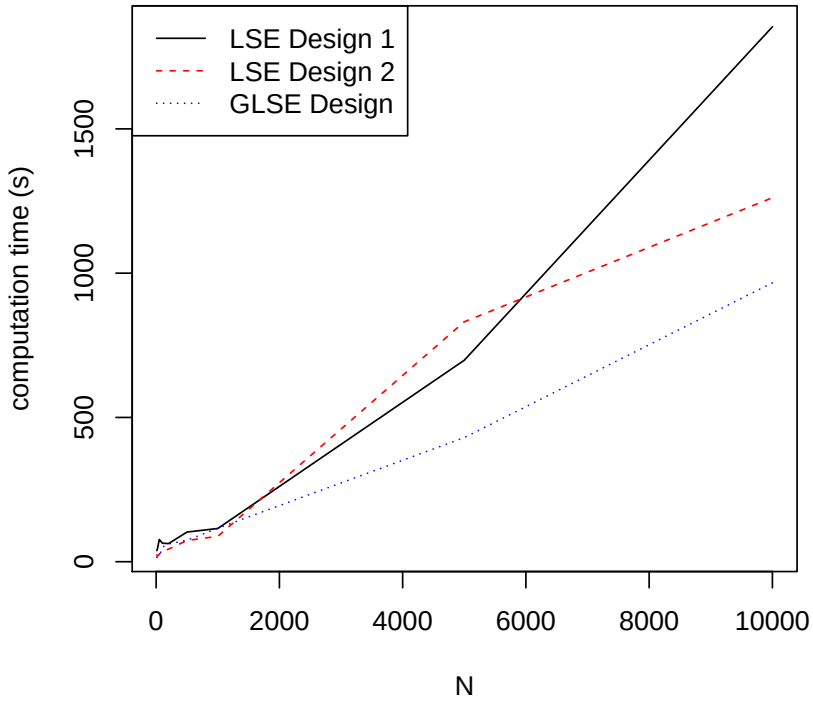

Figure 4.1: A plot of computation time versus the size of design space for $\alpha = 0.5$ in Example 1. The minimax D-optimal designs under the LSE are computed using two different initial weight vectors, and the computation times are indicated by LSE Design 1 and LSE Design 2, respectively.

By Theorem 4 minimax D-optimal designs under the GLSE and the LSE are symmetric

for this model, and the numerical results are consistent with Theorem 4. For $N = 5001$, the minimax D-optimal designs under the GLSE and the LSE are computed for various α values, and they all have four symmetric support points with equal weights (0.25). Two boundary points, -5 and 5, are always among the four support points, and the other two support points change as α changes. Figure 4.2 shows a plot of the two positive support points in the D-optimal designs under the GLSE versus α. A similar plot is obtained for the LSE. When $\alpha = 0$, it shows the support points of the D-optimal design under the GLSE, which are indicated by the two triangles in Figure 4.2. As α increases, the support points converge to those of the D-optimal design under the LSE with constant error variance, and they are indicated by the two diamonds in Figure 4.2. When α is small, the information about the error variance is very accurate and the GLSE is recommended. On the other hand, when α is large, the error variance is almost unknown and the LSE is recommended.

From Figure 4.2 the minimax D-optimal design clearly does not stay at either of the two initial weight vectors we have suggested thus far, but changes with α. This suggests an improvement to the algorithm may be made by using an initial weight vector which incorporates α into its generation. An intuitive way to include α is to use the weight vector found by minimizing

$$-\log\left[\det\left(\sum_{i=1}^{N}[w_i/(g(\mathbf{u}_i)+\alpha)]\mathbf{A}_i\right)\right],$$

which is found by adding α to the variance of a standard GLSE minimization problem.

In Figure 4.3 we see a comparison of the GLSE and the LSE from their original starting weights to adjusted starting weights found by minimizing the above problem. It is clear from the graph that the adjusted start GLSE is significantly faster for all interior points, which is all times alpha has a nonzero definite value. We note different effects for the adjusted start LSE which is only a minor improvement over the default LSE, and not competitive with the adjusted start GLSE. The graph shows the LSE having better performance than

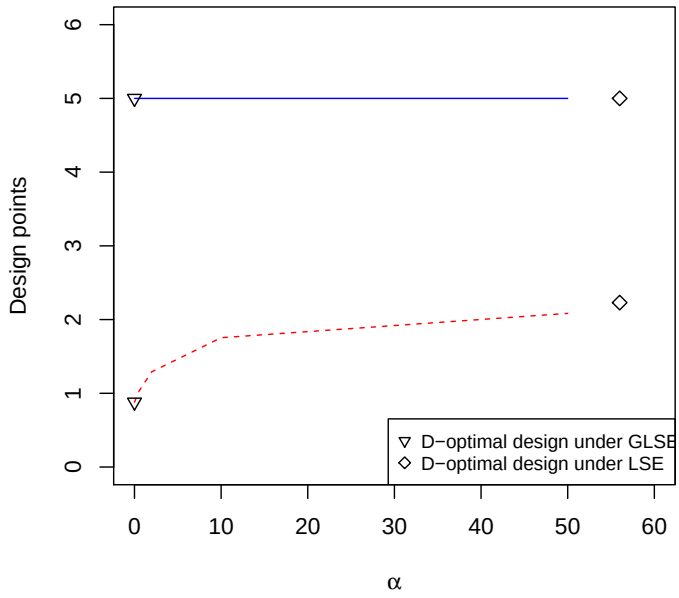

Figure 4.2: A plot of two positive support points in the minimax D-optimal designs under the GLSE versus α, as indicated by the blue and red lines. The two positive support points of the D-optimal designs under the LSE and GLSE are also plotted.

