University of Alberta

COMPARISON OF THREE METHODS OF GENERATING ROBUST STATISTICAL DESIGNS

by

Dengdeng Yu

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Statistics

Department of Mathematical and Statistical Sciences

Fall 2013

Edmonton, Alberta

Abstract

This thesis deals with the problem proposed by Ye and Zhou (2007): Is a *Q-optimal minimax design* still symmetric if the requirement of $\int_{\mathcal{X}} xm(x)dx = 0$ is removed? We have shown that for the simple linear regression, considering only the variance, a Q-optimal minimax design is necessarily symmetric; we have also made an attempt of addressing the symmetry problem considering only the bias which is much more difficult to achieve. However, the numerical results using three different algorithms, *Genetic Algorithm (GA)*, *Particle Swarm Optimization (PSO)* and *Expected Improvement Algorithm (EIA)*, indicate that the claim is true. We have also applied the three algorithm on a non-linear cases correspondingly and make the comparison.

Acknowledgements

First of all, I would like to express my sincere gratitude to my master's supervisor Professor Douglas P. Wiens, who patiently provides me with his unreserved help and guidance and leads me finish my thesis step by step. He always inspires me to think in a rigorous and creative way, which is extremely helpful for my thesis. I am not good at making an efficient time arrangement and sometimes stuck on a single point for several weeks without any progress. His encouragement and advice always ease my drought of mind and confidence like a timely rain. Being a strong and supportive supervisor during my master study, he also gives me great freedom to pursue independent study and research. That is one of the most important factors help me make my determination of pursuing the Ph.D. in statistics here in University of Alberta.

Moreover, I want to give gratitude to the rest of my committee members, Professor Linglong Kong, Professor Keumhee Carriere Chough and Professor Christoph Frei for their support, guidance and valuable suggestions.

Furthermore, I also would like to thank all of my friends and colleagues who give me continuous support and encouragement during my study.

Last but not the least, special thanks go to my parents. They provide me with a carefree environment to grow up and live my own life. Their encouragement, support and love are always something I can have to fall back on.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Robust Design

Statistical inferences are based only partly upon the observations. Even under the simplest situations, there are some explicit or implicit assumptions being made: randomness and independence, distributional models, sometimes a certain prior distribution for the unknown parameters and so on.

As in other branches of statistics, these assumptions are usually introduced to serve the purpose of rationalization and simplification. They are fundamentally important. However, they are not necessarily true. If some assumptions were not exactly true or just slightly violated, do those minor deviations from the assumptions cause merely a small change of the inference?

In general the answer is no. As mentioned in Huber and Rochetti (2009), since the mid 20th century, statisticians have been increasingly aware of the excessive sensitivity of many statistical procedures against seemingly minor deviations from the

assumptions. To address such a problem, the concept of "robustness" is introduced. And a definition is proposed by Huber and Rochetti (2009) of such a new statistical terminology: *robustness signifies insensitivity to small deviations from the assumptions.*

Before talking about the robustness of design, we firstly clarify the concept of design. As explained in Wu and Hamada (2009), (an) *experimental design is a body of knowledge and techniques that enables an investigator to conduct better experiments, analyze data efficiently, and make the connections between the conclusions from the analysis and the original objectives of the investigation.* Design is the guidebook of data collecting and analyzing.

A robust design is required not only to *make the connections between the conclusions from the analysis and the original objectives* but also to make such connections *robust*, or insensitive, against certain assumption contaminations. Take simple linear regression as an example. Though assuming the linear fitted model to be approximately true, we still expect to see that a corresponding design can tolerate some deviations from that model of the actual observations.

There are many ways to generate a robust design. In this thesis, three methods are applied and compared.

The first one is *Genetic Algorithm (GA)* (for an application, see Karami and Wiens (2012)), which is a search heuristic that mimics the process of natural evolution. After introducing some genetic sharing and mutation rules, we expect to see an optimal design winning out after several generations of propagation.

The second method is *Particle Swarm Optimization (PSO)* (for an application, see Chen, Chang, Wang and Wong (2011)), which is inspired by social behaviour of bird flocking or fish schooling. It sorts out an optimal design by mimicking during searching for food the birds' social activities such as information sharing and flocking. *PSO* generates a globally best design by collection information out of locally best designs.

The last one is *Expected Improvement Algorithm (EIA)* (originally from the paper of Jones, Schonlau and Welch (1998)) which is a sequential optimization method to obtain a best design using only a few evaluations of optimized function. It is extremely useful when the optimized function evaluation is extremely expensive.

I am going to illustrate these three methods by applying them to two different problems. One is for linear regression and the other one is for non-linear regression. After that I will compare the corresponding results among all three methods.

In this thesis I also include a symmetry problem which was partly solved by Ye and Zhou (2007). I will investigate the question of whether robust designs for straight line regression are necessarily symmetric by proposing the analytic attempts which have been made so far. Using the three methods mentioned above, we can give numerical evidence for that problem.

1.2 Chapter Structure

The thesis is organized as follows. In Chapter 2, I will describe the ideas of the three numerical algorithms, i.e. *GA*, *PSO* and *EIA*. I will also make a brief review of the pertinent literature for these three methods. In Chapter 3, I will describe the symmetry problem, and the analytic attempts which have been made. For Chapter 4, the symmetry problem of a simple linear case is numerically investigated by using all three of these algorithms. For Chapter 5, a numerical solution of a non-linear problem is given by using all three algorithms. And Chapter 6 will give the summary and conclusions, and make an outlook.

Chapter 2

Optimization Strategies

In this chapter, firstly I will give a brief introduction of problems related to robustness of design. Afterwards I will describe the ideas of the three numerical algorithms, i.e. *Genetic Algorithm (GA)*, *Particle Swarm Optimization (PSO)* and *Expected Improvement Algorithm (EIA)*. I will also make a brief review of the pertinent literature for these three methods.

2.1 Robustness of Design

Generally speaking, a *design* is specified by a probability mass function on the design space. More specifically, a *design* is a specification of $(z^{(1)}, \cdots, z^{(N)}) := (\frac{n_1}{n}, \cdots, \frac{n_N}{n})$, where $n_i \in \mathbb{N}$ for $i = 1, \cdots, N$ and $\sum_{i=1}^{N} n_i = n$. Implementing the design, the experimenter makes n_i observations $\{Y_{ij}\}_{j=1}^{n_i}$ at *location* $\boldsymbol{x}_i$ - a q-dimensional vector of covariates chosen from a design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^{N}$. We note that N, though finite, can be arbitrarily large, approximating a continuous design space, while $n_i \geq 0$ indicates no requirement that observations should be made at every *location*, or design point.

After sampling, a known regression response function $f(\boldsymbol{x}|\cdot)$ will be fitted, i.e. $\hat{Y}(\boldsymbol{x}) = f(\boldsymbol{x}|\hat{\boldsymbol{\theta}})$. The parameter $\boldsymbol{\theta}$ is chosen to minimize an L_2-distance between the true regression response $E[\boldsymbol{Y}(\boldsymbol{x})]$ and the fitted response $f(\boldsymbol{x}|\boldsymbol{\eta})$, i.e.

$$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\eta}} \int_{\mathcal{X}} (E[Y(\boldsymbol{x})] - f(\boldsymbol{x}|\boldsymbol{\eta}))^2 d\boldsymbol{x},$$

or

$$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\eta}} \sum_{i=1}^{N} (E[Y(\boldsymbol{x}_i)] - f(\boldsymbol{x}_i|\boldsymbol{\eta}))^2.$$

While the fitted response $f(\boldsymbol{x}|\boldsymbol{\theta})$ might be inadequate, the difference $\psi(\boldsymbol{x})$ is evaluated, or defined as

$$\psi(\boldsymbol{x}) = E[Y(\boldsymbol{x})] - f(\boldsymbol{x}|\boldsymbol{\theta}).$$

Then an experimenter could take observations on a random variable $\boldsymbol{Y}$ obeying the model as

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}|\boldsymbol{\theta}) + \psi(\boldsymbol{x}) + \epsilon,$$

for some p-dimensional parameter vector $\boldsymbol{\theta}$, errors ϵ and unknown contaminations, or misspecifications, $\psi(\boldsymbol{x})$.

In this thesis, we are looking for designs that are robust against model misspecifications, $\psi(\boldsymbol{x})$, that to each design is assigned a loss value $\mathbb{L}$ (detailed deduction can be found in Chapter 3) reflecting the mean squared error of the estimates or fitted values. In other words, the robustness of a design is measured by a loss value

$\mathbb{L}$ and an optimal design is desired to minimize $\mathbb{L}$. Now we can explain the three algorithms in terms of searching for optimal densities, or designs.

2.2 Genetic Algorithm

Inspired by the ideas of "evolutionary systems" which had been studied by computer scientists in the 1950s and the 1960s, *Genetic Algorithms (GAs)* were invented by John Holland in the 1960s and were developed by Holland and his students and colleagues at the University of Michigan in the 1960s and the 1970s (for complete history of evolutionary computation, see Mitchell (1996)).

The idea of solving the optimization problem in "evolutionary systems" *was to evolve a population of candidate solutions....., using operators inspired by natural genetic variation and natural selection* (Mitchell (1996)).

As a search heuristic that mimics the process of natural evolution, the genetic algorithm used in this thesis is as in Welsh and Wiens (2013) (for complete background material see Mandal, Johnson, Wu and Bornemeier (2007)). It was also applied for a nonlinear regression by Karami (2011) and Karami and Wiens (2012). The basic idea is as explained in section 3 of Karami and Wiens (2012):

> *...Such algorithms have been developed by using notions of evolutionary theory: we generate 'populations' of designs that evolve over 'generations'. In the evolutionary processes 'fit parents' are combined to produce 'children' via stochastic process of 'crossover' and 'mutation'.*

In *GAs*, there are two important parameters - *crossover* and *mutation*. Each of them uses a probability to control the homogeneity and heterogeneity between *parents*

and *children*. Crossover is made in hopes that the children should inherit the best parts from the parents in order to get closer to the optimized solution to the problem. On the other hand, mutation changes some characteristics inherited from parents to prevent all the children from becoming the same and falling into a locally optimized solution.

As the common sense of evolutionary theory, the homogeneity should dominate over heterogeneity. As suggested by Coley (1999), a typical choice of the probability of *crossover* ranges from 0.4 to 0.9, while the probability of *mutation* is of the order of 0.001. However, in this thesis, I am going to use a much larger probability of *mutation*, suggested by Karami and Wiens (2012), which varies linearly from 0 to 0.5 by generations since the last occurrence of an improved design.

After introducing the genetic sharing and mutation rules by using the parameters of *crossover* and *mutation*, we expect to see an optimal *crossover* solution winning out after several generations of propagation and robust against *mutation*.

Get back to our design problem. According to Welsh and Wiens (2013) and the explanation in Section 2.1, we assume a loss function $\mathbb{L}$ to be minimized, the corresponding *GA* can be implemented as follows: (In this algorithm, there are some tuning constants, which could be chosen arbitrarily, but they turned out not to affect much the algorithm performance.)

1. Start by randomly generating the first 'generation' of fixed K (e.g. 40) designs, $\boldsymbol{\xi}_1, \ldots \boldsymbol{\xi}_K$. For our design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^N$, each design (density) can

be identified with a multinomial vector of length N, with sum n.

2. For the current generation of designs, evaluate the loss R_k for each design $\boldsymbol{\xi}_k$, viz. $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, $k = 1, 2, \ldots, K$, and the corresponding 'fitness levels'

$$\text{fitness}_k = \frac{1}{(R_k - 0.99R_{\min})^2}, k = 1, 2, \ldots, K.$$

where $R_{\min}$ is the minimum value of the loss in the current generation. Then we scale the fitness levels to form a probability distribution ϕ_k,

$$\phi_k = \frac{\text{fitness}_k}{\sum_{j=1}^{K} \text{fitness}_j}, k = 1, 2, \ldots, K.$$

3. Form a new generation (children) of K designs to replace the current generation (parent).

   (a) The best (fittest) $N_{elite} = KP_{elite}$ in the current generation always survive to the next generation. The remaining $K - N_{elite}$ members are formed by crossover and mutation. (e.g. If we use $K = 40$ and $P_{elite} = .05$, then $N_{elite} = 2$. In other words, we are going to generate $K - N_{elite} = 38$ new designs as the next generation.)

   (b) Crossover proceeds as follows:

   - Choose two members of the current generation to be parents with probability proportional to their fitness level: If ζ_1,$\zeta_2 \sim$ independent Uniform $(0, 1)$, then choose to be parents the current generation

members $i_1^\star$ and $i_2^\star$, where

$$i_1^\star = \min\{i : \sum_{j=1}^{i} \phi_j \geq \zeta_1\};$$
$$i_2^\star = \min\{i : \sum_{j=1}^{i} \phi_j \geq \zeta_2\}.$$

This is the so-called *roulette-wheel* selection method. It ensures that the most fit members of the current population are the most likely to be chosen as parents. The same parent can be chosen twice without posing difficulties for the algorithm.

- With probability $1 - P_{crossover}$, the child is identical to the fittest parent.
- With probability $P_{crossover}$, the parents both contribute towards the child, in the following manner. Each member of the current generation can be represented by its vector $n\boldsymbol{\xi}$ of design point allocations. The two vectors of allocations arising from the parents are averaged, and then any fractional allocations are rounded down. This results in a vector S with integer elements, with sum s possibly less than n. If $s < n$ then $n - s$ design points are randomly chosen from S (with replacement) and added to the design. The child formed in this way is added to the new generation.

(c) Mutation is applied independently to each child. With probability $P_{mutation}$, κ elements of child design are randomly chosen, and replaced by a multinomial vector of length κ with the same sum as the elements being replaced. The value of κ is chosen by the user; we typically use $2 \leq \kappa \leq 6$. With probability $1 - P_{mutation}$ we do nothing.

4. Step 3 is carried out repeatedly until the next generation has been formed. The loss is guaranteed to decrease, although weakly, in each generation, because of the inclusion of the best two parents. We run the algorithm until the best design does not change for G generations.

Empirically, without introducing any mutation, the final generation of children achieved by crossover are always identical with each other which was an unseemly feature we expect. An optimal design, as a strong species in biological terminology, should win out due to its own genetic advantage. And this advantage should be held robustly against most genetic mutation. Hence mutation should be introduced.

2.3 Particle Swarm Optimization

The *Particle Swarm Optimization (PSO)* is inspired by social behaviour of bird flocking or fish schooling. It was firstly proposed by Eberhart and Kennedy (1995) to deal with the optimization of nonlinear functions.

Since the method was generated, it has gained considerable popularity by *its numerous applications to many disciplines* (see details in Chen, Chang, Wang and Wong (2011)). According to Chen, Chang, Wang and Wong (2011), there are already books, such as Clerc (2006), devoted entirely to *PSO*, though there were already some earlier books talking about the idea of swarm intelligence in general such as Eberhart, Shi and Kennedy (2001).

As a stochastically iterative procedure to function optimization, the idea of *PSO* can be explained by an analogy as proposed by Chen, Chang, Wang and Wong (2011),

One may consider a scenario where there is only one piece of food in the area being searched and all the birds do not know exactly where the food is. However, they increasingly know how far the food is with each iteration. The effective strategy is to share information constantly among the flock and follow the bird which is nearest to the food. In our PSO setup, each single solution is a "bird" or a "particle" in the search space. All of the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct particles where to fly. The particles fly through the problem space by following the current optimum particles.

In abbreviation, *PSO* sorts out a globally optimal solution by collecting information out of locally best solutions. A fixed size of population of flock is given. After choosing an initial position for each "bird", it begins by moving around searching out a local optimal solution. The direction and size of each movement is represented by a vector called *particle velocity*. This *particle velocity* is determined by a hybrid of the former velocity and the distance between the former position and either locally or globally optimal solutions by now.

Particle velocity reflects the information shared among flock and the action which should be taken afterwards. After every step of movement, each bird memorizes the locally better (or no-worse) solution by comparing the new solution with the local best solution in memory. In the meantime, the flock memorizes the globally better (or no-worse) solution by using by comparing the new solution with the global best solution in memory.

After a given number of steps of flock moving, if the globally best solution is observed to be stable or unchanged, we would take that solution to be the optimal.

Getting back to our design problem, using the similar framework as previous by assuming a function $\mathbb{L}$ to be minimized, the corresponding *PSO* can be implemented using the algorithm mentioned as in Chen, Chang, Wang and Wong (2011) (similar to previous algorithm, there are some tuning constants, which could be chosen arbitrarily, but they turned out not to affect much the algorithm performance.):

1. **Initialize** "particles" (or "locations" of "birds").

