PREPRINT

國立臺灣大學 數學系 預印本 Department of Mathematics, National Taiwan University

Optimal Experimental Designs via Particle Swarm Optimization Methods

Ray-Bing Chen, Shin-Perng Chang, Weichung Wang, and Weng Kee Wong

September 7, 2011

Optimal Experimental Designs via Particle Swarm Optimization Methods

Ray-Bing Chen, Shin-Perng Chang, Weichung Wang, and Weng Kee Wong*

September 7, 2011

Abstract

Particle swarm optimization (PSO) method is relatively new, simple yet powerful and widely used in applied fields. However PSO does not seem to have made an impact in mainstream statistical applications hitherto. We propose variants of the PSO method to find optimal experimental designs for both linear and nonlinear models in a novel way. We show that the PSO method can simply generate many types of optimal designs very quickly, including optimal designs under a non-differentiable criterion such as minimax optimal designs where effective algorithms to generate such designs have remained elusive to date.

Keywords. Continuous optimal design, equivalence theorem, Fisher information matrix, minimax optimality criteria, regression model.

1 Introduction

Particle Swarm Optimization (PSO) is a population based stochastic optimization method inspired by social behavior of bird flocking or fish schooling and proposed

*Ray-Bing Chen is with Department of Statistics, National Cheng-Kung University, No. 1 Ta-Hsueh Road, Tainan 70101, Taiwan (e-mail: rbchen@stat.ncku.edu.tw); Shin-Perng Chang is with Department of Network System, Toko University, No. 51, Sec. 2, Syuefu Road, Puzih City, Chiayi 61363, Taiwan (e-mail: gauss.chang@gmail.com); Weichung Wang is with Department of Mathematics, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei 10617, Taiwan (e-mail: wwang@ntu.edu.tw); and Weng Kee Wong is with Department of Biostatistics, School of Public Health, UCLA, 10833 Le Conte Ave., Los Angeles, CA 90095-1772, U.S.A. (e-mail: wkwong@ucla.edu).Weng Kee Wong completed the manuscript when he was visiting The Sir Isaac Newton Institute at Cambridge, England. He would like to thank the Institute for the support during his visit from August 5^{th} to September 7^{th} 2011. This work is also partially supported by the National Science Council of Taiwan (Chen and Wang) and the Taida Institute of Mathematical Sciences (Wang).

by Eberhart and Kennedy (1995). Since then the method has generated considerable and increasing interest in optimization circles as evident by its numerous applications to many disciplines. A sampling of the wide applications of PSO can be gleaned from talk titles at a recent scientific meeting entitled International Conference on Swarm Intelligence: Theoretical Advances and Real world Applications in June 2011 at Cergy, France. They included applications to tackle artificial neural network training, K-means cluster analysis, mathematical finance, social networks, data mining, foraging techniques, intrusion detection, resources allocation problems, course+exam scheduling in real time, designing ideotypes for sustainable product systems in genetics and prediction of stock market indices using hybrid genetic algorithm and PSO with a perturbed term. The importance and popularity of PSO can also be seen in the existence of many websites that provide PSO tutorials, tracking its development and various applications in different fields. Some examples are http://www.swarmintelligence.org/index.php, http://www.cis.syr.edu/∼mohan/pso/ and http://www.particleswarm.info/. The former website shows there is at least workshop or symposium on PSO methodology every year since 2001 and many of these are IEEE sponsored. The speed of development in PSO and more generally in metaheuristic or natured inspired algorithms can also be seen in an exploding volume of papers and book chapters in optimization monographs. In 2007, even a journal called Swarm Intelligence was born and published by Springer to meet the rapid rise of interest in PSO methods. Of particular note is also Yang (2010), who saw a need for a second edition to update PSO developments for his book published 2 years earlier. Clerc (2006) is the first book devoted entirely to PSO, others include Lazinica (2009) and Olsson (2011). There are earlier books that focus on broader theme on swarm intelligence, for example, Eberhart et al. (2001) before researchers focus on methods specifically based on PSO.

Our main aim is to show that the PSO method is effective in finding a wide variety of optimal experimental designs. We focus on the more difficult case where the optimality criterion is non-differentiable for two reasons: (i) we do not know of effective algorithms for finding optimal designs under a non-differentiable criterion in the statistical literature to date and (ii) if the method works for such cases, it should also work for the easier case when the criterion is differentiable. The latter class includes the widely used D-, A-, c- and D_s-optimality criteria. To fix ideas, we choose minimax optimality criteria, each of which is non-differentiable and demonstrate that the PSO algorithm can readily generate different types of minimax optimal designs that agree with the few published results in the literature.

PSO is a stochastically iterative procedure to optimize a function. It has already proven to be a simple and effective way of solving a large class of optimization problems in many disciplines. It is an amazingly simple and powerful optimization tool, yet seems hitherto not discussed much at all in the statistical literature. The key advantages of this approach are that the PSO is fast and flexible, there are few tuning parameters required of the algorithm and the PSO method can be generically written down easily and straightforwardly applied to find optimal designs for each regression

model. For our problems, only the optimality criterion and the information matrix have to be changed in the exemplary pseudo MATLAB codes that we provide in Section 4 to generate the optimal designs.

In the next section, we provide the background. In Section 3, we demonstrate that the PSO method can efficiently generate four types of minimax optimal designs for linear and nonlinear models. The PSO method usually takes a few seconds to find the locally optimal designs, and sometimes even so for minimax optimal designs as well. In Section 4, we provide details and explain how the PSO method works and in Section 5, we present a case study on the effect of tuning parameters on the performance of the PSO method. Section 6 closes with a discussion.