the GLSE at high alpha values, and this occurs on the graph between indicator 0.45 and 0.5, which is between $\alpha = 21.27$ and $\alpha = 26$. Additionally, in 11 of the 19 interior points, the adjusted start GLSE took two iterations to complete, which is the minimum possible to exit the algorithm, so the starting point of the adjusted GLSE may be the minimax D-optimal design in some cases.

We have used $\eta = 10^{-4}$ in the stopping criterion of Algorithm 1. Using the necessary conditions in (3.3) and (3.4), we can verify the numerical results for minimax D-optimal designs. Figure 4.4 gives representative plots of functions $d_1(u)$ and $d_2(u)$ for $u \in [-5, 5]$, and it is clear that $d_1(u) \leq \eta_0$ and $d_2(u) \leq \eta_0$ for a small positive η_0. Thus, the necessary conditions in (3.3) and (3.4) are satisfied.

Alternatively, the necessary condition in (3.3) or (3.4) with a small positive η_0, say

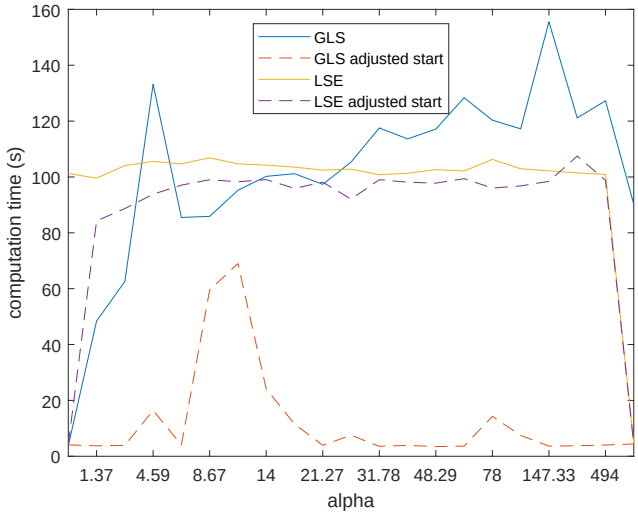

Figure 4.3: A plot of time vs α. Left and right endpoints are $\alpha = 0$ and the homogenous $\mathbf{V} = 1$ respectively. GLSE and LSE use their typical starting weights, while adjusted start indicates using the GLSE weights found by adding α to the variance.

$\eta_0 = 10^{-5}$, can be used as a stopping criterion in Algorithm 1. Similar minimax D-optimal designs are obtained using this stopping criterion. □

Example 2. Consider a regression model for a mixture experiment with four design variables (mixture proportions), and the model is given by

$$y = \theta_1 x_1 + \theta_2 x_2 + \theta_3 (x_1 x_2)^{0.5} + \theta_4 x_3 + \theta_5 x_4 + \epsilon, \quad \text{with} \quad \lambda(\mathbf{x}) = e^{-(x_1 + 2x_3 + x_4)},$$

where the design variables satisfy

$$\sum_{i=1}^{4} x_i = 1, \quad x_i \geq 0, \text{ for } i = 1, \ldots 4. \tag{4.1}$$

This model is studied in Yan et al. (2017), where D-optimal designs are constructed assuming that the error variance function $\lambda(\mathbf{x})$ is exactly correct. We compute minimax D-optimal designs with $g(\mathbf{x}) = e^{-(x_1 + 2x_3 + x_4)}$ and make comparisons.

We first discretize the design space and then compute minimax D-optimal designs. Since