   (a) **Initialize** designs ("particles") $\boldsymbol{\xi}_k$ indexed by k, where $k = 1, \ldots, K$. As the number of "particles" or the size of "flock", K is given fixed (e.g. 40). For our design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^N$, each design (density) $\boldsymbol{\xi}_k$ can be identified with a multinomial vector of length N, with sum n.

   (b) **Calculate** the loss function value for each design, denoted as $\mathbb{L}_v(\boldsymbol{\xi}_k)$.

   (c) **Initialize** the *local* best designs $\mathbf{p}_k$ by $\mathbf{p}_k = \boldsymbol{\xi}_k$ for all the k from 1 to K. Here $\mathbf{p}_k$ records the "locally" memorized best "location" (design) by each individual "bird" indexed by k. A "bird" k can move through multiple locations which are reflected by a varied $\boldsymbol{\xi}_k$. After each step of movement, there is always a new $\boldsymbol{\xi}_k$ to be arrived. But $\mathbf{p}_k$ records only the best location memorized by "bird" k. After "bird" k arriving a new $\boldsymbol{\xi}_k$, a new $\mathbf{p}_k$ can be obtained by simply comparing the existed $\mathbf{p}_k$ and the new $\boldsymbol{\xi}_k$. Initially, in the memory of each "bird", there is only one "location" (design) $\boldsymbol{\xi}_k$ being memorized, hence we make that one the initialization of $\mathbf{p}_k$.

(d) **Initialize** the global (through all the k from 1 to K) best design $\mathbf{p}_g = \text{argmin}_{1\le k\le K}\mathbb{L}_\nu(\mathbf{p}_k)$.

Here $\mathbf{p}_g$ is the "globally" memorized best "location" (design) by all the "birds" indexed by k through 1 to K. In principle, $\mathbf{p}_g$ should be the best "location" (design) out of all the "locations" (designs) having been arrived by the "flock". Under the condition that $\mathbf{p}_k$ has already recorded the "locally" memorized best "location" by each "bird", a "globally" memorized best "location" $\mathbf{p}_g$ can be obtained by simply picking up the best design out of $\mathbf{p}_k$'s.

(e) **Initialize** "particle velocity" $\mathbf{v}_k$ by Uniform $(-1, 1)$ for all the k from 1 to K.

The velocity is a vector representing the speed, including direction and magnitude, of the current design towards the next design. It is determined by the relative positions of current design, local best design and global best design. Here Uniform $(-1, 1)$ is just used to initialize the $\mathbf{v}_k$'s.

2. **Move** particles, **change** velocity and **update** the globally and locally best designs $\mathbf{p}_k$ and $\mathbf{p}_g$.

(a) **Calculate** particle velocity according to

$$\mathbf{v}_{k+1} = w_k\mathbf{v}_k + c_1\boldsymbol{\beta}_1 * (\mathbf{p}_k - \boldsymbol{\xi}_k) + c_2\boldsymbol{\beta}_2 * (\mathbf{p}_g - \boldsymbol{\xi}_k), \quad k = 1, \ldots, K-1;$$
$$\mathbf{v}_1 = w_K\mathbf{v}_K + c_1\boldsymbol{\beta}_1 * (\mathbf{p}_K - \boldsymbol{\xi}_K) + c_2\boldsymbol{\beta}_2 * (\mathbf{p}_g - \boldsymbol{\xi}_K).$$

where $\mathbf{v}_k$ is the particle velocity, w_k is the inertia weight modulating the influence of the former velocity which can be chosen as an either constant or decreasing function indicating the constant and shrinking influence of former velocity. $\boldsymbol{\xi}_k$ is the current particle (design). The variables $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are ran-

dom vectors, each component independently and identically following Uniform $(0, 1)$, with the same dimension as $\boldsymbol{\xi}_k$ and are multiplied "component-wise" represented by the operator $*$. c_1 and c_2 are local and social learning factors, which were assigned fixed c_1, c_2 (e.g. $c_1 = c_2 = 2$). We do not require nonnegativity for each component of velocity. In other words, the probability mass on any specific design point could either increase or decrease. But if any component of $\mathbf{v}_{k+1}$ is smaller than -1, we will shift it to -1. And if any component of $\mathbf{v}_k$ is greater than 1, we will shift it to 1. These velocity re-assignments assure $\mathbf{v}_{k+1}$ will not go beyond a reasonable boundary (e.g. $[-1, 1]$).

(b) **Update** particle position according to

$$\begin{aligned}\boldsymbol{\xi}_{k+1}^{(new)} &= \boldsymbol{\xi}_k^{(old)} + \mathbf{v}_{k+1}, \quad k = 1, \ldots, K-1; \\ \boldsymbol{\xi}_1^{(new)} &= \boldsymbol{\xi}_K^{(old)} + \mathbf{v}_1.\end{aligned}$$

Here if any of the $\boldsymbol{\xi}_{k+1}$, where k is from 0 to $K-1$, has any element of negative, we can shift it to 0 and for any element greater than 1, we will shift it to 1. Similarly to the re-assignments of the particle velocity, these re-assignments assure $\mathbf{v}_{k+1}$ not go beyond a reasonable boundary (e.g. $[0, 1]$). After the shift, if the sum of the components of $\boldsymbol{\xi}_{k+1} = (z_{k+1}^{(1)}, z_{k+1}^{(2)}, \ldots, z_{k+1}^{(N)})$ do not equal to 1, we can standardize it by

$$\frac{\boldsymbol{\xi}_{k+1}}{\mathbf{1}'\boldsymbol{\xi}_{k+1}}.$$

Similar to the genetic algorithm, each member of the current generation can be represented by its vector $n\boldsymbol{\xi}$ of design point allocations. Any fractional allocations are rounded down. This results in a vector $\boldsymbol{S}$ with integer elements, with sums possibly less than n. If $s < n$ then $n - s$ design points are randomly

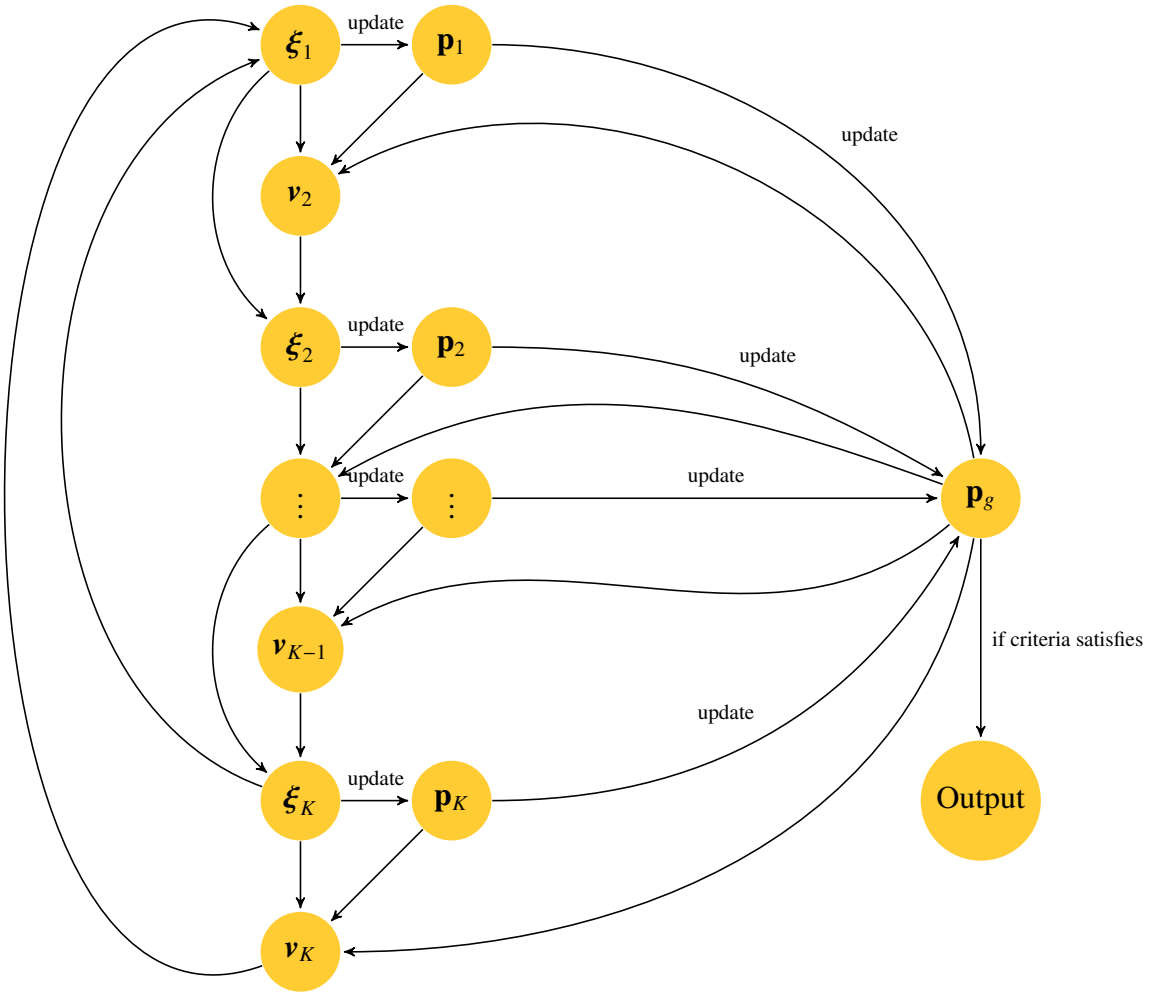
chosen from $\boldsymbol{S}$ (with replacement) and added to the design.

(c) **Calculate** the loss function value for each design, denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

(d) **Update** local best design $\mathbf{p}_k^{(new)} = \text{argmin}\{\mathbb{L}_\nu(\mathbf{p}_k^{(old)}), \mathbb{L}_\nu(\boldsymbol{\xi}_k^{(new)})\}$ and global best design $\mathbf{p}_g^{(new)} = \text{argmin}_{1\le k\le K}\mathbb{L}_\nu(\mathbf{p}_k^{(new)})$.

3. Step 2 is carried out **repeatedly** until certain stopping criteria are satisfied. For example the best design does not change for GK times of running step 2. The output $\mathbf{p}_g$ is as our best design with loss function value $\mathbb{L}_\nu(\mathbf{p}_g)$.

For each k, $\mathbf{p}_k$ is the design with minimum loss, among all locations through which the designs "moved". A diagram is given below, where the arrow without any note means "calculate":



2.4 Expected Improvement Algorithm

Expected Improvement Algorithm (EIA) method (originally from the paper of Jones, Schonlau and Welch (1998)) is a sequential optimization method to obtain a best design using only a few evaluations of the optimized function. It is extremely useful when the optimized function evaluation is very expensive or the number of opti-

mized function evaluations, say $\mathbb{L}_\nu(\boldsymbol{\xi})$, is limited.

Using the same framework as previous, the intuition is as follows. Instead of directly making "expensive" function evaluations $\mathbb{L}_\nu(\cdot)$ on various locations $\boldsymbol{\xi}_k$'s , we choose to evaluate only on the most probably optimal location. *EIA* is a sequential searching strategy such that each new evaluation point is made to maximize the expected difference between the new evaluation (a random value) and the best of the sequence of old evaluations (fixed values).

Making a predictor $\hat{\mathbb{L}}_\nu(\cdot)$ of the "expensive" function $\mathbb{L}_\nu(\cdot)$ interpolating the existing data pairs $\{\boldsymbol{\xi}_k, \mathbb{L}_\nu(\boldsymbol{\xi}_k)\}_{k=1}^m$, we can get the standard error of the predictor which provides confidence intervals on the function's value at un-sampled points. Here we intend to minimize the function $\mathbb{L}_\nu(\cdot)$. A new design $\xi^\star$ is said to make *improvement*, if it can achieve a positive value of $I(\boldsymbol{\xi}^\star) = \max(R_{\min} - R^\star, 0)$, where $R_{\min}$ is the minimum value of the sequence $\{\mathbb{L}_\nu(\boldsymbol{\xi}_k)\}_{k=1}^m$ and $R^\star = \mathbb{L}_\nu(\boldsymbol{\xi}^\star)$. Intuitively we want to maximize such improvement. Since $R^\star$, or $\mathbb{L}_\nu(\boldsymbol{\xi}^\star)$, was not yet observed, it is a random value. But if only we can get the distribution of this random value, we can make the compromise to maximize only the expected improvement, i.e. $E[\max(R_{\min} - R^\star, 0)]$.

It indicates two important tasks which should be be carried out: the choice of an appropriate model for prediction and generating the new evaluation point $\boldsymbol{\xi}^\star$, which is denoted as $\boldsymbol{\xi}_{m+1}$.

Following the recommended by Jones, Schonlau and Welch (1998), the prediction model we apply here is *DACE*, an acronym for *Design and Analysis of Computer*

Experiments, model. It is a stochastic process model named according to Sacks, Welch, Mitchell and Wynn (1989-1) and making prediction of a single point using the surrounding observations in the neighbourhood. Here we just give a brief look at *DACE* predictor and the details for the deduction can be found in Sacks, Welch, Mitchell and Wynn (1989-1) and Jones, Schonlau and Welch (1998).

Let $\{\boldsymbol{\xi}_k\}_{k=1}^m$ be the current designs for the design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^N$,with $\boldsymbol{\xi}_k = (z_k^{(1)}, z_k^{(2)}, \ldots, z_k^{(N)})$ be the corresponding values of "expensive" function $l(\boldsymbol{\xi}_k)$ or l_k (i.e. $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$ as we spoke of previously). Let $\boldsymbol{l}$ denote the m-vector of losses of designs $\{\boldsymbol{\xi}_k\}_{k=1}^m$ which is $\boldsymbol{l} = (l_1, \ldots, l_m)'$. For a new design $\boldsymbol{\xi}^\star$, DACE model, using

$$l(\boldsymbol{\xi}_k) = \mu + \epsilon(\boldsymbol{\xi}_k) \quad k = 1, \ldots, m, \tag{2.1}$$

where $\epsilon(\boldsymbol{\xi}_k) \sim N(0, \sigma^2)$ identically and independently for all k, gives the best linear unbiased predictor of $l(\boldsymbol{\xi}^\star)$ by

$$\hat{l}(\boldsymbol{\xi}^\star) = \hat{\mu} + \mathbf{r}'\mathbf{R}^{-1}(\boldsymbol{l} - \mathbf{1}\hat{\mu}).$$

The derivation can be found in Sacks, Welch, Mitchell and Wynn (1989-2). And

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\boldsymbol{l}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}},$$
$$\hat{\sigma}^2 = \frac{(\boldsymbol{l} - \mathbf{1}\hat{\mu})'\mathbf{R}^{-1}(\boldsymbol{l} - \mathbf{1}\hat{\mu})}{n},$$

are the maximum likelihood estimators of μ and σ^2 and $\mathbf{R}$ is a $n \times n$ correlation matrix for errors of current designs. And for any two designs $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$, the correlation

of the error terms of two designs is defined as

$$Corr[\epsilon(\boldsymbol{\xi}_i), \epsilon(\boldsymbol{\xi}_j)] = exp[-d(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)], \tag{2.2}$$

where the "distance" between two designs $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ is given as

$$d(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) = \sum_{h=1}^{N} \theta_h |z_i^{(h)} - z_j^{(h)}|^{p_h}, \quad \theta_h \geq 0, p_h \in [1, 2]. \tag{2.3}$$

$\mathbf{r}$ is a $n \times 1$ correlation vector between the errors of new design $\boldsymbol{\xi}^\star$ and current designs $\{\boldsymbol{\xi}_k\}_{k=1}^m$. And the *mean squared error (MSE)* $s^2(\boldsymbol{\xi}^\star)$ which is used to measure the uncertainty of the predictor of $\xi^\star$ can be written as:

$$s^2(\boldsymbol{\xi}^\star) = \sigma^2 \left[1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r} + \frac{(1 - \mathbf{1}'\mathbf{R}^{-1}\mathbf{r})^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}} \right].$$

And the full derivation can also be found in Sacks, Welch, Mitchell and Wynn (1989-2).