2 Background

We focus on continuous designs that treat all designs as probability measures on the given design space X. This approach was proposed by Kiefer and his collection of voluminous work in this area is now documented in a single collection (Kiefer et al., 1985). Initially, his work was controversial but it is now recognized as a standard and powerful way to find optimal designs for any regression model when we have a large sample size. The only non-trivial assumption required in Kiefer's approach is that the optimality criterion is a convex function of the designs defined on X.

Suppose a continuous design takes p_i proportion of the total observations at $x_i, i = 1, 2, \ldots, k$ and each x_i is from X. We denote such a generic design by

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_k \\ p_1 & p_2 & \ldots & p_k \end{pmatrix},$$

with $p_1 + p_2 + \cdots + p_k = 1$. Given a fixed sample size N, we implement ξ by taking Np_i observations at $x_i, i = 1, 2, .., k$ subject to $Np_1 + Np_2 + \cdots + Np_k = N$. As Kiefer had shown, one can round each of the Np_i's to the nearest integer so that they sum to N without losing too much efficiency if the sample size is sufficiently large.

In contrast, exact optimal designs require that each of the Np_i's has to be an integer and they sum to N. Exact optimal designs are very difficult to determine and study analytically; in the few cases where closed form descriptions for the exact optimal designs are available, they require number-theoretical considerations to derive them. The exact D-optimal design problem for the homoscedastic quadratic model was only solved in the early eighties (Gaffke and Krafft, 1982).

The main advantages of working with continuous designs are (i) the same methodology can be essentially used to find continuous optimal designs for all design criteria and regression models, whereas exact optimal designs require very specific mathematical technique for each model and design criteria, (ii) unlike continuous designs, exact optimal designs depend sensitively on the value of N and so requires an endless list of optimal designs for each value of N that varies for each model and each design criterion, (iii) there are no known general technique to check whether an exact design

is optimal, whereas one can resort to an equivalence theorem to verify if a design is optimal among all continuous designs defined on the given design space X; if the design is not optimal, lower bound for its efficiency can be calculated without knowing the optimal design, and (iv) there are no effective algorithms for finding exact optimal designs but there are algorithms for finding several types of continuous optimal designs when the design criterion is differentiable.

Our setup assumes a regression problem defined on compact design region X, a univariate response and the mean response is modeled by a known function $g(x, \theta)$ apart from the values of the unknown parameters θ. We assume errors have a known distribution and observations are independent. The mean function $g(x, \theta)$ can be a linear or nonlinear function of θ and the set of independent variables x. Following convention, the worth of a (continuous) design ξ is measured by its Fisher information matrix defined to be the negative of the matrix of second derivatives of the log-likelihood function. For example, consider the widely used Michaelis-Menten model in the biological sciences given by

$$y = \frac{ax}{b+x} + \varepsilon, \quad x > 0$$

where $a > 0$ denotes the maximal response possible and $b > 0$ is the value of x for which there is a half-maximal response. In practice, the design space is truncated to $X = [0, c]$ where c is a sufficiently large user-selected constant. If $\theta^\top = (a, b)$ and the error ε is normally distributed with mean 0 and constant variance, the Fisher information matrix for a given design ξ is

$$I(\theta, \xi) = \int (\frac{ax}{b+x})^2 \begin{pmatrix} \frac{1}{a^2} & -\frac{1}{a(b+x)} \\ -\frac{1}{a(b+x)} & \frac{1}{(b+x)^2} \end{pmatrix} d\xi(x).$$

For nonlinear models, such as the Michaelis-Menten model, the information matrix depends on the model parameters. For linear models, the information matrix does not depend on the model parameters and we will denote it by $I(\xi)$. The information may be singular and when we allow such possibility, we use $I(\xi)^-$ or $I(\xi, \theta)^-$ to denote its generalized inverse.

Following convention, the optimality criterion is formulated as a convex function of the information matrix and the optimal design is found by minimizing the criterion over all designs on the design space X. A common criterion is D-optimality where we want to find a design to minimize $\log |I(\theta, \xi)^{-1}|$ over all designs ξ on X. Because this criterion contains θ, a nominal value or best guess is needed for θ before the function is minimized. The resulting D-optimal design depends on the nominal value and so it is called locally D-optimal. Further, the criterion is a convex function in ξ and this means that a standard directional derivative argument can be applied to produce an equivalence theorem which checks whether any design is D-optimal among all designs on X. Details are available in design monographs, such as Fedorov (1972) or Silvey (1980), for example.

Minimax optimal design arises naturally when we wish to have some protection against the worst case scenario. For example in the Michaelis-Menten model, suppose we now have a plausible range of values for b, say $b \in \Theta$ and Θ is given. If one assumes the most pessimistic case when the true but unknown value of b is the one that minimizes $|I(\theta, \xi)|$, one may wish to have a minimax optimal design ξ^* defined by

$$\xi^* = \arg \min_{\xi} \max_{b \in \Theta} log|I^{-1}(\theta, \xi)|.$$

The optimal design provides some global protection against the worst case scenario by minimizing the maximal inefficiencies of the parameter estimates. Clearly, when Θ is a singleton set, the minimax optimal design becomes a locally optimal design.