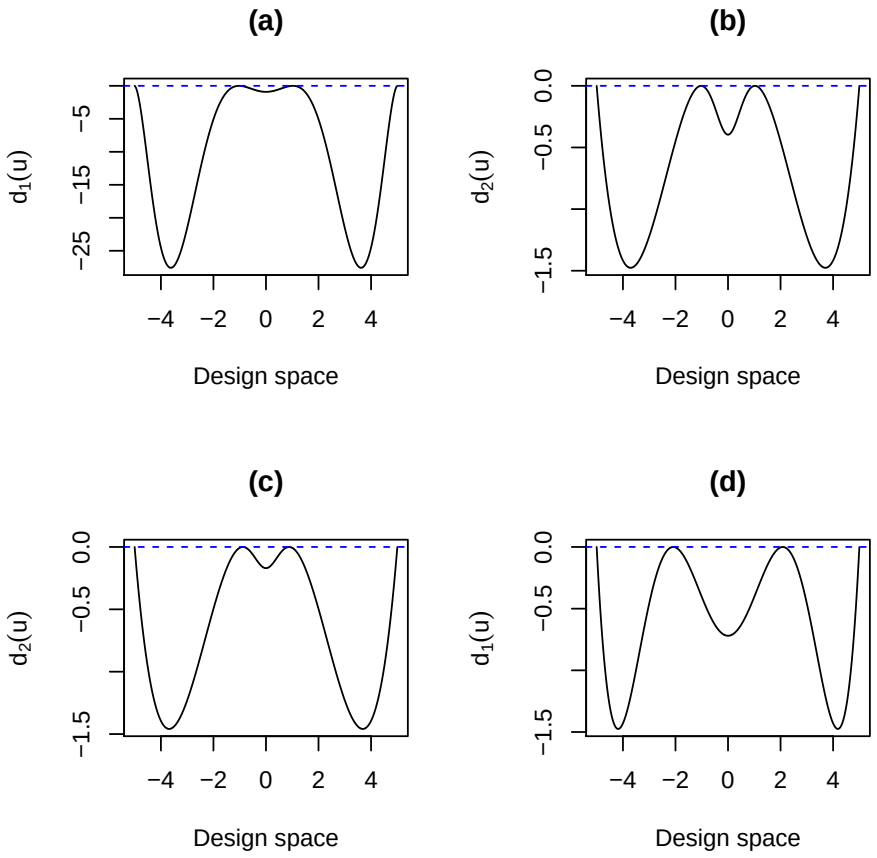

Figure 4.4: Plots of functions $d_1(u)$ and $d_2(u)$ for the minimax D-optimal designs for four cases: (a) LSE and $\alpha = 0.5$, (b) GLSE and $\alpha = 0.5$, (c) GLSE and $\alpha = 0.0$, (d) LSE and $\alpha = 50$.

the design space specified by (4.1) is not a cube, we take the following steps to get a discrete design space S_N.

(i) Let N_1 be an integer. For each variable $x_i \in [0,1]$, $i = 1,\ldots,4$, use N_1 equally spaced points to get $x_{i1},\ldots,x_{iN_1}$, i.e., $x_{ij} = (j-1)/(N_1-1),\ j = 1,\ldots,N_1$.

(ii) Define $S_N = \left\{(x_{1j_1}, x_{2j_2}, x_{3j_3}, x_{4j_4})^\top :\ \sum_{i=1}^4 x_{ij_i} = 1,\ j_1, j_2, j_3, j_4 \in \{1,2,\ldots,N_1\}\right\}$.

Several values of N_1 are used to obtain different sizes of S_N, and minimax D-optimal designs are computed for each S_N. The code used to find minimax D-optimal designs between differing examples will tend to need adjustments in the search space and information matrix creation to work. Here, the code from Example 1 becomes:

```
%Set size per variable and number of variables
n = 11; m = 4;
%get search space: assumed bounds 0 and 1 with summation in design to 1
u = desSpace_Mixture(n,m); %See Appendix for details
N = size(u,1);
%describe size of information matrix, alpha and stopping criteria
q = 5; alpha = 5;
deltaw = 1e-4; delta_equiv = 1e-6; maxiter = 50;
%setup information matrices
infomat = zeros(q,q,N);
inforob = zeros(q,q,N);
for i=1:N
            V = exp(-(u(i,1)+2*u(i,3)+u(i,4)));
          F = [u(i,1) u(i,2) (u(i,1)*(u(i,2)))^(1/2) u(i,3) u(i,4)];
            infomat(:,:,i) = F'*F/V;
            inforob(:,:,i) = F'*F*((V+alpha)/V^2);
end
%initial and iterative weights, see appendix for details
w0 = initialw(infomat);
```

```
[iter,determ,equiv,time,wn] = RobDopt_Diagnostic(w0,infomat,inforob,
    maxiter,deltaw,delta_equiv);
%use a minimum weight (0.0009 here) to get design from weights
design = [u(wn(iter,:)>0.0009)' wn(iter,wn(iter,:)>0.0009)'];
```

The computational results show that the number of support points of minimax D-optimal designs is either 5 or 6, and all the minimax D-optimal designs include the following 4 points,

$$\mathbf{x}_1^* = (1,0,0,0)^\top,\ \mathbf{x}_2^* = (0,1,0,0)^\top,\ \mathbf{x}_3^* = (0,0,1,0)^\top,\ \mathbf{x}_4^* = (0,0,0,1)^\top,$$

and one or two additional points are in the following form and denoted as

$$\mathbf{x}_5^*[x_{15}] = (x_{15}, 1-x_{15}, 0, 0)^\top,\ \mathbf{x}_6^*[x_{16}] = (x_{16}, 1-x_{16}, 0, 0)^\top,$$

where x_{15} and x_{16} depend on N_1, α, and the design criterion. Table 4.1 gives representative results of minimax D-optimal designs for $N_1 = 21$ and 51, and various α values. There are four common support points $(\mathbf{x}_1^*, \mathbf{x}_2^*, \mathbf{x}_3^*, \mathbf{x}_4^*)$ for minimax D-optimal designs under the LSE and GLSE, while the other one or two support points can be different. In some cases the support points are the same for the D-optimal designs under the LSE and GLSE, but the weights are slightly different. For $N_1 = 21$, there are $N = 1584$ points in S_N, and it takes about 13 to 134 seconds to compute each design in Table 4.1. For $N_1 = 51$, there are $N = 22,099$ points in S_N, and it takes about 961 to 2730 seconds to compute each design. For some α values and $N_1 = 51$, it takes a long time for Algorithm 1 to converge and sometimes it fails to converge, in particular for the LSE, since CVX has issues with huge N and can be quite slow.