As suggested in Jones, Schonlau and Welch (1998), the correlation function defined in (2.2) and (2.3) has all the intuitive properties one would like it to have. In particular, when the "similarity" between $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ is small (the "distance" is large), the correlation is near one. Similarly, when the similarity between designs is large (the "distance" is small), the correlation will approach zero. The parameter θ_h in the distance formula (2.3) can be interpreted as measuring the importance or "activity" of the variable $z^{(h)}$. To see this, note that saying "variable h is active" means that even small values of $|z_i^{(h)} - z_j^{(h)}|$ should imply a low correlation between the errors $\epsilon(\boldsymbol{\xi}_i)$ and $\epsilon(\boldsymbol{\xi}_j)$. Looking at Equations (2.2) and (2.3), we see that, if θ_h is very large, then

it will indeed be true that small values of $|z_i^{(h)} - z_j^{(h)}|$ translate to large "distances" and hence low correlation. The exponent p_h is related to the smoothness of the function in coordinate direction index h, *with* $p_h = 2$ *corresponding to smooth functions and values near* 1 *corresponding to less smoothness* (see Jones, Schonlau and Welch (1998)).

It turns out that modeling the correlation in this way is so powerful that we can afford to dispense with the regression terms, replacing them with a simple constant term. This gives us the model we use ... (see Jones, Schonlau and Welch (1998)) as (2.1).

The second task can be solved by simply transforming expected improvement into a more solvable form by applying only some simple integration by parts, under the condition that the DACE model is validated and can be applied.

Let $l_{\min} = \min(l_1, \ldots, l_m)$ be the current best (i.e. minimal) function value. The random value $l(\boldsymbol{\xi})$, denoted as $\mathbf{L}$, for an un-sampled design $\boldsymbol{\xi}$ follows a normal distribution with mean and standard deviation given by the DACE predictor $\hat{l}$ and its *root mean squared error (RMSE) s*. As discussed previously, the expected improvement is given as

$$E[I(\boldsymbol{\xi})] = E[max(l_{\min} - \mathbf{L}, 0)].$$

Using integration by parts we can get a closed form of expected improvement. Actually by denoting the probability density function of $\mathbf{L}$ as $g(\cdot)$, we can deduce as

follows:

$$\begin{aligned} E[I(\boldsymbol{\xi})] &= \int_{-\infty}^{\infty} max(l_{\min} - y, 0) g(y) dy \\ &= \int_{-\infty}^{l_{\min}} (l_{\min} - y) g(y) dy \\ &= s \cdot \int_{-\infty}^{l_{\min}} (l_{\min} - y) \cdot \phi(\frac{y - \hat{l}}{s}) d(\frac{y - \hat{l}}{s}). \end{aligned}$$

Replacing $\frac{y-\hat{l}}{s}$ by t, we get

$$E[I(\boldsymbol{\xi})] = s \cdot \int_{-\infty}^{\frac{l_{\min}-\hat{l}}{s}} (l_{\min} - \hat{l} - st) \cdot \phi(t) dt.$$

Applying integration by parts, we get

$$\begin{aligned} E[I(\boldsymbol{\xi})] &= s(l_{\min} - \hat{l}) \cdot \int_{-\infty}^{\frac{l_{\min}-\hat{l}}{s}} \phi(t) dt - s^2 \cdot \int_{-\infty}^{\frac{l_{\min}-\hat{l}}{s}} t\phi(t) dt \\ &= s(l_{\min} - \hat{l}) \Phi(\frac{l_{\min} - \hat{l}}{s}) + \frac{s^2}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\frac{l_{\min}-\hat{l}}{s}} d(e^{-\frac{t^2}{2}}) \\ &= s(l_{\min} - \hat{l}) \Phi(\frac{l_{\min} - \hat{l}}{s}) + s^2 \phi(\frac{l_{\min} - \hat{l}}{s}). \end{aligned}$$

In abbreviation,

$$E[I(\boldsymbol{\xi})] \propto (l_{\min} - \hat{l}) \Phi(\frac{l_{\min} - \hat{l}}{s}) + s\phi(\frac{l_{\min} - \hat{l}}{s}). \tag{2.4}$$

Then we can get a new design $\boldsymbol{\xi}^{\star}$ maximizing $E[I(\boldsymbol{\xi})]$ and add it as a new design $\boldsymbol{\xi}_{m+1}$ into the design sequences. Afterwards we can get the global optimal design by comparing the function values $l_1, \ldots, l_m, l_{m+1}$.

We repeat the steps above by adding one more design each time. If the global op-

timal design does not change for certain number of steps, say the number is 10, we stop and output the global optimal design. After only a few steps of probable optimal point generating and evaluating, a globally optimal and stable solution can be expected.

The explicit algorithm is given below:

1. Start by randomly generating the initial set $\mathcal{D}$ of designs with size K (e.g. $K = 40$), i.e. $\mathcal{D} = \{\boldsymbol{\xi}_k\}_{k=1}^{K}$. For our design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, each design (density) $\boldsymbol{\xi}_k$ can be identified with a multinomial vector of length N, with sum n.

2. Calculate the loss function value for each design of $\mathcal{D}$ denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, where $k = 1, \ldots, K$. Denote $\boldsymbol{\xi}_{min} = \text{argmin}_{1\le k\le K}\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

3. Denote the size of $\mathcal{D}$ as K.

   (a) Calculate the loss function value for each design of $\mathcal{D}$ denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, where $k = 1, \ldots, K$.

   (b) Get a new design $\boldsymbol{\xi}^\star$ maximizing $E[I(\boldsymbol{\xi})]$, i.e.

   $$\boldsymbol{\xi}^\star = \text{argmax}_{\boldsymbol{\xi}} E[I(\boldsymbol{\xi})],$$

   and add $\boldsymbol{\xi}^\star$ as a new design $\boldsymbol{\xi}_{K+1}$ into the design sequences $\mathcal{D} = \{\boldsymbol{\xi}_k\}_{k=1}^{K+1}$.

   (c) Update $\boldsymbol{\xi}_{min}$ by using $\boldsymbol{\xi}_{min} = \text{argmin}_{1\le k\le K+1}\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

4. Step 3 is carried out repeatedly until certain stopping criteria are satisfied. For example the best design $\boldsymbol{\xi}_{min}$ does not change for G times of running step

3 (e.g. $G = 25$). The output $\boldsymbol{\xi}_{min}$ is as our best design with loss function value $\mathbb{L}_v(\boldsymbol{\xi}_{min})$.

Chapter 3

Symmetric Designs

3.1 Symmetry of Design in SLR

As discussed in Section 2.1, a discrete design space $\chi = \{\boldsymbol{x}_i\}_{i=1}^N$, where N can be arbitrarily large, can be made to approximate a continuous design space. For a *Multiple linear regression* (*MLR*) as

$$Y(\boldsymbol{x}) = \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta} + \psi(\boldsymbol{x}) + \epsilon, \text{ s.t.} \quad var(\epsilon) = \sigma^2, \int_\chi \psi^2(\boldsymbol{x})d\boldsymbol{x} \leq \tau^2/n$$

where $\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}$, the vector of covariates, and $\boldsymbol{\theta}$ is the vector of parameters.

Intuitively, if we wanted to have $E[Y] \approx \boldsymbol{f}'\boldsymbol{\theta}$, we can choose the parameter $\boldsymbol{\theta}$ to minimize an L_2-distance between the true regression response $E[Y(\boldsymbol{x})]$ and the fitted response $\boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\eta}$, i.e.

$$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\theta}} \int_\chi (E[Y(\boldsymbol{x})] - \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\eta})^2 d\boldsymbol{x}.$$

Taking the first derivative of the right hand side by $\boldsymbol{\eta}$ and make it zero, i.e.

$$-2\int_{\mathcal{X}} \boldsymbol{f}(\boldsymbol{x})\{E[Y(\boldsymbol{x})] - \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\eta}\}d\boldsymbol{x} = \mathbf{0}.$$

then according to the definition of $\psi(\boldsymbol{x})$, i.e. $\psi(\boldsymbol{x}) = E[Y(\boldsymbol{x})] - \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\eta}$,

$$\int_{\mathcal{X}} \boldsymbol{f}(\boldsymbol{x})\psi(\boldsymbol{x})d\boldsymbol{x} = \mathbf{0}.$$

For n observations $y_1, y_2, \ldots, y_n$ at locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ correspondingly, denoting $\boldsymbol{Y}$ as $(y_1, \ldots, y_n)'$, the vector of observations, and $\boldsymbol{X}$ as $(\boldsymbol{f}(\boldsymbol{x}_1), \ldots, \boldsymbol{f}(\boldsymbol{x}_n))'$, $\hat{\boldsymbol{\theta}}$ is the minimizer of $\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta}\|^2$, which can be written as $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$:

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{f}'(\boldsymbol{x})\hat{\boldsymbol{\theta}} = \boldsymbol{f}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}.$$

Replacing by $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\psi} + \boldsymbol{\epsilon}$, we get

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{f}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\psi} + \boldsymbol{\epsilon}].$$

And expanding the part for $\boldsymbol{X}'[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\psi} + \boldsymbol{\epsilon}]$ and simplifying the form we get

$$\begin{aligned}
\hat{Y}(\boldsymbol{x}) &= \boldsymbol{f}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\sum_{i=1}^{n}\boldsymbol{f}(\boldsymbol{x}_i)\{\boldsymbol{f}'(\boldsymbol{x}_i)\boldsymbol{\theta} + \psi(\boldsymbol{x}_i) + \boldsymbol{\epsilon}_i\} \\
&= \boldsymbol{f}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\{(\boldsymbol{X}'\boldsymbol{X})\boldsymbol{\theta} + \sum_{i=1}^{n}\boldsymbol{f}(\boldsymbol{x}_i)[\psi(\boldsymbol{x}_i) + \boldsymbol{\epsilon}_i]\} \\
&= \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{f}'(\boldsymbol{x})(\frac{\boldsymbol{X}'\boldsymbol{X}}{n})^{-1}\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{f}(\boldsymbol{x}_i)\psi(\boldsymbol{x}_i) + \boldsymbol{f}'(\boldsymbol{x})(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\epsilon}.
\end{aligned}$$

Denoting

$$\begin{aligned}
\boldsymbol{M}_{\xi} &= \frac{\boldsymbol{X}'\boldsymbol{X}}{n} = \int \boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\xi}(d\boldsymbol{x}), \\
\boldsymbol{b}_{\psi,\xi} &= \frac{1}{n}\boldsymbol{f}'(\boldsymbol{x})\psi(\boldsymbol{x}) = \int \boldsymbol{f}'(\boldsymbol{x})\psi(\boldsymbol{x})\boldsymbol{\xi}(d\boldsymbol{x}) \\
&= \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{f}'(\boldsymbol{x})(\frac{\boldsymbol{X}'\boldsymbol{X}}{n})^{-1}\frac{1}{n}\boldsymbol{f}'(\boldsymbol{x})\psi(\boldsymbol{x}) + \boldsymbol{f}'(\boldsymbol{x})(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\epsilon},
\end{aligned}$$

and replacing $\boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta}$ by $E[Y(\boldsymbol{x})] - \psi(\boldsymbol{x})$, we get

$$\hat{Y}(\boldsymbol{x}) = [E[Y(\boldsymbol{x})] - \psi(\boldsymbol{x})] + \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{M}_{\xi}^{-1}\boldsymbol{b}_{\psi,\xi} + \boldsymbol{f}'(\boldsymbol{x})(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\epsilon}. \tag{3.1}$$

In the case of *Simple Linear Regression (SLR)*, the fitted model is

$$E[Y(\boldsymbol{x})] = \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta} = \theta_0 + \theta_1 x,$$

and the alternative fitted model is

$$E[Y(\boldsymbol{x})] = \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{\theta} + \psi(\boldsymbol{x}) = \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 x + \psi(\boldsymbol{x}).$$

And the *Integrated Mean Squared Error* (*IMSE*) can be calculated by

$$\begin{aligned}
\text{IMSE} &= \int_{\mathcal{X}} MS\,E[\hat{Y}(\boldsymbol{x})]d\boldsymbol{x} \\
&= \int_{\mathcal{X}} E\{\hat{Y}(\boldsymbol{x}) - E[Y(\boldsymbol{x})]\}^2 d\boldsymbol{x}.
\end{aligned}$$

Plugging (3.1) in, and expanding the part inside the expectation, we get

$$\begin{aligned}\text{IMSE} &= \int E\{-\psi(\boldsymbol{x}) + \boldsymbol{f}'(\boldsymbol{x})\boldsymbol{M}_{\xi}^{-1}\boldsymbol{b}_{\psi,\xi} + \boldsymbol{f}'(\boldsymbol{x})(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\epsilon}\}^2 d\boldsymbol{x} \\ &= \int \{\boldsymbol{b}'\boldsymbol{M}^{-T}\boldsymbol{f}\boldsymbol{f}'\boldsymbol{M}^{-1}\boldsymbol{b} + \psi^2 + \boldsymbol{f}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}')\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{f}\} d\boldsymbol{x}.\end{aligned}$$

Then due to $cov(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{I}$, we can get

$$\begin{aligned}\text{IMSE} &= \int \{\boldsymbol{b}'\boldsymbol{M}^{-1}\boldsymbol{f}\boldsymbol{f}'\boldsymbol{M}^{-1}\boldsymbol{b} + \psi^2 + \frac{\sigma^2}{n}\boldsymbol{f}'(\frac{\boldsymbol{X}'\boldsymbol{X}}{n})^{-1}\boldsymbol{f}\} d\boldsymbol{x} \\ &= \int \{\boldsymbol{b}'\boldsymbol{M}^{-1}\boldsymbol{f}\boldsymbol{f}'\boldsymbol{M}^{-1}\boldsymbol{b} + \psi^2 + \frac{\sigma^2}{n}tr(\boldsymbol{f}'\boldsymbol{M}^{-1}\boldsymbol{f})\} d\boldsymbol{x}.\end{aligned}$$

Since $tr(\boldsymbol{A}\boldsymbol{B}) = tr(\boldsymbol{B}\boldsymbol{A})$, we have

$$\text{IMSE} = \int \{\boldsymbol{b}'\boldsymbol{M}^{-1}\boldsymbol{f}\boldsymbol{f}'\boldsymbol{M}^{-1}\boldsymbol{b} + \psi^2 + \frac{\sigma^2}{n}tr(\boldsymbol{M}^{-1}\boldsymbol{f}\boldsymbol{f}')\} d\boldsymbol{x}.$$

In total we have

$$\text{IMSE} = \boldsymbol{b}'_{\psi,\xi}\boldsymbol{M}_{\xi}^{-1}\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1}\boldsymbol{b}_{\psi,\xi} + \int \psi^2(\boldsymbol{x}) d\boldsymbol{x} + \frac{\sigma^2}{n}tr(\boldsymbol{M}_{\xi}^{-1}\boldsymbol{A}),$$

where

$$\begin{aligned}\boldsymbol{A} &:= \int \boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}'(\boldsymbol{x}) d\boldsymbol{x}, \\ \boldsymbol{M}_{\xi} &:= \int \boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}'(\boldsymbol{x}) m(\boldsymbol{x}) d\boldsymbol{x}, \\ \boldsymbol{b}_{\psi,\xi} &:= \int \boldsymbol{f}(\boldsymbol{x})\psi(\boldsymbol{x}) m(\boldsymbol{x}) d\boldsymbol{x}.\end{aligned}$$

By observing the expression of $IMSE(\psi, \boldsymbol{\xi})$, we can see how each factor of $\boldsymbol{\xi}$, ψ and $\boldsymbol{f}$ influence the IMSE. Here $m(\boldsymbol{x})$ is the associated density function of measure

$\boldsymbol{\xi}$, i.e. $m(\boldsymbol{x})d\boldsymbol{x} = \boldsymbol{\xi}(d\boldsymbol{x})$. The reason why it is necessary to talk about the *density* in our problem is as discussed in Section 2.1, a probability mass function on a discrete design space $\boldsymbol{\chi} = \{\boldsymbol{x}_i\}_{i=1}^N$, or a *design*, can approximate a *density* on a continuous design space when N is large enough. Hence it would be meaningful to talk about *design density* in more general cases.

Now we intend to find a design $\boldsymbol{\xi}$ to minimize the IMSE maximized by some ψ i.e. minimize the worst case of ψ, where the integral of ψ is bounded above by τ^2/n. The design here is called a *Q-optimal minimax design*.