The minimax optimality criterion also arises naturally when one considers dose response studies in a heteroscedastic linear model with mean function $f(x)$ and we want to model responses outside the safety limits of the dose range X. If $\lambda(x)$ is the assumed reciprocal variance of the response at dose x and the extrapolated dose of interest is z, we naturally want a design to minimize

$$v(z, \xi) = f^T(z) I(\xi)^{-1} f(z)$$

among all designs on X. However if we only know the dose levels of interest for extrapolation are in the (compact) set Z , one may seek a design to minimize the maximal variance of the predicted responses on Z. Such a criterion is also convex and it can be shown that the following is an equivalence theorem to check whether a design is minimax optimal for extrapolation on Z: ξ^* is minimax-optimal if and only if there exists a probability measure μ^* on $A(\xi^*)$ such that for all x in X,

$$c(x, \mu^*, \xi^*) = \int_{A_(\xi^*)} \lambda(x) r(x, u, \xi^*) \mu^*(du) - v(u, \xi^*) \leq 0$$

with equality at the support points of ξ^*. Here, $A(\xi) = \{u \in Z | v(u, \xi) = \max_{z \in Z} v(z, \xi)\}$ and $r(x, u, \xi) = (f^T(x) M(\xi)^{-1} f(u))^2$. If X is one or two-dimensional, one may visually inspect the plot of $c(x, \mu^*, \xi^*)$ versus values of $x \in X$ to confirm the optimality of ξ^*. Following convention, we display the graph of $c(x, \mu^*, \xi^*)$ to verify the optimality of the design ξ^* without reporting the measure μ^*. A formal proof of this equivalence theorem can be found in Berger et al. (2000). Further details on minimax optimal designs are available in Wong (1992) and, Wong and Cook (1993) with further examples in King and Wong (1998, 2000).

Two points are worth noting: (i) when Z is a singleton set that does not belong to X, we are seeking an optimal design to minimize the variance of the response at an extrapolated point. In this case, the probability measure μ^* is necessarily degenerate at Z and the resulting equivalence theorem reduces to one for checking whether a design is c-optimal, see Fedorov (1972) or Silvey (1980); (ii) equivalence theorems for minimax optimality criteria all have a form similar to the one shown above and they are more complicated because we need to work with the subgradient μ^*. Finding

the subgradient requires another set of optimization procedure which usually is more tricky to handle and this in part explains why minimax optimal designs are much harder to find than optimal designs under a differentiable criterion.

3 Examples of Optimal Designs found via the PSO method

We postpone description of the PSO method to Section 4 and present here our examples using PSO to find different types of continuous minimax optimal designs. These optimal designs are notoriously difficult to find and we know of no algorithm to date that is guaranteed to find such optimal designs. We are therefore naturally interested to test whether PSO provides an effective way of determining minimax optimal designs. Our examples in this section are confined to the scattered few minimax optimal designs reported in the literature, either numerically or analytically. The hope is that all optimal designs found by the PSO method agree with those published in the literature and this would then suggest that the algorithm should also work well for problems whose minimax optimal designs are unknown. Of course, we can also confirm the optimality of the design found by the PSO method using an equivalence theorem.

We use four examples to demonstrate that the PSO method is able to find different types of minimax optimal designs. Two models have binary responses and two have continuous responses. The first example seeks to find a design to minimize the maximum eigenvalue of the inverse of the Fisher information matrix and the second example finds a design to minimize the maximum variance of the estimated model parameters. These two examples have closed form solutions that were elegantly derived and as expected, they are complicated. The optimal designs are called E-optimal in Example 1 and minimax single parameters in Example 2. Example 3 seeks a best design for estimating parameters in a two-parameter logistic model when we have apriori a range of plausible values for each of the two parameters. The desired design sought is the one that maximizes the smallest determinant of the information matrix over all possible nominal values of the two parameters in the plausible region. Equivalently, this is the minimax optimal design that minimizes the maximum determinant of the inverse of the information matrix where the maximum is over the set of all nominal values in the plausible region for the pair of parameters. The numerically minimax optimal design for Example 3 was found in King and Wong (2000) with the aid of Mathematica and we will compare it with our design found from the PSO method. The last example concerns a heteroscedastic model with a continuous outcome and a known efficiency function and we want a design to minimize the maximum predicted variances across the design space or a given extrapolated region. The minimax optimal designs are unknown for these two cases and we will check the optimality of the generated design from the PSO method using an equivalence theorem.

The key tuning parameters in the PSO method are (i) flock size, i.e. number of

particles (designs) to use in the search, (ii) the number of common support points the particles have, (iii) the number of iterations allowed in the search for the optimal design ξ^* and (iv) the number of iterations used to search for the maximal value in the minimax problem. For brevity, we call the third and fourth parameters the outer and inner iteration number, respectively. Unless mentioned otherwise, we will use the same value for both numbers and call them simply the iteration number. We examine briefly the impact of having different values of these two iteration numbers in Section 5. Throughout, we programmed using MATLAB version: R2010b 64bit and all CPU computing time was based on a Intel Core2 6300 computer with 5 GB RAM and operating system was Ubuntu 64bit Linux with kernel 2.6.35-30.

We now present the four examples with a bit more details for the first example.

3.1 Example 1: E-optimal designs for the Michaelis-Menten model.

The Michaelis-Menten model is one of the simplest and most widely used model in the biological sciences. Dette and Wong (1999) used a geometric argument based on the celebrated Elfving's theorem and constructed locally E-optimal designs for model with two parameters $\theta^\top = (a, b)$. Such optimal designs are useful for making inference on θ by making the area of the confidence ellipsoid small in terms of minimizing the length of the principal axis. This is achieved by minimizing the larger of the two eigenvalues of the inverse of the information matrix over all designs on X. For a given θ, they showed that on a given design space $X = [0, \tilde{x}]$, the E-optimal design is supported on $\tilde{x}$ and $\{(\sqrt{2}-1)b\tilde{x}\}/\{2-\sqrt{2})\tilde{x}+b\}$, and the weight at the latter support point is

$$w = \frac{\sqrt{2}(a/b)^2(1-\tilde{z})\{\sqrt{2}-(4-2\sqrt{2})\tilde{z}\}}{2+(a/b)^2\{\sqrt{2}-(4-2\sqrt{2})\tilde{z}\}^2},$$

where $\tilde{z} = \tilde{x}/(b+\tilde{x})$.

We now use the PSO procedure to numerically search for the E-optimal design with two support points. For this example, we choose 128 particles and iterate 100 times. The minimax optimal designs are shown in Table 1 for various combinations of the nominal values of θ. All our numerically generated designs are close to the E-optimal designs reported in Dette and Wong (1999).