For $\alpha = 0$, the results are the D-optimal designs under the LSE and GLSE. Notice that the D-optimal design under the GLSE is obtained in Yan et al. (2017) using a very technical method, which is difficult to extend for finding D-optimal designs if we change

Table 4.1: Minimax D-optimal designs under the LSE and GLSE in Example 2 with $g(\mathbf{x}) = e^{-(x_1+2x_3+x_4)}$

Case	criterion	support points (and weights in parentheses)					
$N_1 = 21$	$N = 1584$						
$\alpha = 0.0$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.60] (.2000)	
	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.65] (.0424)	$\mathbf{x}_6^*$[.60] (.1578)
$\alpha = 0.3$	LSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.60] (.1471)	$\mathbf{x}_6^*$[.55] (.0531)
	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.60] (.1363)	$\mathbf{x}_6^*$[.55] (.0639)
$\alpha = 0.6$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.55] (.2000)	
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.55] (.2000)	
$\alpha = 5.0$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.55] (.0058)	$\mathbf{x}_6^*$[.50] (.1942)
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.50] (.2000)	
$N_1 = 51$	$N = 22,099$						
$\alpha = 0.0$	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.62] (.2001)	
$\alpha = 0.1$	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.1999)	$\mathbf{x}_5^*$[.60] (.2001)	
$\alpha = 0.6$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2001)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.56] (.2000)	
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.1999)	$\mathbf{x}_5^*$[.56] (.2001)	
$\alpha = 5.0$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.52] (.1675)	$\mathbf{x}_6^*$[.50] (.0325)
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.52] (.2000)	

the response function in the model or function $\lambda(\mathbf{x})$ slightly. However, Algorithm 1 can be applied to find D-optimal or minimax D-optimal designs easily for any model and any function $\lambda(\mathbf{x})$. For instance, Table 4.2 gives minimax D-optimal designs for a different $\lambda(\mathbf{x})$ with $g(\mathbf{x}) = e^{(x_1+2x_3+x_4)}$.

The D-optimal design under the LSE is new, not reported in Yan et al. (2017). The D-optimal design under the GLSE in Yan et al. (2017) has the following support points with equal weight 0.2000 at each support point:

$$\mathbf{x}_1^*,\ \mathbf{x}_2^*,\ \mathbf{x}_3^*,\ \mathbf{x}_4^*,\ \mathbf{x}_5^*[(\sqrt{5}-1)/2],$$

which are almost the same as the D-optimal design under the GLSE in Table 4.1 for $N_1 = 51$ and $\alpha = 0$. In practice it is probably reasonable to use discretized design points, since we may not be able to measure an input variable level as accurately as $(\sqrt{5}-1)/2$. □

Table 4.2: Minimax D-optimal designs under the LSE and GLSE in Example 2 with $g(\mathbf{x}) = e^{(x_1+2x_3+x_4)}$

Case	criterion	support points (and weights in parentheses)					
$N_1 = 21$	$N = 1584$						
$\alpha = 0.0$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.50] (.2000)	
	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.40] (.1578)	$\mathbf{x}_6^*$[.35] (.0424)
$\alpha = 0.6$	LSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.1999)	$\mathbf{x}_4^*$ (.1999)	$\mathbf{x}_5^*$[.40] (.2004)	
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.45] (.0020)	$\mathbf{x}_6^*$[.40] (.1980)
$\alpha = 1.0$	LSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.45] (.1133)	$\mathbf{x}_6^*$[.40] (.0869)
	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.45] (.1108)	$\mathbf{x}_6^*$[.40] (.0894)
$\alpha = 5.0$	LSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.50] (.0571)	$\mathbf{x}_6^*$[.45] (.1431)
	GLSE	$\mathbf{x}_1^*$ (.1999)	$\mathbf{x}_2^*$ (.1999)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.50] (.0291)	$\mathbf{x}_6^*$[.45] (.1711)
$N_1 = 51$	$N = 22,099$						
$\alpha = 0.0$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.50] (.2000)	
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.38] (.2000)	
$\alpha = 0.1$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.40] (.0437)	$\mathbf{x}_6^*$[.38] (.1563)
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.40] (.0756)	$\mathbf{x}_6^*$[.38] (.1244)
$\alpha = 0.6$	LSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.42] (.1975)	$\mathbf{x}_6^*$[.40] (.0025)
	GLSE	$\mathbf{x}_1^*$ (.2000)	$\mathbf{x}_2^*$ (.2000)	$\mathbf{x}_3^*$ (.2000)	$\mathbf{x}_4^*$ (.2000)	$\mathbf{x}_5^*$[.42] (.1683)	$\mathbf{x}_6^*$[.40] (.0317)