After imposing the restriction that $\int \psi^2(\boldsymbol{x})d\boldsymbol{x} \leq \tau^2/n$ for a given constant τ, we can write the maximum of IMSE as (according to Heo, Schmuland and Wiens (2001))

$$max_{\psi}\text{IMSE} = \frac{\sigma^2 + \tau^2}{n} \cdot [(1-\nu)tr(\boldsymbol{A}\boldsymbol{M}_{\boldsymbol{\xi}}^{-1}) + \nu ch_{max}(\boldsymbol{K}_{\boldsymbol{\xi}}\boldsymbol{H}_{\boldsymbol{\xi}}^{-1})]$$

where

$$\begin{aligned}
\boldsymbol{H}_{\boldsymbol{\xi}} &:= \boldsymbol{M}_{\boldsymbol{\xi}}\boldsymbol{A}^{-1}\boldsymbol{M}_{\boldsymbol{\xi}}, \\
\boldsymbol{K}_{\boldsymbol{\xi}} &:= \int \boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}'(\boldsymbol{x})m^2(\boldsymbol{x})d\boldsymbol{x}, \\
\nu &:= \frac{\tau^2}{\sigma^2 + \tau^2}.
\end{aligned}$$

Hence we have $max_{\psi}\text{IMSE} \propto \mathbb{L}_{\nu}(\boldsymbol{\xi})$, where

$$\mathbb{L}_{\nu}(\boldsymbol{\xi}) := (1-\nu)tr(\boldsymbol{A}\boldsymbol{M}_{\boldsymbol{\xi}}^{-1}) + \nu ch_{max}(\boldsymbol{K}_{\boldsymbol{\xi}}\boldsymbol{H}_{\boldsymbol{\xi}}^{-1}). \tag{3.2}$$

Note that it is the choice of experimenter to determine ν, the relative importance of

errors due to bias rather than variance. The limiting cases $\nu = 0$ and $\nu = 1$ correspond to 'pure variance' and 'pure bias' problems, respectively.

In straight line regression on $\chi = [-1, 1]$,

$$\boldsymbol{M}_{\xi} = \begin{pmatrix} 1 & \mu_1 \\ \mu_1 & \mu_2 \end{pmatrix}.$$

with

$$\mu_k(\boldsymbol{\xi}) = \int_{\chi} x^k m(x) dx,$$

and

$$\boldsymbol{A} = \int_{\chi} \boldsymbol{f}(\boldsymbol{x}) \boldsymbol{f}'(\boldsymbol{x}) dx = \int_{-1}^{1} \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 & x \end{pmatrix} dx$$

$$= \begin{pmatrix} x & x^2/2 \\ x^2/2 & x^3/3 \end{pmatrix} |_{-1}^{1} = \begin{pmatrix} 2 & 0 \\ 0 & 2/3 \end{pmatrix}.$$

Ye and Zhou (2007) has given an important result about symmetry of the *Q-optimal minimax design* for the case of the simple linear regression. That is a *Q-optimal minimax design* is necessarily symmetric if we require that $\int_{\chi} xm(x)dx = 0$. At the end, they propose that even if the condition $\int_{\chi} xm(x)dx = 0$ is removed, the claim may still be true that *Q-optimal minimax design* is symmetric. However they cannot provide a proof. Here is where we start investigating the question of whether robust designs for straight line regression are necessarily symmetric by proposing the analytic attempts which have been made so far.

3.2 Symmetry of Design in SLR for Pure Variance

We can start to verify that $tr(\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1})$ is minimized by a symmetric design, where $\boldsymbol{A}$ and $\boldsymbol{M}_{\xi}$ are defined in the previous section. If $\nu = 0$, we are interested in the case of "pure variance". We have

$$tr(\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1}) = 2 + 2\frac{1/3 + \mu_1^2}{\mu_2 - \mu_1^2}.$$

After observing the forms that (which was indicated from Huber and Rochetti (2009)),

$$\mu_1 = \int_{-1}^{1} xm(x)dx,$$
$$\mu_2 = \int_{-1}^{1} x^2 m(x)dx,$$

an obvious and only solution is a design putting the equally (minimizing $|\mu_1|$) the design points on both ends (maximizing μ_2) of $[-1, 1]$.

Actually for $\mu_2 - \mu_1^2 \geq 0$, in order to minimize $tr(\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1})$, we want to minimize μ_2 and maximize $\mu_2 - \mu_1^2 \geq 0$ at the same time. And $\mu_1 = 0$ is the global minimizer of $tr(\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1})$. Under the restriction of $\mu_1 = 0$ we can rewrite $tr(\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1}) = 2 + \frac{2}{3\mu_2}$. then we find μ_2.

We can also show the symmetry in the other way. Firstly we write

$$\Phi(\boldsymbol{\xi}) := tr(\boldsymbol{A}\boldsymbol{M}_{\boldsymbol{\xi}}^{-1})$$

$$\phi(t) := \Phi(\boldsymbol{\xi}_t),$$

where

$$\boldsymbol{\xi}_t = (1-t)\boldsymbol{\xi}_0 + t\boldsymbol{\xi}_1 \text{ and } t \in [0,1].$$

Then

$$\phi(t) = \Phi(\boldsymbol{\xi}_t) = tr(AM_{\boldsymbol{\xi}_t}^{-1}) = tr(\boldsymbol{A}^{1/2}\boldsymbol{M}_{\xi_t}^{-1}\boldsymbol{A}^{1/2}) = \sum_{i=1,2} \boldsymbol{e}_i'\boldsymbol{A}^{1/2}\boldsymbol{M}_{\boldsymbol{\xi}_t}^{-1}\boldsymbol{A}^{1/2}\boldsymbol{e}_i.$$

Lemma 2 of Wiens (1993) stated that for fixed $i = 1, 2$, if $\boldsymbol{A}^{1/2}$ and $\boldsymbol{M}_{\boldsymbol{\xi}_t}$ are matrices each of whose elements is a linear function of a real variable t, and if $\boldsymbol{M}_{\boldsymbol{\xi}_t}$ is positive definite, then $\phi(t)$ is a sum of two convex functions of t hence convex.

And from the property of convexity, one can check that

$$\Phi(\boldsymbol{\xi}_t) = \phi(t) = \phi((1-t)\cdot 0 + t\cdot 1) \leq (1-t)\phi(0) + t\phi(1) = (1-t)\Phi(\boldsymbol{\xi}_0) + t\Phi(\boldsymbol{\xi}_1).$$

Now we just need to verify the convexity of $\phi(t)$. In fact, each of the elements of

$\boldsymbol{M}_{\xi_t}$ is indeed a linear combination of t,

$$\begin{aligned}\mu_{k,\xi_t} &= \int_{\mathcal{X}} x^k d(\boldsymbol{\xi}_t(x)) \\ &= \int_{\mathcal{X}} x^k((1-t)d(\boldsymbol{\xi}_0(x)) + td(\boldsymbol{\xi}_1(x))) \\ &= (1-t)\int_{\mathcal{X}} x^k m_0(x)dx + t\int_{\mathcal{X}} x^k m_1(x)dx,\end{aligned}$$

and $\boldsymbol{A}^{1/2}$ is independent of t. According to Sylvester's criterion, if all of the leading principal minors are positive, we can claim $\boldsymbol{M}_{\xi_t}$ as positive definite. Since μ_{2,ξ_t} and $\mu_{2,\xi_t} - \mu_{1,\xi_t}^2$ are both greater than 0 for $t \in [0,1]$. Hence $\boldsymbol{M}_{\xi_t}$ is positive definite. Here we have verified the convexity of $\phi(t)$.

Denoting $\boldsymbol{\xi}_0$ as any design on a symmetric interval $\mathcal{X}$, w.l.o.g. $\mathcal{X} = [-1,1]$, with density $m(x)$ and $\boldsymbol{\xi}_1$ as the design with density $m(-x)$. The design $\boldsymbol{\xi}_1$ used here is just to replace x with $-x$ everywhere. And the design $\boldsymbol{\xi}_1$ has the same loss as the original $\boldsymbol{\xi}_0$. Hence $\tilde{\boldsymbol{\xi}}$ with associated density as $\tilde{m}(x) := \frac{m(x)+m(-x)}{2}$ is a symmetric design. And

$$\Phi(\tilde{\boldsymbol{\xi}}) \leq 1/2\Phi(\boldsymbol{\xi}_0) + 1/2\Phi(\boldsymbol{\xi}_1) = \Phi(\boldsymbol{\xi}_0) = \Phi(\boldsymbol{\xi}_1).$$

Here we have already shown that for the case of the simple linear regression, *Q-optimal minimax design* is symmetric, when $\nu = 0$. In other words, for the simple linear regression, considering only the variance, a *Q-optimal minimax design* is necessarily symmetric.

3.3 Symmetry of Design in SLR for Pure Bias

Secondly, we can let $\nu = 1$ and think about only the bias part, $ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1})$. For an arbitrary measure $\boldsymbol{\xi}(x)$ associated to a density $m(x)$, we can have a new measure $\tilde{\boldsymbol{\xi}}(x)$ associated to the density $\tilde{m}(x) := \frac{m(x)+m(-x)}{2}$, which is obviously symmetric. We have already shown in Section 3.2, for the case of pure variance, design $\tilde{\boldsymbol{\xi}}$ tends to have no greater loss value than $\boldsymbol{\xi}$. If we can show the measure $\tilde{\boldsymbol{\xi}}(x)$ will always achieve a no greater $ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1})$, i.e. $ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}) \geq ch_{max}(\boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1})$, the claim is true that any arbitrary design density $m(x)$ can be *improved* by taking $\tilde{m}(x) = \frac{m(x)+m(-x)}{2}$.

According to basic Algebra, we know that $\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1} - \boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1}$ being $p.s.d.$ would be a sufficient condition for $ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}) \geq ch_{max}(\boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1})$. We have

$$\boldsymbol{K}_{\xi} = \begin{pmatrix} k_0 & k_1 \\ k_1 & k_2 \end{pmatrix}$$

where

$$k_i = \int_{-1}^{1} x^i m^2(x) dx.$$

We can also get that

$$\boldsymbol{K}_{\tilde{\xi}} = \begin{pmatrix} \tilde{k}_0 & \tilde{k}_1 \\ \tilde{k}_1 & \tilde{k}_2 \end{pmatrix},$$

where

$$\tilde{k}_i = \int_{-1}^{1} x^i \tilde{m}^2(x) dx.$$

Knowing that $\tilde{k}_1 = \int_{-1}^{1} x\tilde{m}^2(x)dx = 0$, and

$$\int_{-1}^{1} m^2(-x)dx = \int_{-1}^{1} m^2(x)dx$$
$$\int_{-1}^{1} x^2m^2(-x)dx = \int_{-1}^{1} x^2m^2(x)dx$$

we can get

$$\begin{aligned}\boldsymbol{K}_{\tilde{\xi}} &= \begin{pmatrix} \int_{-1}^{1} \tilde{m}^2(x)dx & 0 \\ 0 & \int_{-1}^{1} x^2\tilde{m}^2(x)dx \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2}\int_{-1}^{1}[m^2(x) + m(x)m(-x)]dx & 0 \\ 0 & \frac{1}{2}\int_{-1}^{1} x^2[m^2(x) + m(x)m(-x)]dx \end{pmatrix}.\end{aligned}$$

And we have

$$\begin{aligned}\boldsymbol{H}_{\tilde{\xi}}^{-1} &= \begin{pmatrix} h_0 & h_1 \\ h_1 & h_2 \end{pmatrix} \\ &= \frac{1}{(\mu_2 - \mu_1^2)^2}\begin{pmatrix} \frac{2}{3}\mu_1^2 + 2\mu_2^2 & -2\mu_1\mu_2 - \frac{2}{3}\mu_1 \\ -2\mu_1\mu_2 - \frac{2}{3}\mu_1 & 2\mu_1^2 + \frac{2}{3} \end{pmatrix}.\end{aligned}$$

And

$$\begin{aligned}\boldsymbol{H}_{\tilde{\xi}}^{-1} &= \begin{pmatrix} \tilde{h}_0 & \tilde{h}_1 \\ \tilde{h}_1 & \tilde{h}_2 \end{pmatrix} \\ &= \begin{pmatrix} 2 & 0 \\ 0 & \frac{2}{3\tilde{\mu}_2^2} \end{pmatrix}\end{aligned}$$

where

$$\tilde{\mu}_i = \int_{-1}^{1} x^i \tilde{m}(x) dx.$$

We can show that for a density as $m(x) = \frac{3}{4}x^2 + \frac{1}{4} + 0.45x$, the $\boldsymbol{K}_{\boldsymbol{\xi}}\boldsymbol{H}_{\boldsymbol{\xi}}^{-1} - \boldsymbol{K}_{\tilde{\boldsymbol{\xi}}}\boldsymbol{H}_{\tilde{\boldsymbol{\xi}}}^{-1}$ is not p.s.d. Hence that sufficient condition may not be fulfilled.

Since the sufficient condition mentioned above seems too strong, we come back to checking the original form of the eigenvalues. For the $\boldsymbol{K}_{\tilde{\boldsymbol{\xi}}}\boldsymbol{H}_{\tilde{\boldsymbol{\xi}}}^{-1}$, i.e.

$$\begin{aligned}
\boldsymbol{K}_{\tilde{\boldsymbol{\xi}}}\boldsymbol{H}_{\tilde{\boldsymbol{\xi}}}^{-1} &= \begin{pmatrix} 2\tilde{k}_0 & 0 \\ 0 & \frac{2\tilde{k}_2}{3\tilde{\mu}_2^2} \end{pmatrix} \\
&= \begin{pmatrix} \int_{-1}^{1}[m^2(x) + m(x)m(-x)]dx & 0 \\ 0 & \frac{\int_{-1}^{1} x^2[m^2(x)+m(x)m(-x)]dx}{3[\int_{-1}^{1} x^2[\frac{m(x)+m(-x)}{2}]dx]^2} \end{pmatrix}.
\end{aligned}$$

Is there a relationship that one eigenvalue larger than the other one? Actually in many cases, the first eigenvalue is larger than the second one, i.e. the inequality holds as

$$\int_{-1}^{1}[m^2(x) + m(x)m(-x)]dx \geq \frac{4\int_{-1}^{1} x^2[m^2(x) + m(x)m(-x)]dx}{3[\int_{-1}^{1} x^2[m(x) + m(-x)]dx]^2}.$$

However the equivalent inequality

$$\frac{LHS}{RHS} = \frac{\int_{-1}^{1} m(x)[m(x) + m(-x)]dx \cdot 3 \cdot [\int_{-1}^{1} x^2[m(x) + m(-x)]dx]^2}{4\int_{-1}^{1} x^2 m(x)[m(x) + m(-x)]dx} \geq 1$$

does not necessarily hold. While $m(x) = \frac{n}{2}\chi_{[-\frac{1}{n},\frac{1}{n}]}(x)$, one can easily show that the otherwise if n is large (not necessarily to go infinity, e.g. $n = 100$ is enough to sabotage the above inequality).