It is instructive to demonstrate the search process of the PSO method in a bit more detail for this example; similar demonstrations can be shown for the other examples as well. As an illustrative example, consider the case when $a = 100$ and $b = 150$ and we decide to use 128 particles and 100 iterations. Figure 1 is the plot of the "best" maximum eigenvalue of $I(\xi, \theta)$ obtained in each of the first 10 iterations of PSO procedure. Notice how quickly the PSO method finds the smallest of the larger eigenvalue after just 2 iterations. To show the movements of the particles, we plot in Figure 2 the support points of the first 128 particle designs and show how they change after the 1st, 5th and the 10th iterations when they converged.

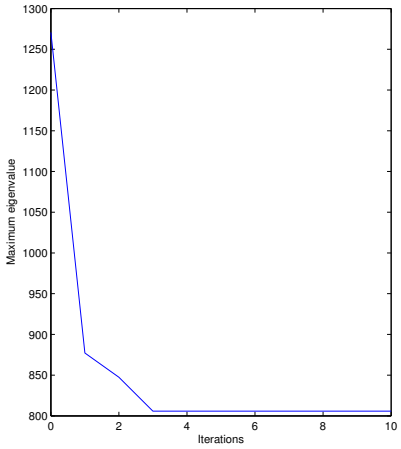


Figure 1: Plot of the maximum eigenvalue of $I(\xi, \theta)^{-1}$ versus the number of PSO iterations in Example 1.

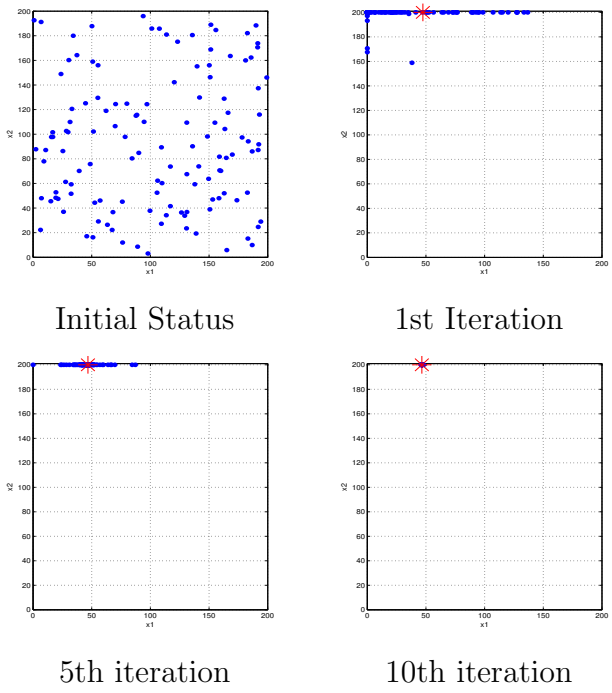


Figure 2: The movement of particles in the PSO search for the E-optimal design for the Michaelis-Menten model at various stages. The red star in each of the three plots indicates the current best design.

Table 1: Locally E-optimal designs for the Michaelis-Menten model on $[0, \tilde{x}] = [0, 200]$ in Example 1.

a	b	ξ_{PSO}		E-optimal designs	
100	150	46.5197(0.6925)	200(0.3075)	45.51(0.6927)	200(0.3073)
100	100	38.1523(0.6770)	200(0.3230)	38.15(0.6769)	200(0.3231)
100	50	24.7828(0.6171)	200(0.3829)	24.78(0.6171)	200(0.3829)
100	10	6.5157(0.2600)	200(0.7400)	6.515(0.2600)	200(0.7400)
100	1	0.7009(0.0222)	200(0.9778)	0.701(0.0220)	200(0.9778)
10	150	46.4971(0.7071)	200(0.2929)	46.51(0.7070)	200(0.2931)
10	100	38.1422(0.7068)	200(0.2932)	38.15(0.7068)	200(0.2933)
10	50	24.7783(0.7058)	200(0.2942)	24.78(0.7058)	200(0.2942)
10	10	6.5154(0.6837)	200(0.3163)	6.515(0.6838)	200(0.3162)
10	1	0.7012(0.1882)	200(0.8118)	0.701(0.1881)	200(0.8119)

To get a sense of computing time that PSO required to run through a procedure for the Michaelis-Menten model using MATLAB, we briefly consider the case when the model parameters are $(a, b)^T = (100, 150)$ using different numbers of particles and iteration. When the iteration is fixed at 100, and the number of particles is 128, 256, 512, 1024 and 2048, the CPU time required is 0.87, 1.65, 3.16, 6.32, 12.58 respectively. When the number of particles is fixed at 128, and the iteration number is 200, 500 and 1000, the time required is 1.68, 4.13 and 8.05 respectively of CPU time. In all cases, the generated designs agree up to 5 decimal places in terms of both weights and design points. Clearly larger flock size requires more time to partake in the sharing of information and larger number of iterations requires longer time. Section 5 explores relationship among the tuning parameters in a bit more detail for a specific case.

3.2 Example 2: Locally minimax single parameters optimal designs for the Double Exponential model.

Dette and Sahm (1998) found theoretical minimax optimal design for a class of binary response models. Let y be the binary response with probability of success $p(x, \theta)$, and suppose $p(x, \theta) = F(\beta(x - \mu))$ where $x \in \mathcal{R}$, $\theta^T = (\beta, \mu)$ and F is some appropriate function. We focus on the double exponential model given by

$$F(x) = \frac{1 + \text{sign}(x)}{2} - \frac{\text{sign}(x)}{2} \exp(-|x|).$$

Given a design ξ, the Fisher information matrix for θ is

$$I(\theta, \xi) = \int h(\beta(x - \mu)) \begin{pmatrix} \beta^2 & -\beta(x - \mu) \\ -\beta(x - \mu) & (x - \mu)^2 \end{pmatrix} d\xi(x),$$

where $h(x) = \frac{f^2(x)}{F(x)(1-F(x))} = (2\exp(-|x|) - 1)^{-1}$. The locally minimax design sought for a pre-specified θ is the one that minimizes $max_{i \in \{1,2\}} e_i^\top I^-(\theta, \xi) e_i$. Here e_i is the

Table 2: Minimax single parameter optimal designs for the double exponential model found from the PSO method and the from the formula.