4.2 Nonlinear regression model

Example 3. Consider a segmented polynomial regression model,

$$y_i = \theta_0 + \theta_1 x_i + \theta_2 (x_i - \theta_4)_+ + \theta_3 (x_i - \theta_4)_+^2 + \epsilon_i, \quad a \le x_i \le b,$$

where function $(x_i - \theta_4)_+ = \max\{0, (x_i - \theta_4)\}$, and $V(\epsilon_i) = \sigma^2 \exp(x_i)$. This model is studied in Wu (2007) for finding D-optimal designs. We compute minimax D-optimal designs assuming that θ_4 is a regression parameter and the error variance may not be exactly as $\sigma^2 \exp(x_i)$. Since θ_4 is unknown, this is a nonlinear model and the minimax D-optimal design depends on θ_2, θ_3 and θ_4 from the discussion in Section 3.2. Let θ_2^*, θ_3^* and θ_4^* be the true value of θ_2, θ_3 and θ_4, respectively. In Algorithm 1, we use $g(x) = \exp(x)$ and N equally spaced discrete design points, $u_i = a + (b-a)(i-1)/(N-1)$, to form S_N. We can use the

following Matlab code for the minimax D-optimal designs under GLSE:

```
%beginning end and size of search space
a = 4.7; b = 6.3; N = 1001;
u = zeros(1,N);  %Points in the search space
for i = 1:N
    u(1,i) = a+(i-1)*(b-a)/(N-1);
end
%describe size of information matrix, alpha and stopping criteria
q = 5; alpha = 1;
deltaw = 1e-4; delta_equiv = 1e-6; maxiter = 50;
%Set priors for nonlinear:
theta3 = 0.3261;
theta4 = 0.0735;
theta5 = 5.1;
%setup information matrices
infomat = zeros(q,q,N);
inforob = zeros(q,q,N);
for i=1:N
            F =[1 u(i) 0 0 0];
   %If > is used, left limit is used
   %if >=, right limit is used.
    if(u(i)>=theta5)
            F(1,3)=u(i)-theta5;
            F(1,4)=(u(i)-theta5)^2;
            F(1,5)=(-theta3-2*theta4*(u(i)-theta5))/2;
    end
      V =  exp(u(i));
            infomat(:,:,i) = F'*F/V;
            inforob(:,:,i) = F'*F*((V+alpha)/V^2);
end
%obtain initial and iterative weights, see code in appendix for details
```

Table 4.3: Minimax D-optimal designs under the GLSE with $N = 5001$ in Example 3: Case (i) $a = 4.7,\ b = 6.3,\ \theta_4^* = 5.1$, Case (ii) $a = 4.7,\ b = 6.3,\ \theta_4^* = 5.5$, Case (iii) $a = 3.9,\ b = 7.1,\ \theta_4^* = 5.1$.

Case (i)	Support points with weights in parentheses
$\alpha = 0$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6120** (.14) **5.6123** (.06) 6.3000 (0.20)
$\alpha = 5$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6136** (.20) 6.3000 (.20)
$\alpha = 10$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6152** (.20) 6.3000 (.20)
$\alpha = 50$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6251** (.20) 6.3000 (.20)
$\alpha = 100$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6347** (.20) 6.3000 (.20)
$\alpha = 200$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6478** (.20) 6.3000 (.20)
$\alpha = 500$	4.7000 (.20) 5.1000 (.20) 5.1003 (.20) **5.6674** (.20) 6.3000 (.20)
Case (ii)	
$\alpha = 0$	4.7000 (.20) 5.5000 (.20) 5.5003 (.20) **5.8606** (.20) 6.3000 (.20)
$\alpha = 50$	4.7000 (.20) 5.5000 (.20) 5.5003 (.20) **5.8654** (.20) 6.3000 (.20)
$\alpha = 100$	4.7000 (.20) 5.5000 (.20) 5.5003 (.20) **5.8693** (.20) 6.3000 (.20)
Case (iii)	
$\alpha = 0$	3.9000 (.20) 5.1000 (.20) 5.1006 (.20) **5.8642** (.19) **5.8648** (.01) 7.1000 (.20)
$\alpha = 10$	3.9000 (.20) 5.1000 (.20) 5.1006 (.20) **5.8699** (.13) **5.8706** (.07) 7.1000 (.20)
$\alpha = 50$	3.9000 (.20) 5.1000 (.20) 5.1006 (.20) **5.8904** (.20) 7.1000 (.20)
$\alpha = 100$	3.9000 (.20) 5.1000 (.20) 5.1006 (.20) **5.9109** (.20) 7.1000 (.20)

```
w0=initialw(infomat);
[iter,determ,equiv,time,wn] = RobDopt_Diagnostic(w0,infomat,inforob,
   maxiter,deltaw,delta_equiv);
%use a minimum weight (0.0009 here) to get design from weights
design = [u(wn(iter,:)>0.0009)' wn(iter,wn(iter,:)>0.0009)'];
```

Table 4.3 presents several minimax designs under the GLSE for various values of a, b and α, where $\theta_2^* = 0.3261, \theta_3^* = 0.0735$ and $\theta_4^* = 5.1$ are from Wu (2007, page 50). We also have results for $\theta_4^* = 5.5$ in the table to discuss the sensitivity of the minimax designs.