Now we start to compare directly the maximal eigenvalues of $\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}$ and $\boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1}$:

$$\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1} = \boldsymbol{K}_{\xi}\boldsymbol{M}_{\xi}^{-1}\boldsymbol{A}\boldsymbol{M}_{\xi}^{-1}.$$

Abbreviating $\boldsymbol{H}_{\xi}^{-1}$ and $\boldsymbol{K}_{\xi}$ ($\boldsymbol{H}_{\tilde{\xi}}^{-1}$ and $\boldsymbol{K}_{\tilde{\xi}}$ could be abbreviated similarly) by

$$\boldsymbol{H}_{\xi}^{-1} = \begin{pmatrix} h_0 & h_1 \\ h_1 & h_2 \end{pmatrix},$$

$$\boldsymbol{K}_{\xi} = \begin{pmatrix} k_0 & k_1 \\ k_1 & k_2 \end{pmatrix}.$$

Then

$$\begin{aligned} \boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1} &= \begin{pmatrix} k_0 & k_1 \\ k_1 & k_2 \end{pmatrix}\begin{pmatrix} h_0 & h_1 \\ h_1 & h_2 \end{pmatrix} \\ &= \begin{pmatrix} k_0h_0 + k_1h_1 & k_0h_1 + k_1h_2 \\ k_1h_0 + k_2h_1 & k_1h_1 + k_2h_2 \end{pmatrix}. \end{aligned}$$

From here we can get

$$
\begin{aligned}
|\lambda I - \boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}| &= \begin{pmatrix} \lambda - (k_0h_0 + k_1h_1) & -k_0h_1 - k_1h_2 \\ -k_1h_0 - k_2h_1 & \lambda - (k_1h_1 + k_2h_2) \end{pmatrix} \\
&= [\lambda - (k_0h_0 + k_1h_1)][\lambda - (k_1h_1 + k_2h_2)] - \\
&\quad -(k_0h_1 + k_1h_2)(k_1h_0 + k_2h_1) \\
&= \lambda^2 - (k_0h_0 + 2k_1h_1 + k_2h_2)\lambda + \\
&\quad +k_1^2h_1^2 + k_0h_0k_2h_2 - k_0k_2h_1^2 - k_1^2h_0h_2 \\
&= \lambda^2 - (k_0h_0 + 2k_1h_1 + k_2h_2)\lambda + (k_1^2 - k_0k_2)(h_1^2 - h_0h_2).
\end{aligned}
$$

And the maximal eigenvalue for the equation above has the expression of

$$
\begin{aligned}
&ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}) \\
&= \frac{(k_0h_0 + 2k_1h_1 + k_2h_2) + \sqrt{(k_0h_0 + 2k_1h_1 + k_2h_2)^2 - 4(k_1^2 - k_0k_2)(h_1^2 - h_0h_2)}}{2} \\
&= \frac{(k_0h_0 + 2k_1h_1 + k_2h_2) + \sqrt{(k_0h_0 - k_2h_2)^2 + 4(k_0h_1 + k_1h_2)(h_0k_1 + h_1k_2)}}{2}.
\end{aligned}
$$

Hence for the $\tilde{\xi}$ case,

$$
ch_{max}(\boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1}) = \frac{(\tilde{k}_0\tilde{h}_0 + \tilde{k}_2\tilde{h}_2) + |\tilde{k}_0\tilde{h}_0 - \tilde{k}_2\tilde{h}_2|}{2}.
$$

However, when we take $m(x) = \frac{3}{4}x^2 + \frac{1}{4} + 0.45x$, we found that $ch_{max}(\boldsymbol{K}_{\tilde{\xi}}\boldsymbol{H}_{\tilde{\xi}}^{-1}) = 1.20$ which is larger than $ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}) = 1.17$. Hence the claim is not true that an arbitrary design density $m(x)$ can be *improved* by taking $\tilde{m}(x) = \frac{m(x)+m(-x)}{2}$.

After giving this counterexample, we may need take a new direction of the design symmetrization. Actually if we can show for a symmetric density with $\tilde{m}(x) = \frac{m(x)+m(-x)}{2}$, if

$$\frac{d}{dt}\Phi((1-t)m(x) + t\tilde{m}(x))|_{t=0}$$

is negative, then the loss decreases as one starts to move away from a nonsymmetric design, in the direction of its symmetrization.

For $t = 0$, the term of $(1-t)m(x) + t\tilde{m}(x)$ is actually $m(x)$. For any $t > 0$, replacing $\tilde{m}(x)$ by $\frac{m(x)+m(-x)}{2}$, that term becomes $(1-\frac{t}{2})m(x) + \frac{t}{2}m(-x)$. Our problem is equivalent to verify whether $\Phi(m(x)) - \Phi((1-\frac{t}{2})m(x) + \frac{t}{2}m(-x)) \geq 0$, when $t \to 0^+$. It can be further simplified by showing for all x

$$\Phi(m(x)) - \Phi((1-t)m(x) + tm(-x)) \geq 0, \quad t \to 0^+.$$

Denoting the new measure as $\xi_t^\star$ associated to the density $(1-t)m(x) + tm(-x)$, in order to show that $ch_{max}(\boldsymbol{K}_\xi) \geq ch_{max}(\boldsymbol{K}_{\xi_t^\star})$ when $t \to 0^+$, there are several different methods.

The first one is to check whether $\boldsymbol{K}_\xi \boldsymbol{H}_\xi^{-1} - \boldsymbol{K}_{\xi_t^\star} \boldsymbol{H}_{\xi_t^\star}^{-1}$ is *p.s.d.*. Actually, it seems not to be the case. In fact, using the same density $m(x) = \frac{3}{4}x^2 + \frac{1}{4} + 0.45x$ above, taking even a very small value of t, say $t = 0.0001$, we cannot assure $\boldsymbol{K}_\xi \boldsymbol{H}_\xi^{-1} - \boldsymbol{K}_{\xi_t^\star} \boldsymbol{H}_{\xi_t^\star}^{-1}$ *p.s.d.*.

Now we start to compare directly the maximal eigenvalues of $\boldsymbol{K}_\xi \boldsymbol{H}_\xi^{-1}$ and $\boldsymbol{K}_{\xi_t^\star} \boldsymbol{H}_{\xi_t^\star}^{-1}$.

After carrying out the similar steps for ξ, denoting

$$\boldsymbol{H}_{\xi_t^\star}^{-1} = \begin{pmatrix} \eta_0 & \eta_1 \\ \eta_1 & \eta_2 \end{pmatrix},$$
$$\boldsymbol{K}_{\xi_t^\star} = \begin{pmatrix} \kappa_0 & \kappa_1 \\ \kappa_1 & \kappa_2 \end{pmatrix},$$

we can easily get the similar form of maximal eigenvalue for $\boldsymbol{K}_{\xi_t^\star}\boldsymbol{H}_{\xi_t^\star}^{-1}$ as

$$\begin{aligned} & ch_{max}(\boldsymbol{K}_{\xi_t^\star}\boldsymbol{H}_{\xi_t^\star}^{-1}) \\ = & \frac{(\kappa_0\eta_0 + 2\kappa_1\eta_1 + \kappa_2\eta_2) + \sqrt{(\kappa_0\eta_0 - \kappa_2\eta_2)^2 + 4(\kappa_0\eta_1 + \kappa_1\eta_2)(h_0\kappa_1 + \eta_1\kappa_2)}}{2}. \end{aligned}$$

We may want to compare it term by term with

$$\begin{aligned} & ch_{max}(\boldsymbol{K}_{\xi}\boldsymbol{H}_{\xi}^{-1}) \\ = & \frac{(k_0h_0 + 2k_1h_1 + k_2h_2) + \sqrt{(k_0h_0 - k_2h_2)^2 + 4(k_0h_1 + k_1h_2)(h_0k_1 + h_1k_2)}}{2}. \end{aligned}$$

After some try, I guess for the simplest cases we can only decompose the comparison above by claiming: $k_0h_0 + 2k_1h_1 + k_2h_2 \geq \kappa_0\eta_0 + 2\kappa_1\eta_1 + \kappa_2\eta_2$ and $(k_0h_0 - k_2h_2)^2 + 4(k_0h_1 + k_1h_2)(h_0k_1 + h_1k_2) \geq (\kappa_0\eta_0 - \kappa_2\eta_2)^2 + 4(\kappa_0\eta_1 + \kappa_1\eta_2)(h_0\kappa_1 + \eta_1\kappa_2)$. In fact there is an example with $(k_0h_1 + k_1h_2)(h_0k_1 + h_1k_2) \leq (\kappa_0\eta_1 + \kappa_1\eta_2)(h_0\kappa_1 + \eta_1\kappa_2)$, which indicates further term decomposition might not be possible.

Chapter 4

Optimal Design for SLR

In this chapter, we will numerically investigate the symmetry problem of *simple linear regression (SLR)* using the three algorithms introduced in Chapter 2. Here the loss function is given as (3.2). And for the convenience of comparison of three algorithms, I will choose some parameters for the general framework in Section 1.1. Here we will choose $N = 10$ and $n = 10$, then the design space is as $\chi = \{-1.00, -0.78, -0.56, -0.33, -0.11, 0.11, 0.33, 0.56, 0.78, 1.00\}$. The reason why we cannot make here n and N very large is due to the method of *Expected Improvement*. As indicated in Section 2.4, in order to save the expense of evaluating the *optimized function*, we need to evaluate the *expected improvement function* (2.4) more. And (2.4) will be evaluated as many as $\binom{n+N-1}{N-1}$ times, which could be very large when N and n are chosen to be large. In other words, in *EIA* algorithm, there is a trade-off of sacrificing the time of evaluating *expected improvement function* in return for saving the expense of evaluating *optimized function*.

The *optimized function* here is our loss function, whose form for the case of simple linear regression is given as (3.2): after determining the value of ν and the design

$\boldsymbol{\xi}$, we can easily calculate out the value of loss. For each algorithm, an optimized design, which minimizes the loss function, is obtained.

4.1 Genetic Algorithm

As mentioned at the beginning of this chapter, we would like to get an $n = 10$ point design minimizing the expression (3.2). The algorithm is as indicated in Section 2.2 with the tuning parameters chosen as follows. We use generations of size $K = 40$, and vary $P_{mutation}$ linearly from 0 to 0.5. We also consider $P_{crossover} = 0.90$, $P_{elite} = 0.05$, $G = 200$. The algorithm restricted to the SLR case is as follows (for a detailed explanation of each term, please see Section 2.2):

1. Randomly generate the first 'generation' of fixed $K = 40$ designs, $\boldsymbol{\xi}_1, \ldots \boldsymbol{\xi}_K$, each of which can be identified with a multinomial vector of length N, with sum n.

2. Evaluate the loss R_k for each design $\boldsymbol{\xi}_k$, viz. $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, $k = 1, 2, \ldots, K$, and the corresponding 'fitness levels'

$$\text{fitness}_k = \frac{1}{(R_k - 0.99R_{\min})^2}, k = 1, 2, \ldots, K.$$

   where $R_{\min} = \min_{1 \le k \le K} R_k$. Then we scale the fitness levels to form a probability distribution ϕ_k,

$$\phi_k = \frac{\text{fitness}_k}{\sum_{j=1}^{K} \text{fitness}_j}, k = 1, 2, \ldots, K.$$

3. Form a new generation of K designs to replace the current generation.

(a) Using $P_{elite} = .05$, we get that $N_{elite} = KP_{elite} = 2$. The remaining $K - N_{elite} = 38$ members are formed by crossover and mutation.

(b) Crossover proceeds as follows, where $P_{crossover} = 0.90$:

- Choose two members of the current generation to be parents with probability proportional to their fitness level: If ζ_1,$\zeta_2 \sim$ independent Uniform $(0, 1)$, then choose to be parents the current generation members $i_1^\star$ and $i_2^\star$, where

$$i_1^\star = \min\{i : \sum_{j=1}^{i} \phi_j \geq \zeta_1\};$$
$$i_2^\star = \min\{i : \sum_{j=1}^{i} \phi_j \geq \zeta_2\}.$$

  The same parent can be chosen twice without posing difficulties for the algorithm.

- With probability $1 - P_{crossover} = 0.10$, the child is identical to the fittest parent.

- With probability $P_{crossover} = 0.90$, the parents both contribute towards the child, in the following manner. Each member of the current generation can be represented by its vector $n\boldsymbol{\xi}$ of design point allocations. The two vectors of allocations arising from the parents are averaged, and then any fractional allocations are rounded down. This results in a vector S with integer elements, with sum s possibly less than n. If $s < n$ then $n - s$ design points are randomly chosen from S (with replacement) and added to the design. The child formed in this way is added to the new generation.

(c) Mutation is applied independently to each child. With probability $P_{mutation}$,

where $P_{mutation}$ changes linearly from 0 to 0.5, κ elements of child design are randomly chosen, and replaced by a multinomial vector of length κ with the same sum as the elements being replaced. The value of $\kappa = 4$ is chosen by the user. With probability $1 - P_{mutation}$ we do nothing.

4. Step 3 is carried out repeatedly until the next generation has been formed. We run the algorithm until the best design does not change for $G = 200$ generations.

As is shown in Figure 4.1, we use different value of ν to investigate the optimal design under different weight of bias. we can see from (f) that, the loss deceases by generation at the beginning and remains fixed after certain number of generations. From (a) to (e), we see how the value of ν, i.e. the weight on bias, influences the minimal design. Considering only the case of *'pure variance'*, we are going to split our observations on the both end of the design space χ equally. With the increase of weight on bias, the observations will become more and more equally distributed over the whole design space. And for the case of *'pure bias'*, all the observations should distribute equally over the whole design space. Last but not the least, the result here indicates the claim which was raised at the end of Section 3.1 may probably be true: for the simple linear regression, even if the condition $\int_{\chi} xm(x)dx = 0$ is removed, a Q-optimal minimax design is still symmetric.

4.2 Particle Swarm Optimization

Using the same framework as last section, we would like to get an $n = 10$ point design minimizing the expresssion (3.2). As indicated in Section 2.3, we need to

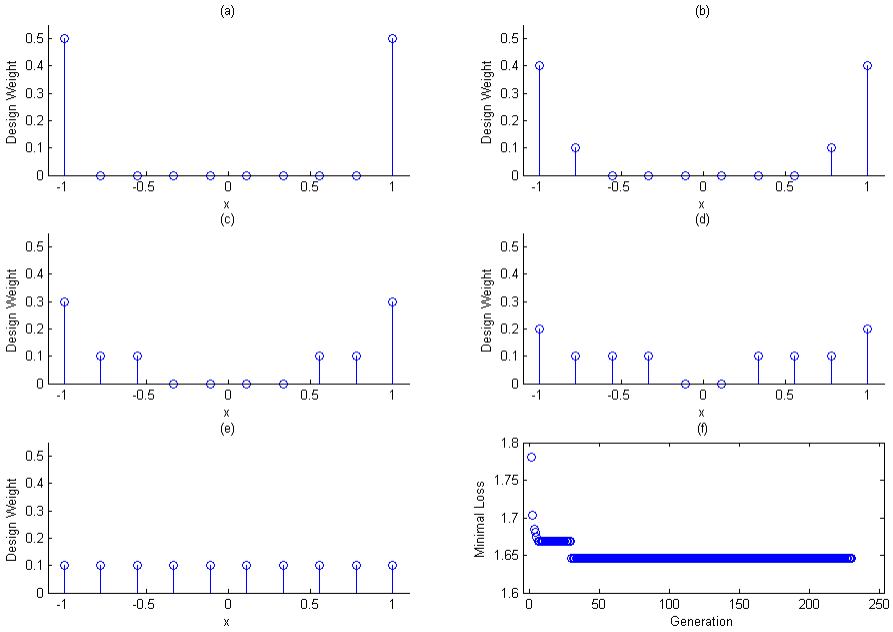


Figure 4.1: The minimax design with $n = 10$, $N = 10$, $K = 40$ generated by Genetic Algorithm. (a) $\nu = 0.0$, $loss = 2.67$; (b) $\nu = 0.25$, $loss = 2.21$; (c) $\nu = 0.50$, $loss = 1.65$; (d) $\nu = 0.75$, $loss = 0.99$; (e) $\nu = 1.0$, $loss = 0.20$. (f) shows minimal loss against generation for the case of $\nu = 0.5$.

specify some parameters before using the algorithm. Taking the suggestions from a research paper Kang, Cai, Yan and Liu (2008), we would like to choose $K = 200$ as the size of the initial designs (*'particles'*), and decrease w_k linearly from 0.9 to 0.4. We all consider $c_1 = c_2 = 1.49$, $G = 500$. The algorithm restricted to the SLR case is as follows (for a detailed explanation of each term, please see Section 2.3):

1. Initialize design.

   (a) Choose initial designs $\boldsymbol{\xi}_k$, represented by a N-dimensional vector, indexed by k, where $k = 1, \ldots, K$. The number of initial designs K is given fixed as $K = 200$. Each design $\boldsymbol{\xi}_k$ can be identified with a multinomial vector of length N, with sum n.

   (b) Calculate the loss function value for each design, denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

   (c) Initialize the local best designs $\mathbf{p}_k$ by $\mathbf{p}_k = \boldsymbol{\xi}_k$ for all the k from 1 to K.

   (d) Initialize the global (through all the k from 1 to K) best design $\mathbf{p}_g = \text{argmin}_{1 \le k \le K} \mathbb{L}_\nu(\mathbf{p}_k)$.

   (e) Initialize velocity $\mathbf{v}_k$, represented by a N-dimensional vector, by Uniform $(-1, 1)$ for all k from 1 to K.