Cases	ξ_{PSO}	ξ^* in Dette and Sahm (1998)
$(\mu,\beta)=(1,1)^T$	$\begin{pmatrix} -0.5856 & 1.0000 & 2.5856 \\ 0.4263 & 0.1474 & 0.4263 \end{pmatrix}$	$\begin{pmatrix} -0.5936 & 1.0000 & 2.5936 \\ 0.4259 & 0.1482 & 0.4259 \end{pmatrix}$
$(\mu,\beta)=(1,1.3)^T$	$\begin{pmatrix} -0.3000 & 2.3000 \\ 0.5000 & 0.5000 \end{pmatrix}$	$\begin{pmatrix} -0.3000 & 2.3000 \\ 0.5000 & 0.5000 \end{pmatrix}$
$(\mu,\beta)=(1,1.5)^T$	$\begin{pmatrix} -0.2276 & 2.2276 \\ 0.5000 & 0.5000 \end{pmatrix}$	$\begin{pmatrix} -0.2278 & 2.228 \\ 0.5000 & 0.5000 \end{pmatrix}$

i^{th} unit vector in $\mathcal{R}^2$ and $I^-(\theta,\xi)$ is a generalized inverse matrix of $I(\theta,\xi)$. These optimal designs minimize the larger of the two variances from the two estimated model parameters. Such minimax optimal designs appear to be first proposed by Murty (1971) when he considered polynomial models and he called them minimax single parameters optimal designs. Recognizing that the magnitude of the variances of the estimates can vary from parameter to parameter, extensions to construct standardized minimax single parameters optimal designs have also been proposed, see Dette et al. (2003) for example. For our purpose here, it suffices to concentrate on the unstandardized version for simplicity.

Results from Dette and Sahm (1998) showed that there are three important constants to work with in the construction of the minimax optimal design for the double exponential model: $c = 1.84141$, $k = 0.5404$ and $v_0 = 1.59362$. Letting $w = (v_0^2 - \beta^4)h(v_0)/\{h(v_0)(v_0^2 - \beta^4) + \beta^4\}$, the minimax single parameter optimal design ξ^* has three forms depending on their nominal values:

Case 1 $(\beta^2 < v_0)$**:** $\xi^* = \begin{pmatrix} \mu - \frac{v_0}{\beta} & \mu & \mu + \frac{v_0}{\beta} \\ \frac{1-w}{2} & w & \frac{1-w}{2} \end{pmatrix}$.

Case 2 $(v_0 \le \beta^2 \le c)$**:** $\xi^* = \begin{pmatrix} \mu - \beta & \mu + \beta \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

Case 3 $(\beta^2 > c)$**:** $\xi^* = \begin{pmatrix} \mu - \frac{c}{\beta} & \mu + \frac{c}{\beta} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

Following Dette and Sahm (1998), we assume the design points are symmetric around μ. Table 2 displays a numerical example for each of the three different cases calculated from the formula, along with those found from the PSO method using 128 particles and 100 iterations. The two sets of results agree except possibly in case(i) where some disagreement occurs between the theoretical minimax single parameters design reported in Dette and Sahm (1998) and the one found by the PSO method. We revisit this issue just before Section 4.

3.3 Example 3: A minimax D-optimal design for the two-parameter logistic regression model when we have plausible ranges for the two parameters.

The widely used two-parameter logistic model assumes the probability of response is $p(x,\theta) = 1/\{1+\exp(-b(x-a))\}$ with $\theta^T = (a,b)$. Assuming nominal values for the parameters are available, the Fisher information matrix for a given design ξ is

$$I(\xi,\theta) = \int \begin{pmatrix} b^2 p(x,\theta)(1-p(x,\theta)) & -b(x-a)p(x,\theta)(1-p(x,\theta)) \\ -b(x-a)p(x,\theta)(1-p(x,\theta)) & (x-a)^2 p(x,\theta)(1-p(x,\theta)) \end{pmatrix} d\xi(x).$$

Our interest is when we apriori know the possible range of values for a and b, i.e. $\theta \in \Theta$ and Θ is a known set containing all plausible values of a and b. We wish to find a minimax D-optimal design ξ^* such that

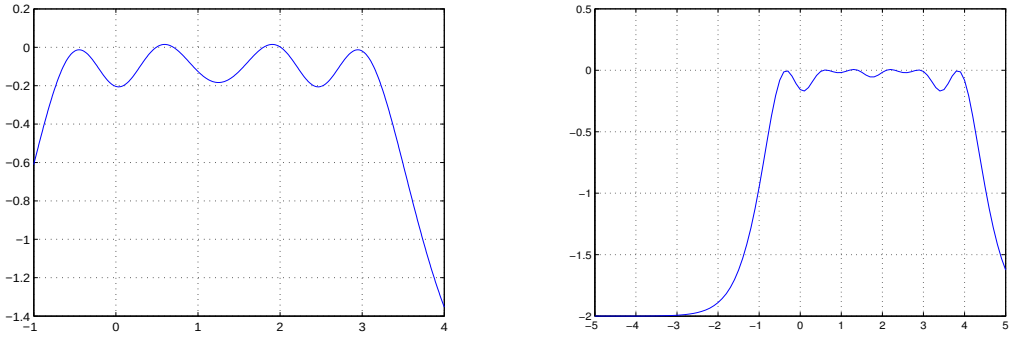
$$\xi^* = \arg\min_{\xi} \max_{\theta\in\Theta} \log(|I^{-1}(\xi,\theta)|).$$

Clearly this minimax optimal design becomes a locally D-optimal design when each of the plausible intervals degenerate to a single point.