The results indicate that the minimax D-optimal designs depend on a, b and θ_4^* heavily, and are not very sensitive to α. For each case in Table 4.3, the support points of the minimax designs include the boundary points of the design space, a and b, two points around θ_4^*, and one or two points around the middle of interval (θ_4^*, b). As α changes, four of the support points do not change at all for each case and the four points are the boundary points and

the two points around θ_4^*. The support points that change as α changes are highlighted using boldface in Table 4.3, and it is clear that they only change slightly. Similar results are obtained for the minimax D-optimal designs under the LSE. Algorithm 1 is flexible and fast to compute minimax D-optimal designs for various values of parameters and design spaces, which is useful in doing sensitivity analysis for the designs and in implementing designs in practice. □

After computing many minimax D-optimal designs using Algorithm 1 for various models, we have additional comments about Algorithm 1 below.

Since we use CVX to solve convex optimization problems in Step (iii) of Algorithm 1 and CVX is very fast when $N \leq 10,000$, Algorithm 1 is also fast for $N \leq 10,000$. When $N > 10,000$, both CVX and Algorithm 1 can be very slow for complicated regression models.

Choosing a good initial vector $\mathbf{w}^{(0)}$ is important for $\mathbf{w}^{(j)}$ to converge quickly. In some cases, $\mathbf{w}^{(j)}$ may fail to converge when the initial vector $\mathbf{w}^{(0)}$ is not close to the solution of the optimization problem (2.8), since the objective function of the problem is not a convex function of $\mathbf{w}$. In Chapter 3, we mentioned that we could use the minimizer of $v_1(\mathbf{w})$ as $\mathbf{w}^{(0)}$, and it works well for small α. For large α, it may be better to use one of the minimizers of the following functions:

$$-\log\left(\det\left(\sum_{i=1}^{N}[w_i/(g(\mathbf{u}_i)+\alpha)]\mathbf{A}_i\right)\right), \quad -\log\left(\det\left(\sum_{i=1}^{N}w_i\mathbf{A}_i\right)\right).$$

Notice that the above functions are convex functions of $\mathbf{w}$, and the minimizers can be easily found using CVX. In Example 1 we explored the first of these minimizers and noted that it always significantly improved the speed of the algorithm for the GLSE, but for Example 2 the same minimizer was worse for small values of α. In Example 3, it was also consistently the best method until α was very large. In practice, we do not know if α is relatively small or large, or what method will be the best for a model, but we can use several initial vectors

for $\mathbf{w}^{(0)}$ to run Algorithm 1 and choose the best result as the minimax D-optimal design.

Chapter 5

Conclusion

We have studied minimax D-optimal designs for regression models with heteroscedastic errors. Several theoretical results for the minimax D-optimal designs are derived, and an effective algorithm is developed for computing those designs for any linear or nonlinear regression model. Since we never know the exact error variance in practice, our results are quite useful to construct minimax D-optimal designs and conduct sensitivity analysis for those designs.

We have found that for some models, alternate initial weight vectors may bring significant improvements to the speed of the algorithm. In some cases, alternate initial weight vectors may also be the solution to the minimax D-optimal design problem. If it could be shown that a specific minimizer will always or usually be the solution to a minimax D-optimal design problem for a given model, then it could provide a method to analyze problems with uncertain error variance without needing our algorithm.

From equation (2.2) we defined an α which was fixed across all variances in the sample space, but this restriction is not required for our algorithm. Throughout our theorems and proofs, we have only used α when it was associated with a specific variance such as $g(\mathbf{x}_i)$. We could instead use an α which varies across the sample space, finding our optimal designs given $\alpha(\mathbf{x}_i)$. While this is an interesting result, using $\alpha(\mathbf{x}_i)$ implies knowledge of how the

uncertainty of our variance estimate changes with the input $\mathbf{x}_i$, which may be difficult to justify in practical applications.

We have focused on the D-optimality criterion in this thesis, but our methodology can be applied to other optimality criteria, such as A-optimality and c-optimality. It would be interesting to investigate minimax A-optimal or minimax c-optimal designs. However, the objective functions of the related optimization problems are not convex functions and are not a difference of two convex functions. Thus, it may be challenging to work on minimax A-optimal or minimax c-optimal designs.