2. Move designs, change velocity and update the globally and locally best designs $\mathbf{p}_k$ and $\mathbf{p}_g$.

   (a) Calculate design velocity according to

$$\mathbf{v}_{k+1} = w_k \mathbf{v}_k + c_1 \boldsymbol{\beta}_1 * (\mathbf{p}_k - \boldsymbol{\xi}_k) + c_2 \boldsymbol{\beta}_2 * (\mathbf{p}_g - \boldsymbol{\xi}_k), \quad k = 1, \ldots, K-1;$$

$$\mathbf{v}_1 = w_K \mathbf{v}_K + c_1 \boldsymbol{\beta}_1 * (\mathbf{p}_K - \boldsymbol{\xi}_K) + c_2 \boldsymbol{\beta}_2 * (\mathbf{p}_g - \boldsymbol{\xi}_K).$$

where $\mathbf{v}_k$ is the velocity, and w_k decrease linearly from 0.9 to 0.4. $\boldsymbol{\xi}_k$ is the current design. The variables $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are N-dimensional random vectors, each component independently and identically following Uniform $(0, 1)$, with the same dimension as $\boldsymbol{\xi}_k$ and are multiplied "componentwise" represented by the operator $*$. c_1 and c_2 are assigned fixed c_1, c_2 (e.g. $c_1 = c_2 = 1.49$). If any component of $\mathbf{v}_{k+1}$ is smaller than -1, we will shift it to -1. And if any component of $\mathbf{v}_k$ is greater than 1, we will shift it to 1.

(b) Update new designs according to

$$\boldsymbol{\xi}_{k+1}^{(new)} = \boldsymbol{\xi}_k^{(old)} + \mathbf{v}_{k+1}, \quad k = 1, \ldots, K-1;$$
$$\boldsymbol{\xi}_1^{(new)} = \boldsymbol{\xi}_K^{(old)} + \mathbf{v}_1.$$

Here if any of the $\boldsymbol{\xi}_{k+1}$, where k is from 0 to $K-1$, has any negative element, we can shift it to 0 and for any element greater than 1, we will shift it to 1. After the shift, if the sum of the components of $\boldsymbol{\xi}_{k+1} = (z_{k+1}^{(1)}, z_{k+1}^{(2)}, \ldots, z_{k+1}^{(N)})$ do not equal to 1, we can standardize it by

$$\frac{\boldsymbol{\xi}_{k+1}}{\mathbf{1}'\boldsymbol{\xi}_{k+1}}.$$

Similar to the genetic algorithm, each member of the current generation can be represented by its vector $n\boldsymbol{\xi}$ of design point allocations. Any fractional allocations are rounded down. This results in a vector $\boldsymbol{S}$ with integer elements, with sums possibly less than n. If $s < n$ then $n - s$ design points are randomly chosen from $\boldsymbol{S}$ (with replacement) and added to the design.

(c) Calculate the loss function value for each design, denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

(d) Update local best design $\mathbf{p}_k^{(new)} = \text{argmin}\{\mathbb{L}_\nu(\mathbf{p}_k^{(old)}), \mathbb{L}_\nu(\boldsymbol{\xi}_k^{(new)})\}$ and global best design $\mathbf{p}_g^{(new)} = \text{argmin}_{1\le k\le K}\mathbb{L}_\nu(\mathbf{p}_k^{(new)})$.

3. Step 2 is carried out repeatedly until the best design does not change for $GK = 100,000$ times of running step 2. The output $\mathbf{p}_g$ is as our best design with loss function value $\mathbb{L}_\nu(\mathbf{p}_g)$.

The results are as shown in Figure 4.2. Similarly as previous, we use different value of ν to investigate the optimal design under different weight of bias. We can see from (f) that, the loss deceases, by the times of going through all the designs, at the beginning and remains fixed after certain number of times. From (a) to (e), the results are consistent with what we have got from Genetic Algorithm which were shown in Figure 4.2.

Compared with Genetic Algorithm, the efficiency of Particle Swarm Optimization is largely controlled by the choice of parameters such as c_1, c_2 and w_k and K. The reason why the size of initial designs K is chosen much larger (200 v.s. 40) than the one for Genetic Algorithm is quite obvious. In order to mimic the birds flocking, the initial designs should spread as widely as possible. In our case, the *'birds'* flocking happens in a space of dimension $N = 10$, hence K should be chosen relatively large. The choice of $K = 200$ here was mostly based on the practice of carrying out the Particle Swarm Optimization method.

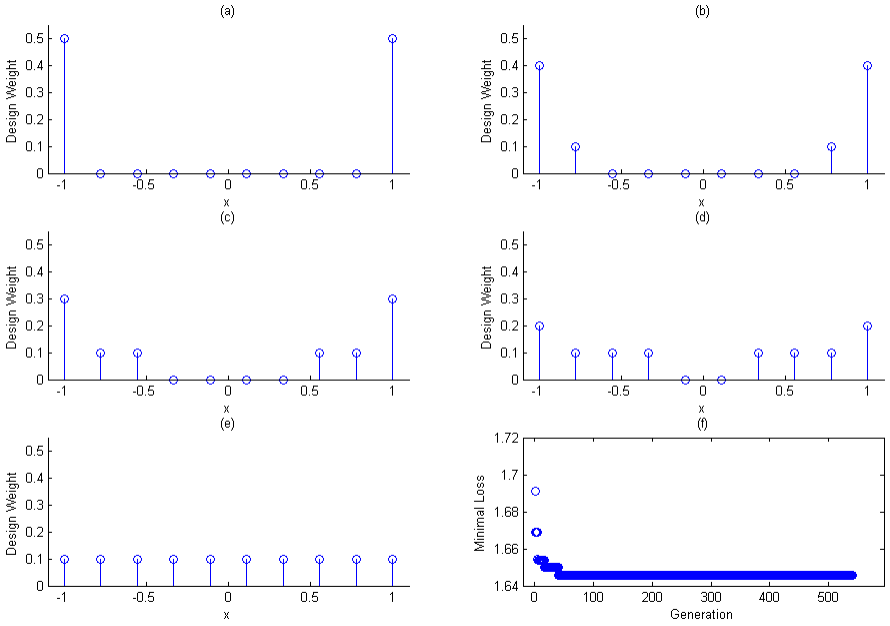


Figure 4.2: The minimax design with $n = 10$, $N = 10$, $K = 200$ generated by Particle Swarm Optimization. (a) $\nu = 0.0$, $loss = 2.67$; (b) $\nu = 0.25$, $loss = 2.21$; (c) $\nu = 0.50$, $loss = 1.65$; (d) $\nu = 0.75$, $loss = 0.99$; (e) $\nu = 1.0$, $loss = 0.20$. (f) shows minimal loss against generation for the case of $\nu = 0.5$.

4.3 Expected Improvement Algorithm

Using the same framework as previous sections, we would like to get a $n = 10$ point design minimizing the expression (3.2). Here we choose $K = 40$ to be the initial designs. As indicated in Section 2.4, there are two steps of carrying out a Expected Improvement Algorithm: the choice of an appropriate model for prediction and generating the new evaluation point $\xi^\star$. Let us discuss the first one.

4.3.1 Model Validation

To employ the *DACE* model expressed as (2.1), we need to firstly verify our responses, i.e. *losses*, to be approximatively normal distributed. Unfortunately, as shown in (a) of Figure 4.3, under the original scale, the losses exhibit no normality at all. Hence we use a Box-Cox transformation, which is one particular way of parameterising a power transform.

Power transform (see Box and Cox (1964)) is a family of transformation functions that are applied to create a rank-preserving transformation of data. This is a useful data (pre)processing technique used to stabilize variance, make the data more normal distribution-like, improve the correlation between variables and for other data stabilization procedures. The Box-Cox transform, named after the statisticians George E. P. Box and David Cox (see Box and Cox (1964)) is one particular way of parameterising a power transform that has advantageous properties.

The transform chosen here is defined as

$$l_i^{(\lambda)} = \begin{cases} (l_i^{\lambda} - 1)/(\lambda \cdot GM(l)^{\lambda-1}) & \text{if } \lambda \neq 0, \\ \\ GM(l) \cdot lnl_i & \text{if } \lambda = 0; \end{cases}$$

where

$$GM(l) = \left(\prod_{i=1}^{M} l_i \right)^{\frac{1}{K}}.$$

And by taking $\lambda = -1.5$, the transform made the original responses $\{l_i\}_{i=1}^{M}$ to be $\{l_i^{(\lambda)}\}_{i=1}^{K}$, and ordering relation does not change. In other words, if $l_i \leq l_j$, the relation holds that $l_i^{(\lambda)} \leq l_j^{(\lambda)}$. And the transformed responses follow an approximately normal distribution as shown in (b) of Figure 4.3.

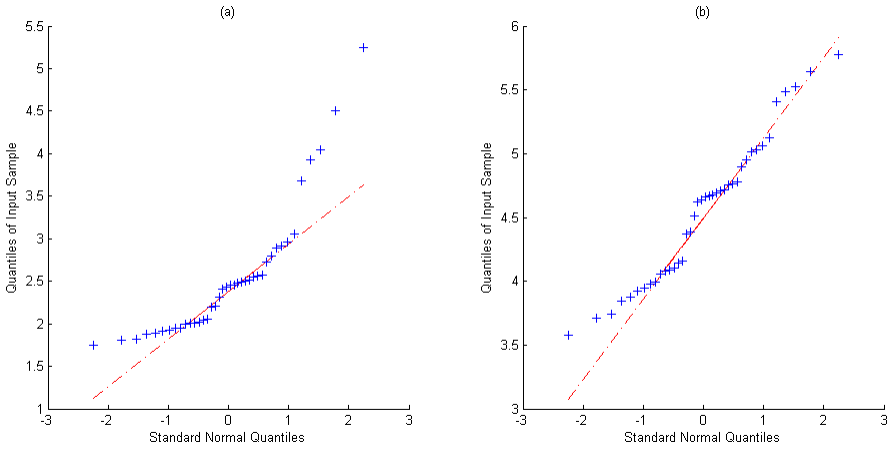


Figure 4.3: The normality plots for the losses. (a) The normality plots for the losses under the original scale; (b) The normality plots for the power transformed losses.

To assess the accuracy of the *DACE* model without sampling any points beyond those used to fit the model, we are going to use the procedure of *'cross validation'*.

The basic idea of *cross validation* is to leave out one observation, say $l_i^{(\lambda)}$, and then predict it back based only on the $n-1$ remaining points. We call this prediction the *'cross validation'* prediction of $l_i^{(\lambda)}$ and denote it by $\hat{l}_{-i}^{(\lambda)}$. The subscript $-i$ emphasizes that observation i is not used in making the prediction. When making these cross-validated predictions, one should re-estimate all the *DACE* parameters using the reduced sample.

In addition to the cross-validated prediction $\hat{l}_{-i}^{(\lambda)}$, we also get a cross-validated standard error of prediction shown as (for the derivation, see Sacks, Welch, Mitchell and Wynn (1989-2))):

$$s_{-i}^2 = \hat{\sigma}^2 \left[1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r} + \frac{(1 - \mathbf{1}'\mathbf{R}^{-1}\mathbf{r})^2}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}} \right].$$

It is convenient to compute the number of standard errors that the actual value is above or below the predicted value:

$$\frac{l_i^{(\lambda)} - \hat{l}_{-i}^{(\lambda)}}{s_{-i}^2}.$$

We will refer to this quantity as the *'standardized cross-validated residual'*. If the model is valid, the value should be roughly in the interval $[-3, +3]$. To illustrate, Figure 4.4 shows the diagnostic tests for the *DACE* model fitting the transformed losses. The surface was based on the initial $K = 40$ designs. In Figure 4.4, (a) plots the actual function value versus the cross-validated prediction. If the model were good, the points should lie on a 45-degree line; in this case, the relationship is very good; (b) plots the standardized cross-validated residuals versus the cross-validated

predictions. All of residuals are distributed evenly around 0 and all of them fall within the interval $[-3, +3]$; Moreover the Q-Q plot in (c) indicates a fairly normal distribution for the standardized residuals. The diagnostic plots suggest that *DACE* model is fairly accurate of making predictions.

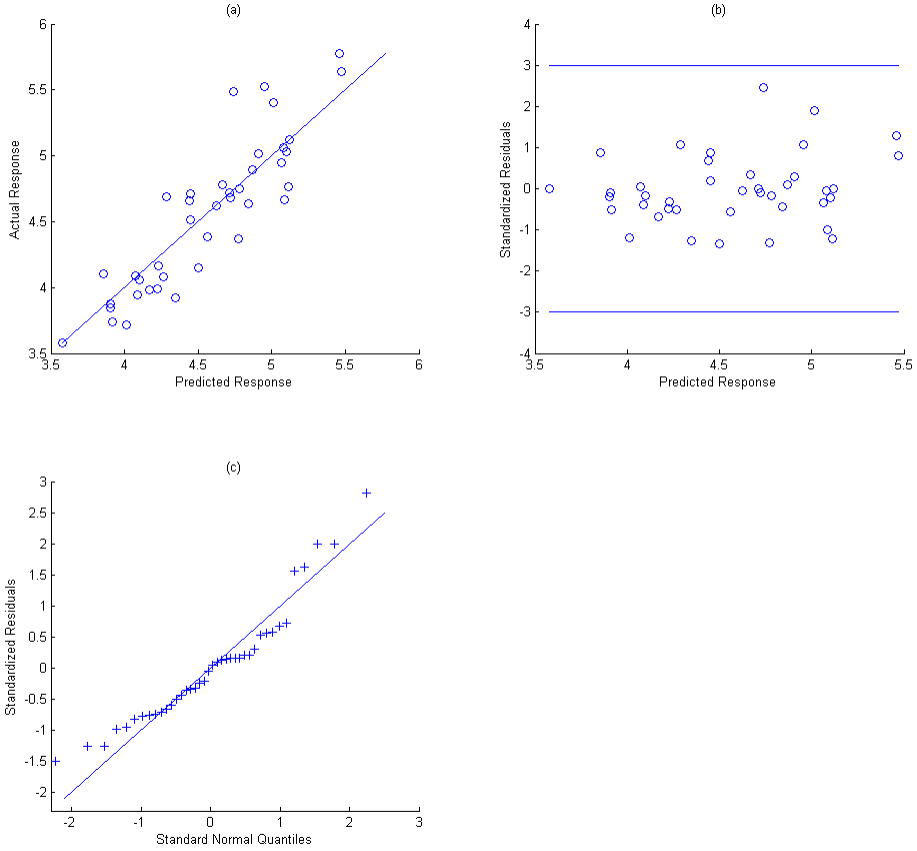


Figure 4.4: Diagnostic tests for *DACE* model fitting transformed losses. (a) The normality plots for the losses under the original scale; (b) The normality plots for the power transformed losses.

4.3.2 Expected Improvement Maximization

Then we can get a new design $\boldsymbol{\xi}^\star$ by maximizing the expected improvement $E[I(\boldsymbol{\xi})]$ given as (2.4) and add it as a new design $\boldsymbol{\xi}_{m+1}$ into the design sequences. Afterwards we can get the global optimal design by comparing the function values $l_1, \ldots, l_m, l_{m+1}$. We can repeat the steps above by adding one more design each time. If the global optimal design does not change for certain number of steps, say the number is $G = 10$, we stop and output the global optimal design. The explicit algorithm restricted to the SLR case is as follows (for a detailed explanation of each term, please see Section 2.4):

1. Start by randomly generating the initial set $\mathcal{D}$ of designs with size $K = 40$, i.e. $\mathcal{D} = \{\boldsymbol{\xi}_k\}_{k=1}^K$. And each design (density) $\boldsymbol{\xi}_k$ can be identified with a multinomial vector of length N, with sum n.

2. Calculate the loss function value for each design of $\mathcal{D}$ denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, where $k = 1, \ldots, K$. Denote $\boldsymbol{\xi}_{min} = \text{argmin}_{1\le k\le K}\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

3. Denote the size of $\mathcal{D}$ as K.

   (a) Calculate the loss function value for each design of $\mathcal{D}$ denoted as $\mathbb{L}_\nu(\boldsymbol{\xi}_k)$, where $k = 1, \ldots, K$.

   (b) Get a new design $\boldsymbol{\xi}^\star$ maximizing $E[I(\boldsymbol{\xi})]$, i.e.

   $$\boldsymbol{\xi}^\star = \text{argmin}_{\boldsymbol{\xi}} E[I(\boldsymbol{\xi})],$$

   and add $\boldsymbol{\xi}^\star$ as a new design $\boldsymbol{\xi}_{K+1}$ into the design sequences $\mathcal{D} = \{\boldsymbol{\xi}_k\}_{k=1}^{K+1}$.

   (c) Update $\boldsymbol{\xi}_{min}$ by using $\boldsymbol{\xi}_{min} = \text{argmin}_{1\le k\le K+1}\mathbb{L}_\nu(\boldsymbol{\xi}_k)$.