We follow King and Wong (2000) and assume that $\Theta = [a_L, a_U] \times [b_L, b_U]$, where a_L, a_U, b_L and b_U are the known limits of the lower and upper bounds for a and b. In King and Wong (2000), the numerically minimax D-optimal designs were found by first running the Fedorov-Wynn algorithm. In each case, the algorithm took a very long time to converge and frequently it did not. However, the algorithm gave us an idea on the form of the optimal design that enabled us to use the equivalence theorem to help us find the numerically optimal design using Mathematica. A certain amount of guesswork was still necessary because we did not have a good understanding of the measure μ^*. In summary, the process was labor intensive and time consuming to find the minimax optimal design. We now apply PSO to find two exemplary cases from King and Wong (2000) and ascertain we obtain the same optimal designs. To do this, we employ the nested PSO and use two sets of PSO parameters to find the optimal designs for the two cases. For case (a), the number of particles for the inner loop is 64 and the number for the outer loop is 32. The number of outer iterations is 100 and the number of inner iterations is 50. In case (2), the number of inner particles is 256 and the number for the outer particles is 512. The outer iteration number is 200 and the inner iteration is 100. In both cases, the time it took the PSO to find the optimal designs was a mere fraction of time it took Mathematica to find the optimal design numerically. We work with different design intervals, symmetric and non-symmetric that contain both positive and negative values.

Case 1: Our first application of the PSO method uses $\Theta = [0, 2.5] \times [1, 3]$. Here we choose design space as $[-1, 4]$. The generated 4-point design ξ from the PSO procedure is

$$\begin{pmatrix} -0.4230 & 0.6164 & 1.8836 & 2.9230 \\ 0.2481 & 0.2519 & 0.2519 & 0.2481 \end{pmatrix}.$$



(a) (b)

Figure 3: Plot of $c(x, \xi, \mu^*)$ versus x for Example 3 for case (1): with $\Theta = [0, 2.5] \times [1, 3]$ and case (2): with $\Theta = [0, 3.5] \times [1, 3.5]$.

This design is close to the one reported in King and Wong (2000). Figure 3(a) is the plot of $c(x, \xi, \mu^*)$ versus $x \in X$ and shows the design ξ found by the PSO method is nearly optimal or optimal.

Case 2: Following Example 3.2 in King and Wong (2000), we set $\Theta = [0, 3.5] \times [1, 3.5]$ and $X = [-5, 5]$. The generated 6-point design ξ from the PSO procedure is

$$\begin{pmatrix} -0.3504 & 0.6075 & 1.4146 & 2.0854 & 2.8925 & 3.8504 \\ 0.1799 & 0.2151 & 0.1050 & 0.1050 & 0.2151 & 0.1799 \end{pmatrix}.$$

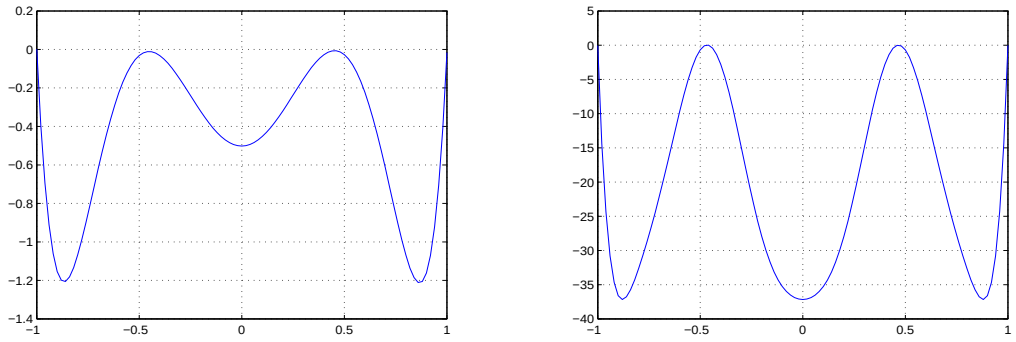
This design is close to the one reported in King and Wong (2000). Figure 3(b) is the plot of $c(x, \xi, \mu^*)$ versus $x \in X$ and shows the design found by the PSO method is nearly optimal or optimal.

3.4 Example 4: A heteroscedastic minimax design for a polynomial model on the prototype design interval $X = [-1, 1]$ with various efficiency functions.

Consider heteroscedastic polynomial models on a given compact design space X and have the form

$$y(x) = h^\top(x)\beta + e(x)/\sqrt{\lambda(x)},$$

where $h^\top(x) = (1, x, \ldots, x^d)$ is a vector of linearly independent continuous functions, $\beta^\top = (\beta_0, \beta_1, \ldots, \beta_d)$ is the vector of unknown parameters and $e(x)$ is a random error having mean 0 and constant variance σ^2. The efficiency function $\lambda(x)$ is a known, bounded, positive real-valued continuous function on X and is inversely proportional to the variance of the response at x.



(a) (b)

Figure 4: Plot of $c(x, \xi, \mu^*)$ versus x over the design interval $X = [-1, 1]$ for the cubic regression model in Example 4 for case (1): $\lambda(x) = 0.5x^2 + 1$ with $Z = [-1, 1]$ and case (2): $\lambda(x) = x^4 + 1 + \sin^2(4x)$ with $Z = [1, 1.5]$.