Appendix A

Matlab code

```
%Included Functions:
%the RobDopt_Diagnostic function which does the overall structure of the
    algorithm
%getDet which retrieves the determinant under the robust case
%getB2 which retrieves  tr(infomat^-1 * infomat.*w), required for the
    algorithm
%initialw for the nonrobust starting point, and getw for the adjustments
    on a prior weighting
%desSpace for a standard search space
%desSpace_Mixture for search space from 0 to 1 with sum of variables = 1

%Diagnostic version, simplified version only requires last iteration
    values to be returned
%Inputs: information matrix, information matrix including robust variance
    adjustment, number of loops permitted (safety), delta for weight
    stopping criterion or equivalence theorem.
%Outputs: iterations to complete, determinants of each iteration,
    equivalence theorem matrix for each iteration, time per iteration, and
    weights for each iteration.
function [iter,d2,equiv,time,wn] = RobDopt_Diagnostic(w0,infomat,inforob,
```

```
    Nloop,delta,delta_equiv)
%Default values
    if(Nloop<1)
        Nloop=50;
    end
    if(delta_equiv<=0)
        delta_equiv=1e-3
    end
    if(sum(w0)==0)
        w0=initialw(infomat);
    end
    tic;

    %retrieve basic info from information matrix
    N=size(infomat,3);
    q=size(infomat,2);

    %setup determinant and time matrices
    d2=zeros(Nloop,1);
    time = zeros(Nloop,1);
    %setup weights and equivalence results
    wn = zeros(Nloop,N);
    equiv= zeros(Nloop,N);

    %initialize base case of weights
    wn(1,:)=w0;
    d2(1) = getDet(wn(1,:),inforob,infomat);

    %get B2 retrieves trace(info^(-1) (info.*w)) which is required for
    both the computation loop, and the equivalence thm result
    A2 = getB2(wn(1,:),infomat);
```

```
42      B2 = getB2(wn(1,:),inforob);
43
44      %equivalence result:
45      equiv(1,:)=2*A2-B2-q;
46      %iter is marker to know how many cycles were done
47      iter=0;
48
49      %working for loop for the algorithm:
50      for i = 2:Nloop
51          time(i-1)=toc;
52          tic;
53          iter=i;
54
55          wn(i,:)=getw(wn(i-1,:),B2,infomat);
56
57          A2=getB2(wn(i,:),infomat);
58          B2=getB2(wn(i,:),inforob);
59          equiv(i,:) = 2*A2-B2-q;
60
61          d2(i)=getDet(wn(i,:),inforob,infomat);
62
63          %test by max difference in weights
64          maxwdiff= max(wn(i,:)-wn(i-1,:))
65          if(maxwdiff<delta)
66              break
67          end
68
69          %check equivalence theorem result
70          if(max(equiv(i,:))<delta_equiv)
71              break
72          end
```

```
73      end
74      time(iter)=toc;
75  end
76
77  %getDet retrieves the determinant of w given true case is robust (Balpha
        information matrix), (B is regular information matrix)
78  function d = getDet(w,Balpha,B)
79      q=size(Balpha,2);
80      N=size(Balpha,3);
81      A=zeros(q,q);
82      A2=zeros(q,q);
83
84     for i=1:N
85      A=A+B(:,:,i)*w(i);
86      A2=A2+Balpha(:,:,i)*w(i);
87     end
88     Ainv=inv(A)^2;
89      d = det(Ainv)*det(A2);
90  end
91
92  %getB2 gets trace(Ainv*information matrix with alpha), used in Difference
        of convex functions equation
93  %also used in equiv thm
94  function B2 = getB2(w,Balpha)
95      q=size(Balpha,2);
96      N=size(Balpha,3);
97      Aalpha=zeros(q,q);
98
99      for i=1:N
100         Aalpha =Aalpha+Balpha(:,:,i)*w(i);
101     end
```

```
102
103     Ainv=inv(Aalpha);
104     B2=1:N;
105
106     for i=1:N
107         B2(i)=trace(Ainv*Balpha(:,:,i));
108     end
109 end
110
111 %inputs information matrix B, previous weight set w
112 %derivative vector B2 (from the Difference of Convex functions equation)
113 %outputs weights from an iteration on the DC equation
114 function w2 = getw(w,B2,B)
115     %using B2 to find N,q sizes
116     N=size(B,3);
117     q=size(B,2);
118
119     cvx_begin quiet
120        cvx_precision default
121        variable w2(N);      %design weights
122        expression A(q,q);
123        expression A2;  %information matrix - target function
124        for i=1:N
125            A=A+B(:,:,i)*w2(i);
126            A2=A2+B2(i)*w2(i);  %D-optimal
127        end
128       minimize (-2*log_det(A)+A2);  %D-optimality
129        subject to
130                sum(w2)==1;
131                w2>=0;
132     cvx_end
```

```
133 end
134
135 %initial set of weights from information matrix B
136 function w = initialw(B)
137     N=size(B,3);
138     q=size(B,2);
139     %initialize w
140     cvx_begin quiet
141        cvx_precision high
142        variable w(N);      %design weights
143        expression A(q,q);  %information matrix - target function
144        for i=1:N
145         A=A+B(:,:,i)*w(i);  %D-optimal        %
146        end
147       minimize (-2*log_det(A))  %D-optimality
148        subject to
149                sum(w)==1;
150                w>=0;
151     cvx_end
152 end
153
154 %Design space construction: (n evenly spaced points from a to b)
155 %inputs may be vectors, included due to mixture models always needing the
    support. See example 1 for simplified form.
156 function design_space = desSpace(a,b,n)
157     m=length(n);
158     N=1;
159     d=1:m;
160     k=(1:m)*0;
161
162 %spacing between points for a given variable:
```

```
163     for j = 1:m
164        N=N*n(j);
165        d(j) = (b(j)-a(j))/(n(j)-1);
166     end
167
168     design_space = zeros(N,m);
169
170 %this will give every combination of inputs of all variables (scalar input
      silmplifies to n evenly spaced points between a and b)
171     for i = 1:N
172         design_space(i,1) = a(1) + k(1)*d(1);
173         k(1) = k(1)+1;
174         for j = 2:m
175             design_space(i,j) = a(j) + k(j)*d(j);
176            if(k(j-1)>=n(j-1))
177                k(j-1)=0;
178                k(j)= k(j)+1;
179            end
180         end
181     end
182     design_space = design_space(:,1:m);
183 end
184
185 %Design space for mixture models:
186 %Premise: get full design space for all combinations of inputs, restrict
      to the sum being 1.
187 %This is a general version, which supports vector inputs of length k,
      allowing for k different sizes n(k) for m(k) variables. For example 2
      inputs are scalar, which simplifies this function
188 function design_space = desSpace_Mixture(n,m)
189     k=length(m);
```

```
190     N=1;
191     n2=n;
192 % k=1 for example 2:
193     for i = 1:k
194         a = (1:m(k))*0;
195         b = (1:m(k))*0 +1;
196         n2 = (1:m(k))*0 +n(k);
197 
198 %This gets all combinations of inputs via desSpace with vector input
199         temp = desSpace(a,b,n2);
200 
201 % This can have rounding errors in the sum, so using -1<1e-15 to detect "
    equality"
202         if(i==1)
203             temp2 ={temp(abs(sum(temp.')-1)<1e-15,:)};
204         end
205         temp2{k} =temp(abs(sum(temp.')-1)<1e-15,:);
206         n(i) = size(temp2{i},1);
207         N=N*n(i);
208     end
209 
210     q=(1:k)*0+1;
211     design_space = zeros(N,sum(m));
212 
213     if(k==1)
214        design_space=temp2{1};
215        return
216     end
217 %Example 2 ends at the return above, below is for vector inputs
218     for i = 1:N
219         design_space(i,1:m(1)) = temp2{1}(q(1),:);
```

```
220         q(1) = q(1)+1;
221         for j = 2:k
222             design_space(i,(j+1):(j+m(j))) = temp2{j}(q(j),:);
223             if(q(j-1)>n(j-1))
224                 q(j-1)=1;
225                 q(j)= q(j)+1;
226             end
227         end
228     end
229 end
```