4. Step 3 is carried out repeatedly until certain stopping criteria are satisfied. For example the best design $\boldsymbol{\xi}_{min}$ does not change for $G = 10$ times of running step 3. The output $\boldsymbol{\xi}_{min}$ is as our best design with loss function value $\mathbb{L}_{\nu}(\boldsymbol{\xi}_{min})$.

The results for varied ν are shown in Figure 4.5. We can see from (f) that, the minimal loss deceases, by the times of evaluating original loss functions, at the beginning and remains fixed after certain number of times. From (a) to (e), the results are consistent with what we have got using two algorithms previously. Compared with the methods of Genetic Algorithm and Particle Swarm Optimization, the efficiency of Expected Improvement Algorithm is very weak. As indicated at the beginning of this chapter, in order to save the number of times of evaluating optimized function, we need to go through evaluating the expected improvement function (2.4) at each possible point, i.e. for each possible design density. And (2.4) will be evaluated as many as $\binom{n+N-1}{N-1}$ times, which could be very large when N and n are chosen to be large.

However it does save the number of evaluations of the original loss function. Instead of evaluating the loss function by a huge amount of times in order to get an optimal design (2000 times for *GA* and 10000 times for *PSO*), *EIA* algorithm only makes 30 times of loss function evaluating. It is extremely useful when the loss function evaluation is extremely expensive. Actually in the algorithm of *EIA*, there is a trade-off of sacrificing the time of evaluating *expected improvement function* in return for saving the expense of evaluating *loss function*.

Considering the computing time for the loss function of SLR case with with $\nu = 0.5$, the *GA* takes 0.83 seconds, the *PSO* takes 9.75 seconds and *EIA* takes 329.09 sec-

onds. The *GA* is the most efficient one compared with other two methods and *EIA* is the least efficient due to its need of large amount of evaluations of *expected improvement function*.

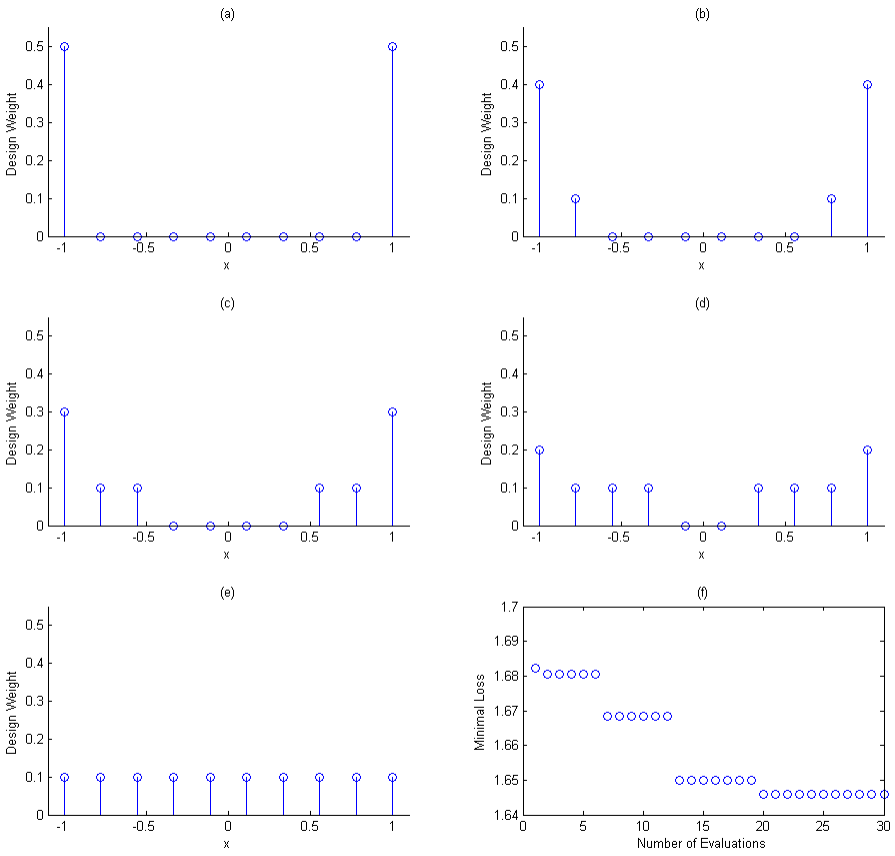


Figure 4.5: The minimax design with $n = 10$, $N = 10$, $K = 40$, $G = 10$ generated by Expected Improvement Algorithm. (a) $\nu = 0.0$, $loss = 2.67$; (b) $\nu = 0.25$, $loss = 2.21$; (c) $\nu = 0.50$, $loss = 1.65$; (d) $\nu = 0.75$, $loss = 0.99$; (e) $\nu = 1.0$, $loss = 0.20$. (f) shows minimal loss against number of evaluations for the case of $\nu = 0.5$.

Chapter 5

Optimal Design for Exponential Model

In this chapter, we will numerically investigate the minimax design problem for a non-linear regression which is given in Karami (2011) and Karami and Wiens (2012), using three algorithms introduced in Chapter 2. The results are shown to be consistent.

Let the model error $d(x|\boldsymbol{\eta})$ evaluated at $\boldsymbol{\eta}$ be defined as $d(x|\boldsymbol{\eta}) = E[Y(x)] - f(x|\boldsymbol{\eta})$. The approximate nature of the model that $E[Y(x)] \approx f(x|\boldsymbol{\theta})$ for some p-dimensional $\boldsymbol{\theta}$ - is characterized by

$$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\eta}} \sum_{i=1}^{N} (E[Y(x_i)] - f(x_i|\boldsymbol{\eta}))^2.$$

Then the probability model is

$$Y(x) = f(x|\boldsymbol{\theta}) + d(x|\boldsymbol{\theta}) + \epsilon,$$

with random error ϵ assumed to be independently and identically distributed with mean 0 and variance σ^2.

Assume that $f(x|\cdot)$ is differentiable with respect to $\boldsymbol{\eta}$, define $\boldsymbol{Z}(\boldsymbol{\eta})$ to be $N \times p$ matrix with i-th row as

$$z'(x_i|\boldsymbol{\theta}) = \frac{\partial f(x_i|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}},$$

and assume that $\boldsymbol{Z}(\boldsymbol{\theta})$ has full column rank. If $\boldsymbol{d}(\boldsymbol{\eta}) = (d(x_1|\boldsymbol{\eta}), \ldots, d(x_N|\boldsymbol{\eta}))'$, then

$$\boldsymbol{Z}'(\boldsymbol{\theta})\boldsymbol{d}(\boldsymbol{\theta}) = \mathbf{0}_{p\times 1}. \tag{5.1}$$

Here we impose a bounded requirement of $\boldsymbol{d}(\boldsymbol{\theta})$ as

$$\|\boldsymbol{d}(\boldsymbol{\theta})\| \leq \frac{\tau}{\sqrt{n}}, \tag{5.2}$$

for a non-negative constant τ.

We are intent to evaluate the asymptotic *Mean Squared Error (MSE)* of the predicted values, to maximize this *MSE* over a class of models defined above, and then to average this maximized loss with respect to a prior on $\boldsymbol{\theta}$. Given the least squared estimate $\hat{\boldsymbol{\theta}}_n$, a measure of loss here we use over $\boldsymbol{\chi}$ is *Average Mean Squared Error(AMSE)*:

$$\text{AMSE} = \frac{1}{N}\sum_{i=1}^{N} E\left[\{f(x_i|\hat{\boldsymbol{\theta}}_n) - E[Y(x_i)]\}^2\right]. \tag{5.3}$$

Given the definition that $\xi_i = n_i/n$, $\boldsymbol{D}_\xi = diag(\xi_1, \ldots, \xi_N)$ and

$$\boldsymbol{R}_\xi(\boldsymbol{\theta}) = \boldsymbol{Z}(\boldsymbol{\theta})(\boldsymbol{Z}'(\boldsymbol{\theta})\boldsymbol{D}_\xi\boldsymbol{Z}(\boldsymbol{\theta}))^{-1}\boldsymbol{Z}'(\boldsymbol{\theta}),$$

Karami (2011) proves that an asymptotic, first order approximation to (5.3), maximized over the neighbourhood given by vector $\boldsymbol{d}$ satisfying (5.1) and (5.2), is $\frac{\tau^2+\sigma^2}{nN}\mathcal{L}_\nu(\boldsymbol{\theta}|\xi)$, where

$$\mathcal{L}_\nu(\boldsymbol{\theta}|\xi) = (1-\nu)tr[\boldsymbol{R}_\xi(\boldsymbol{\theta})] + \nu ch_{max}[\boldsymbol{R}_\xi(\boldsymbol{\theta})\boldsymbol{D}_\xi^2(\boldsymbol{\theta})\boldsymbol{R}_\xi(\boldsymbol{\theta})]. \tag{5.4}$$

Similarly to the case of *SLR*, the weight of bias is $\nu = \frac{\tau^2}{\tau^2+\sigma^2}$. And let $p(\boldsymbol{\theta})$ be a density on the parameter space Θ, integrate (5.4) over Θ, we can have the loss of the form of

$$\mathbb{L}_\nu(\xi) = \int_\Theta \mathcal{L}_\nu(\boldsymbol{\theta}|\xi)p(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{5.5}$$

Let us consider a nonlinear function of the form

$$f(x;\theta) = \exp(-\theta x); \quad 0 < \theta < 1, x \geq 0.$$

After assuming the prior distribution on θ as $U(0,1)$ and equal weights of variance and bias, i.e. $\nu = 0.5$, the loss function from (5.5) is of the form

$$\mathbb{L}(\boldsymbol{\xi}) = \frac{1}{2}\int_0^1 \left[\frac{f(\theta;\boldsymbol{\xi})}{g(\theta;\boldsymbol{\xi})} + \frac{h(\theta;\boldsymbol{\xi})f(\theta;\boldsymbol{\xi})}{(g(\theta;\boldsymbol{\xi}))^2}\right]d\theta, \tag{5.6}$$

where $f(\theta;\boldsymbol{\xi}) = \sum_{i=1}^N x_i^2\exp(-2\theta x_i)$, $g(\theta;\boldsymbol{\xi}) = \sum_{i=1}^N \frac{n_i}{n}x_i^2\exp(-2\theta x_i)$, $h(\theta;\boldsymbol{\xi}) = \sum_{i=1}^N \frac{n_i^2}{n^2}x_i^2\exp(-2\theta x_i)$ (The detailed deduction can be found in Karami (2011)).

And for the convenience of comparison of three algorithm, we will choose some parameters for the general framework in Section 1.1. Here we will choose $N = 10$ and $n = 10$ over $[0, 10]$, then the design space is as $\chi = \{0, 1.11, 2.22, 3.33, 4.44, 5.56, 6.67, 7.78, 8.89, 10.00\}$.

The *optimized function* here is the new loss function, whose form for the case of equal weighted exponential model is given as (5.6): after determining the design density ξ, we can easily calculate out the value of loss. Similarly as chapter 4, for each algorithm, an optimized design density, which minimized the loss function. is expected.

5.1 Genetic Algorithm

As mentioned at the beginning of this chapter, we would like to get a $n = 10$ point design minimizing the expresssion (5.6). As indicated in Section 2.2, we use generations of size $K = 40$, and vary $P_{mutation}$ linearly from 0 to 0.5. We all consider $P_{crossover} = 0.90$, $P_{elite} = 0.05$, $G = 200$. The algorithm restricted to the exponential case is the same as shown in Section 4.1 except for the loss function is as (5.6).

Given the tuning values above, the result is as shown in Figure 5.1. In Figure 5.1(a), the loss deceases by generation at the beginning and remains fixed after certain number of generations. Moreover, the design density have been shown in plot (b). It is observed that larger masses are made at around the site of $x = 1$, $x = 2$ and $x = 3$.

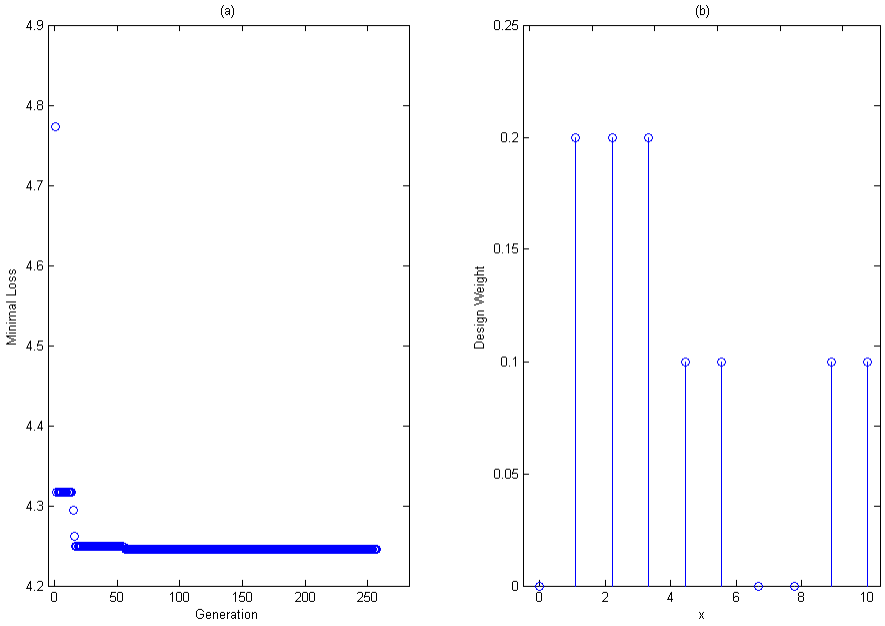


Figure 5.1: The minimax design with $n = 10$, $N = 10$, $K = 40$ generated by Genetic Algorithm for exponential model. (a) shows minimal loss against generation. (b) Best design density with $loss = 4.25$.

5.2 Particle Swarm Optimization

Using the same framework as last section, we would like to get a $n = 10$ point design minimizing the expresssion (5.6). As indicated in Section 2.3, we need to specify some parameters before using the algorithm. Using similar structure as Section 4.2, we would like to choose $K = 200$ as the size of the initial designs (*"particles"*), and w_k decrease linearly from 0.9 to 0.4. We all consider $c_1 = c_2 = 1.49$, $G = 500$. Similarly to the previous section, the explicit algorithm is as the same as Section 4.2 except for the loss function.

After giving the tuning values above, the results are shown in Figure 5.2. Similarly as previous, we can see from (a) that, the loss deceases, by the times of going through all the particles, at the beginning and remains fixed after certain number of times. The result in (b) are consistent with what we have got from Genetic Algorithm shown in Figure 5.1. For the exponential minimax design we got the conclusion as for the simple linear model: Compared with Genetic Algorithm, the efficiency of Particle Swarm Optimization is largely controlled by the choice of parameters such as c_1, c_2 and w_k and K. Here the choice of $K = 200$ here was still mostly based on the practice of carrying out the Particle Swarm Optimization method.

5.3 Expected Improvement Algorithm

Using the same framework as previous sections, we would like to get a $n = 10$ point design minimizing the expression (5.6). Here we choose $K = 40$ to be the initial

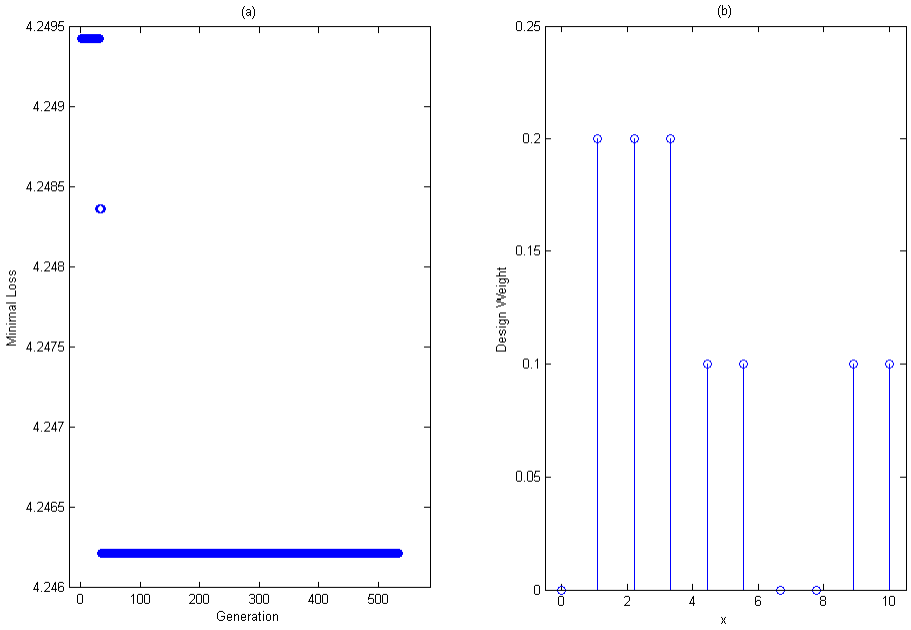


Figure 5.2: The minimax design with $n = 10$, $N = 10$, $K = 200$ generated by Particle Swarm Optimization for exponential model. (a) shows minimal loss against generation. (b) Best design density with $loss = 4.25$.

designs. As section 4.3, here are still two steps of carrying out a Expected Improvement Algorithm: the choice of an appropriate model for prediction and generating the new evaluation point $\xi^{\star}$. Let us see the first one.