Given a design ξ and we have uncorrelated observations, the $(d + 1) \times (d + 1)$ information matrix is proportional to

$$I(\xi) = \int_X \lambda(x) h(x) h^\top(x) \xi(dx),$$

and the variance function of ξ at the point x is proportional to

$$d(x, \xi) = h^\top(x) I^{-1}(\xi) h(x).$$

For this purpose, a design, ξ^*, is called minimax optimal design if

$$\xi^* = \arg \min_{\xi} \max_{x \in Z} d(x, \xi),$$

where Z is a compact set and pre-selected for prediction purposes. When $Z = X$, this minimax design is also called the G-optimal design (Wong and Cook, 1993). King and Wong (1998), Brown and Wong (2000) and Chen et al. (2008) proposed algorithms and discussed computational issues for finding such designs in simple and quadratic models. Our experience with the proposed algorithm is that it may not work well with a more complex model and a more complicated efficiency function. Accordingly, we use the PSO method and tackle the more challenging problems when we have a cubic model for two situations; one concerns G-optimality when $X = Z$ and the other concerns an optimal extrapolation problem when $X \neq Z$. We choose two contrived and different efficiency functions since our purpose is just to ascertain whether the PSO method can find the numerically minimax optimal designs as judged by the equivalence theorem. In both cases, we use 128 particles and 100 iterations to find the minimax optimal designs.

Case 1: $X = Z = [-1, 1]$ and $\lambda(x) = 0.5x^2 + 1$. The generated design ξ found by PSO method is

$$\begin{pmatrix} -1.0000 & -0.4659 & 0.4659 & 1.0000 \\ 0.2113 & 0.2885 & 0.2883 & 0.2119 \end{pmatrix}.$$

Figure 4(a) is the graph of $c(x, \xi, \mu^*)$ and visually one may confirm the optimality of the generated design ξ from the PSO algorithm. A direct calculation shows the efficiency of the above design is 0.9996.

Case 2: $X = [-1, 1]$, $Z = [1, 1.5]$ and $\lambda(x) = x^4 + 1 + \sin^2(4x)$. The generated design ξ found by our PSO procedure is

$$\begin{pmatrix} -1.0000 & -0.4666 & 0.4666 & 1.0000 \\ 0.0665 & 0.2071 & 0.3942 & 0.3322 \end{pmatrix}.$$

The checking conditions of the equivalence theorem can be similarly seen to be satisfied in Figure 4(b). One can show that the efficiency of the generated design ξ is 0.9998.

We note that earlier work on optimal extrapolation designs for polynomial models were carried out in a series of papers by Kiefer and Wolfowitz (1964a,b, 1965); Levine (1966) assuming the efficiency function $\lambda(x)$ was a constant. Under the homoscedastic model, they were able to obtain analytic results when $X = [-1, 1]$ and $Z = [a, b]$ for selected values of a and b, including results for non-polynomial regression problems involving Chebyshev systems. Spruill (1984, 1990) worked on similar problems where bias was factored into the criterion as well. Interest in such design problems continues to date, see Broniatowski and Celant (2007) For example.

In the next section, we provide computational details and explain how the PSO method works, along with the choice of the tuning parameters for the flock size and number of iterations. As may have been already been noticed in the above examples, a couple of the designs found by the PSO method appeared to be slightly numerically different from the theoretical optimal designs. Our experience is that the discrepancy can be entirely attributed to the choices for these tuning parameters. We used the same PSO settings for all cases in the same example but this may not be adequate for all the cases in the sample example. When more particles and more iterations are added, the discrepancy usually disappears. For instance,when we used 128 particles and 200 iterations in Example 2, the results in the first row agreed up to 5 decimal places. Interestingly, when we used 256 particles and 500 iterations in Example 1, the discrepancy persisted and continued to do so when we increased the iteration and particle numbers to the thousands. Further investigation revealed that the smaller optimal design reported in the first row of Table 1 calculated from the formula was wrong. A direct calculation revealed the correct value was the one found by the PSO method.

4 PSO Method Explained

Computer algorithms have played and will continue to play an important role in our search of optimal designs. They are usually sequential in nature and typically involve the addition of a carefully selected new design point to the current design by mixing them appropriately to form a new design. The generated design accumulates many points or clusters of points as the algorithm proceeds and judicious rules for collapsing points into distinct points is required. The sequence of weights used to form a new design at each iteration is chosen so that they are all between 0 and 1, have a finite sum but does not converge too quickly or prematurely. Stopping rules are employed to decide when to terminate the search; they typically require either a maximum number of iterations allowed or when the change in the value of the optimality criterion in successive searches is negligible according to a user-selected tolerance level. An example of such an algorithm is the noted Fedorov-Wynn algorithm which is still popular after more than 3 decades of use. Details and exemplary codes for generating D- and c-optimal designs can be found in design monographs like Silvey (1980) and Pázman (1986). Research in this area remains active with new applications to tackle more challenging problems, such as finding optimal designs for pharmacokinetic models with subject random effects using a modified Fedorov's algorithm (Ogungbenro et al., 2005).

PSO is an iterative method that can be generically and readily coded as in Algorithm 1 to simulate the behaviors of bird flocking in search for food. 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.

Algorithm 1 ***PSO for the minimization problem*** (1)

(1) Initialize *particles*

(1.1) Choose *initial positions* $\mathbf{x}_i$ *and velocities* $\mathbf{v}_i$, *for* $i = 1, \ldots, n$.

(1.2) Calculate *fitness values* $f(\mathbf{x}_i)$.

(1.3) Determine *local and global best positions* $\mathbf{p}_i = \mathbf{x}_i$ *and* $\mathbf{p}_g$.

(2) Repeat until *stopping criteria are satisfied.*

(2.1) Calculate *particle velocity according Eq.*(2a).

(2.2) Update *particle position according Eq.* (2b).