References

Berger, M.P.F. and Wong, W.K. (2009). *An Introduction to Optimal Designs for Social and Biomedical Research*. Wiley, New York.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York.

Dean, A., Morris, M., Stufken, J. and Bingham, D. (2015). *Handbook of Design and Analysis of Experiments*. CRC Press, Boca Raton.

Dette, H., Haines, L.M. and Imhof, L. (1999). Optimal desgins for rational models and weighted polynomial regression. *The Annals of Statistics*, 27, 1272-1293.

Dette, H. and Wong, W.K. (1999). Optimal designs when the variance is a function of the mean. *Biometrics*, 55, 925-929.

Gao, L.L. and J. Zhou (2020). Minimax D-optimal designs for multivariate regression models with multi-factors. *Journal of Statistical Planning and Inference*, 209, 160-173.

Grant, M.C. and Boyd, S.P. (2013). The CVX Users Guide, Release 2.0 (beta). CVX Research, Inc., Stanford University, Stanford.

Kitsos, C.P., Lopez-Fidalgo, J. and Trandafir, P.C. (2006). Optimal designs for Michaelis-Menten model. Greek Statistical Institute annual meeting, January 2006, 467-475.

Lipp, T. and Boyd, S. (2016). Variations and extensions of the convex-concave procedure. *Optimization and Engineering*, 17, 263-287.

Martínez-López, I., Ortiz-Rodríguez, I.M. and Rodríguez-Torreblanca, C. (2009). Optimal designs for weighted rational models. *Applied Mathematics Letters*, 22, 1892-1895.

Papp, D. (2012). Optimal designs for rational function regression. *Journal of the American Statistical Association*, 107, 400-411.

Pukelsheim, F. (2006). *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics.

Wiens, D.P. (2015). Robustness of Design, in *Handbook of Design and Analysis of Experiments*, edited by Dean, A., Morris, M., Stufken, J. and Bingham, D., CRC Press, Boca Raton.

Wiens, D.P. and Li, P. (2014). V-optimal designs for heteroscedastic regression. *Journal of Statistical Planning and Inference*, 145, 125–138.

Wong, W.K., Yin, Y. and Zhou, J. (2019). Using SeDuMi to find various optimal designs for regression models, *Statistical Papers*, 60, 1583–1603.

Wu, X.F. (2007). *Optimal Designs for Segmented Polynomial Regression Models And Web-based Implementation of Optimal Design Software*. PhD Thesis, Stony Brook University.

Yan, F., Zhang, C. and Peng, H. (2017). Optimal designs for additive mixture model with heteroscedastic errors. *Communications in Statistics - Theory and Methods*, 46, 6401-6411.

Ye, J.J., Zhou, J. and Zhou, W. (2017). Computing A-optimal and E-optimal designs for regression models via semidefinite programming. *Communications in Statistics - Simulation and Computation*, 46, 2011-2024.

Yzenbrandt, K. and Zhou, J. (2020). Minimax robust designs for regression models with heteroscedastic errors. Submitted.

Zhai, Y. and Fang, Z. (2018). Locally optimal designs for some dose–response models with continuous endpoints. *Communications in Statistics - Theory and Methods*, 47, 3803–3819.