5.3.1 Model Validation

By taking $\lambda = -1.5$, the transform made the original responses $\{l_i\}_{i=1}^{K}$ to be $\{l_i^{(\lambda)}\}_{i=1}^{K}$, and ordering relation does not change. In other words, if $l_i \leq l_j$, the relation holds that $l_i^{(\lambda)} \leq l_j^{(\lambda)}$. And the transformed responses follow an approximately normal distribution as shown in (b) of Figure 5.3.

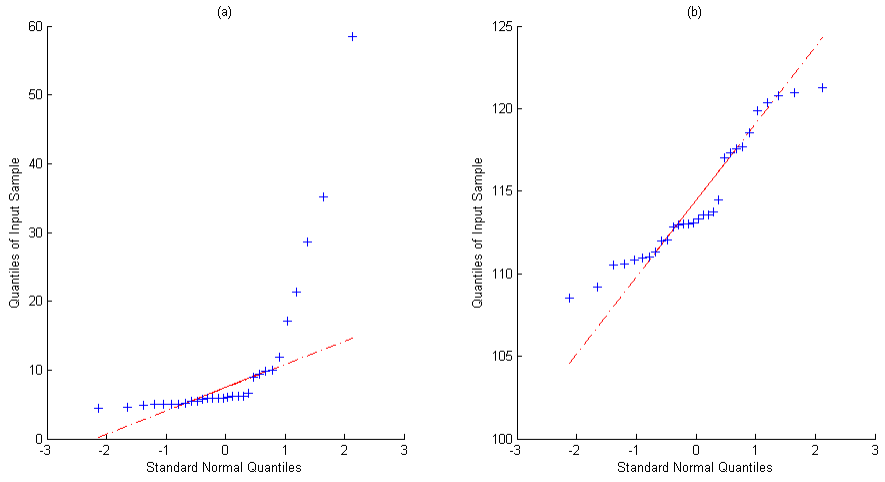


Figure 5.3: The normality plots for the losses. (a) The normality plots for the losses under the original scale; (b) The normality plots for the power transformed losses.

To assess the accuracy of the *DACE* model without sampling any points beyond those used to fit the model, we still use here the procedure of *'cross validation'* (see Section 4.3 for details) .

Figure 5.4 shows the diagnostic tests for the *DACE* model fitting the transformed losses. The surface was based on the initial $K = 40$ designs. In Figure 5.4, (a) plots the actual function value versus the cross-validated prediction. If the model were good, the points should lie on a 45-degree line; in this case, the relationship is very good; (b) plots the standardized cross-validated residuals versus the cross-validated predictions. All of residuals are distributed evenly around 0 and all of them fall within the interval $[-3, +3]$; Moreover the Q-Q plot in (c) indicates a fairly normal distribution for the standardized residuals. The diagnostic plots suggest that *DACE* model is fairly accurate of making predictions.

5.3.2 Expected Improvement Maximization

Then we can get a new design $\boldsymbol{\xi}^{\star}$ by maximizing the expected improvement $E[I(\boldsymbol{\xi})]$ given as (2.4) and add it as a new design $\boldsymbol{\xi}_{m+1}$ into the design sequences. Afterwards we can get the global optimal design by comparing the function values $l_1, \ldots, l_m, l_{m+1}$. We can repeat the steps above by adding one more design each time. If the global optimal design does not change for certain number of steps, say the number is $G = 25$, we stop and output the global optimal design. The explicit algorithm is the same as Section 5.3.2 with replacing only the loss function.

The results are shown in Figure 5.5. We can see from (a) that, the minimal loss deceases, by the times of evaluating original loss functions, at the beginning and remains fixed after certain number of times. The plot (a) is consistent with what we have got using two algorithms previously. Even for the exponential model, com-

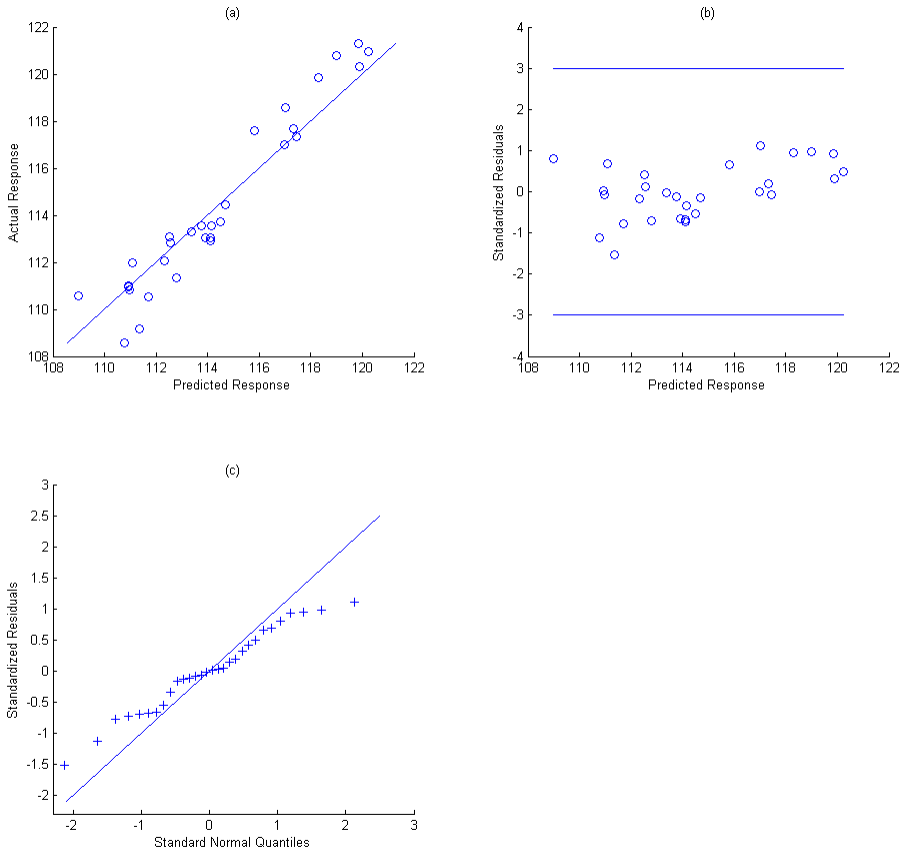


Figure 5.4: Diagnostic tests for *DACE* model fitting transformed losses. (a) The normality plots for the losses under the original scale; (b) The normality plots for the power transformed losses.

pared with the methods of Genetic Algorithm and Particle Swarm Optimization, the efficiency of Expected Improvement Algorithm is still very weak. As indicated at the beginning of Chapter 4, in order to save the number of times of evaluating optimized function, we need to go through evaluating the expected improvement function (2.4) at each possible point, i.e. for each possible design density. And (2.4) will be evaluated as many as $\binom{n+N-1}{N-1}$ times, which could be very large when N and n are chosen to be large.

However it does save the number of evaluating the original loss function. Instead of evaluating the loss function by a huge amount of times in order to get an optimal design (2000 times for *GA* and 10000 times for *PSO*), *EIA* algorithm only makes 27 times of loss function evaluating. It is extremely useful when the loss function evaluation is extremely expensive. Actually in the algorithm of *EIA*, there is a trade-off of sacrificing the time of evaluating *expected improvement function* in return for saving the expense of evaluating *loss function*.

Considering the computing time for the loss function of exponential function case here, the *GA* takes 15.19 seconds, the *PSO* takes 188.25 seconds and *EIA* takes 290.50 seconds. The *GA* is still the most efficient one compared with other two methods and *EIA* is the least efficient due to its need of large amount of evaluations of *expected improvement function*.

However, it is of great interest to point out that comparing with the computing time of the SLR case, the *GA* and *PSO* take substantially longer time, while *EIA* takes even shorter. This is probably due to the longer time of evaluating the loss function. This indicates the advantage for taking the method of *EIA* in the first place. When

the loss function evaluation is very expensive or costs a lot of time, the *EIA* may probably a better method compared to the method of the other two.

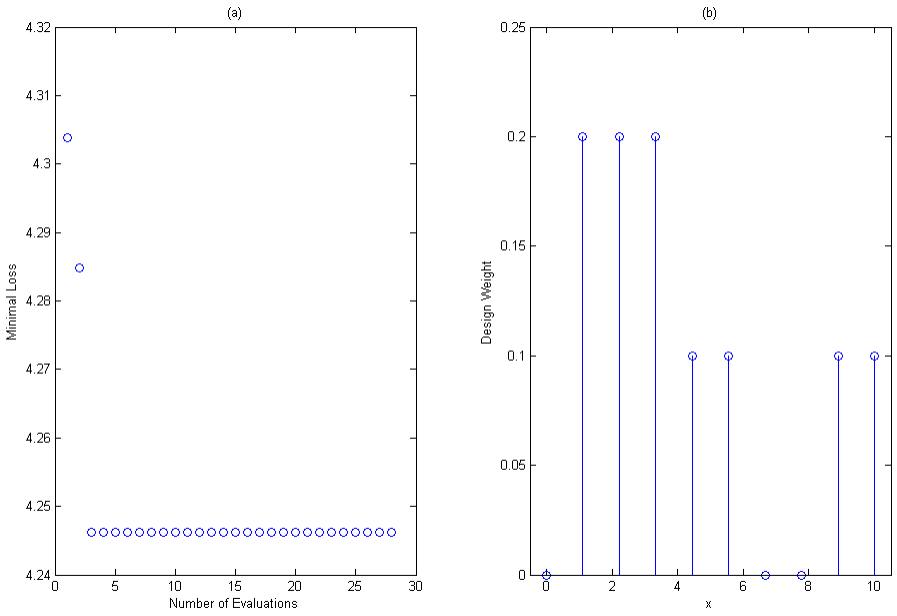


Figure 5.5: The minimax design with $n = 10$, $N = 10$, $K = 40$, $G = 25$ generated by Expected Improvement Algorithm for exponential model. (a) shows minimal loss against number of evaluations; (b) Best design density with $loss = 4.25$.

Chapter 6

Conclusions and Discussions

Ye and Zhou (2007) have given an important result about symmetry of the *Q-optimal minimax design* for the case of the simple linear regression. That is a *Q-optimal minimax design* is symmetric if we require that $\int_{\chi} xm(x)dx = 0$. At the end, they propose that even if the condition $\int_{\chi} xm(x)dx = 0$ is removed, the claim may still be true that *Q-optimal minimax design* is symmetric. In Chapter 3.1, we have shown that for the simple linear regression, considering only the variance, a Q-optimal minimax design is necessarily symmetric; we have also made an attempt of addressing the symmetry problem considering only the bias which is much more difficult to achieve.

However, the numerical results in Chapter 4 using three different algorithms indicate that the claim is true that: for the simple linear regression, even if the condition $\int_{\chi} xm(x)dx = 0$ is removed, a Q-optimal minimax design is still symmetric. Actually, making no requirement of symmetry for the initial design densities, the Q-optimal design densities are always symmetric. To prove it mathematically is still of interest.

And in Chapter 4 and Chapter 5, we apply the three algorithms on a linear and a non-linear cases correspondingly and make the comparison. Generally speaking, considering the time of program running, Genetic Algorithm and Particle Swarm Optimization are very efficient. Compared to Particle Swarm Optimization, Genetic Algorithm requires fewer number of initial designs. Compared to Genetic Algorithm, Particle Swarm Optimization is more sensitive against parameter choosing. However, since Particle Swarm Optimization requires a largely spread initial designs, the final optimal solution would surely be a global optimal one, while under some extreme situation, Genetic Algorithm might only achieve a locally best solution. Compared with the methods of Genetic Algorithm and Particle Swarm Optimization, the efficiency of Expected Improvement Algorithm is very weak. In order to save the number of times of evaluating optimized function, we need to go through evaluating the expected improvement function (2.4) at each possible point, i.e. for each possible design density. It requires a huge amount of calculating. On the other hand. Expected Improvement Algorithm does save the number of evaluating the original loss function. Instead of evaluating the loss function by a huge amount of times in order to get an optimal design, *EIA* algorithm only makes only a few times of loss function evaluating. It is extremely useful when the loss function evaluation is extremely expensive. In the algorithm of *EIA*, there is a trade-off of sacrificing the time of evaluating *expected improvement function* in return for saving the expense of evaluating *loss function*.

Bibliography

Box, G. E. P., and Cox, D. R. (1964), "An Analysis of Transformations", *Journal of The Royal Statistical Society, Series B*, **26 (2)**: 211-252.

Chen, R. B., Chang, S. P., Wang, W. C., and Wong, W. K. (2011), "Optimal Experimental Designs via Particle Swarm Optimization Methods", Preprint, *Department of Mathematics, National Taiwan University*, 2011-03.

Clerc, M. (2006), "Particle Swarm Optimization "(1st ed.), *London, U.K.: Wiley-ISTE*.

Coley, D.A. (1999), "An Introduction to Genetic Algorithms For Scientists and Engineers", *Singapore: World Scientific Publishing Co.*.

Eberhart, R., and Kennedy, J. (1995), "A New Optimizer Using Particle Swarm Theory", *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium*, IEEE, 39-43.

Eberhart, R., Shi, Y., and Kennedy, J. (2001), "Swarm Intelligence ", *Singapore: Elsevier Science Press*.

Heo, G., Schmuland, B., and Wiens, D. P. (2001), "Restricted Minimax Robust Designs for Misspecified Regression Models", *The Canadian Journal of Statistics*, **29**, 117-128.

Huber, P. J., and Ronchetti, E. M. (2009), "Robust Statistics"(2nd ed.), *New York: Wiley*, 1-21, 239-248.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998), "Efficient Global Optimization of Expensive Black-Box Functions", *Journal of Global Optimization*, **13**, 455-492.

Kang, L., Cai, Z., Yan, X., and Liu, Y. (2008), "Advances in Computation and Intelligence: Third International Symposium on Intelligence Computation and Applications", *ISICA 2008 Wuhan, China, December 19-21, 2008 Proceedings*, 101-105.

Karami, J. H. (2011), "Designs for Nonlinear Regression With a Prior on the Parameters", unpublished MSc thesis, University of Alberta, Department of Mathematical and Statistical Sciences.

Karami, J. H. and Wiens, D. P. (2012), "Robust Static Designs for Approximately Specified Nonlinear Regression Models", *Journal of Statistical Planning and Inference*, in press.

Mandal, A., Johnson, K., Wu, J. C. F., and Bornemeier, D. (2007), "Identifying Promising Compounds in Drug Discovery: Genetic Algorithms and Some New Statistical Techniques", *Journal of Chemical Information and Modeling* , **47**, 981-988.

Mitchell, M. (1996), "An Introduction to Genetic Algorithms", *Cambridge, MA MIT Press*.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments", *Statistical Science*, **4**, 409-435.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments "(with discussion), *Statistical Science*, **4**, 409-435.

Wu, C. F. J., and Hamada, M. S. (2009), "Experiments: Planning, Analysis, and Optimization"(2nd Ed.), *New York: Wiley* , 1-2.

Wiens, D. P. (1993), "Designs For Approximately Linear Regression: Maximizing the Minimum Coverage Probability of Confidence Ellipsoids", *Canadian Journal of Statistics*, **21**, 59-70.

Welsh, A. H. and Wiens, D. P. (2013), "Robust Model-based Sampling Designs ", *Statistics and Computing*, **23**, 689-701.

Ye, J. J., and Zhou, J. (2007), "Existence and Symmetry of Minimax Regression Designs ", *Journal of Statistical Planning and Inference*, **137**, 344-354.