(2.3) Calculate *fitness values* $f(\mathbf{x}_i)$.

(2.4) Update *best positions* $\mathbf{p}_i$ *and* $\mathbf{p}_g$ *and best values* $lbest_i$ *and* $gbest$.

(3) Output $\mathbf{p}_g = \arg\min f(\mathbf{x})$ *with* $gbest = f(\mathbf{p}_g)$.

Let X be a given domain and let $f(x)$ represent the function to be minimized. In design language, X is the design space and $f(x)$ is the objective function or design criterion. We want to find points x in X to minimize $f(x)$. To solve the minimization problem,

$$\min_{\mathbf{x} \in X} f(\mathbf{x}), \tag{1}$$

PSO is first initialized with a group of random particles (solutions). At every iteration, it searches for the optima by updating each particle its two "best" values. The first one is the best solution (fitness) it has achieved so far. This local best value is called *lbest* and is stored. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called *gbest*. After finding the two best values, the particle updates its velocity and positions using the equations.

$$\mathbf{v}_{i+1} = \omega_i \mathbf{v}_i + c_1 \boldsymbol{\beta}_1 (\mathbf{p}_i - \mathbf{x}_i) + c_2 \boldsymbol{\beta}_2 (\mathbf{p}_g - \mathbf{x}_i), \tag{2a}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i. \tag{2b}$$

Here, $\mathbf{v}_i$ is the particle velocity, ω_i is the inertia weight that modulates the influence of the former velocity and can be a constant or a decreasing function (Shi and Eberhart, 1998a,b; Chatterjee and Siarry, 2006; Fan and Chang, 2007), $\mathbf{x}_i$ is the current particle (solution). The vectors $\mathbf{p}_i$ and $\mathbf{p}_g$ denote the local best position for the ith particle and the global best position for all particles, respectively. Furthermore, $lbest_i = f(\mathbf{p}_i)$ and $gbest = f(\mathbf{p}_g)$ for any objective function f. The constant c_1 is the cognitive learning factor, c_2 is the social learning factor. The variables $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are random vectors with the same dimension as $\mathbf{x}_i$ and in the first equation, multiplication of two random vectors are taken to be componentwise. The two constants c_1 and c_2 control how each particle moves toward its own local best position and overall global best position, respectively. For many applications, $c_1 = c_2 = 2$ seems to work well

(Kennedy, 1997) and so we also use these two values in our pseudo code in our PSO procedure below.

Note that the particles' velocities on each dimension are clamped to a user-specified maximum velocity V_{max}. If the sum of the accelerations cause the velocity on that dimension to exceed V_{max}, the velocity on that dimension will be limited to V_{max}.

4.1 Proposed PSO for the minimax problems

The problems in Section 3 can be formulated as a minimax problem

$$\min_{\mathbf{u}\in\mathcal{U}} \max_{\mathbf{v}\in\mathcal{V}} g(\mathbf{u},\mathbf{v}), \tag{3}$$

where $\mathcal{U}$ and $\mathcal{V}$ are the corresponding design domains. References that use PSO to solve the minimax type problems in literature are sparse. One example is Laskari et al. (2002). However, they consider the problems that there are only finite elements in $\mathcal{V}$. Thus their algorithm is not suitable for our problems. Here, we propose two PSO-based algorithms to solve such minimax problems (3).

The first algorithm is an Alternate-Conditional PSO. The main idea is to solve the minimization and the maximization problem alternately by using PSO (Algorithm 1) until the fixed-point is reached. In other words, we solve for $\mathbf{u}$ (or $\mathbf{v}$) conditionally by fixing the previous $\mathbf{v}$ (or $\mathbf{u}$), so that we can find the intermediate solutions of $\mathbf{u}$ and $\mathbf{v}$ sequentially and alternately one after another. The key idea is presented in Algorithm 2.

Algorithm 2 ***Alternate-Conditional PSO for the minimax problem*** (3)

(1) Choose *an initial guess* $\mathbf{u_0}$ *and* Set $j = 0$.

(2) Repeat *until converge to a fixed-point* $(\mathbf{u}^*, \mathbf{v}^*)$.

- *(2.1)* Find $\mathbf{v_{j+1}} = \arg\max_{\mathbf{v}\in\mathcal{V}} g(\mathbf{u_j}, \mathbf{v})$ *by Algorithm 1.*
- *(2.2)* Find $\mathbf{u_{j+1}} = \arg\min_{\mathbf{u}\in\mathcal{U}} g(\mathbf{u}, \mathbf{v_{j+1}})$ *by Algorithm 1.*
- *(2.3)* Set $j = j + 1$.

(3) Output $\mathbf{u}^*$ *and* $\mathbf{v}^*$ *as the solution with best value* $g(\mathbf{u}^*, \mathbf{v}^*)$.

Although the the standard PSO (i.e. Algorithm 1) is intended to solve an minimization problem, to solve the maximization problem in Step (2.1) of Algorithm 2 by PSO, we simply define $f(\mathbf{x}) = -g(\mathbf{u_j}, \mathbf{x})$. In Step (2.2), we define $f(\mathbf{x}) = g(\mathbf{x}, \mathbf{v_{j+1}})$ and then apply PSO. One practical stopping criterion in Step (2) is to stop the Repeat-loop after a pre-defined computational budget (e.g. iteration number) is reached. Another practical stopping criterion is to stop the loop for small changes of the successive intermediate $\mathbf{u_{j+1}}$ and $\mathbf{v_{j+1}}$.

The second algorithm is a Nested PSO that involves solving an outer and an inner minimization problems by PSO. To explain the idea, we first define an outer objective function

$$f_{outer}(\mathbf{x}) = \max_{\mathbf{v}\in\mathcal{V}} g(\mathbf{x}, \mathbf{v}). \tag{4}$$

and rewrite the minimax problem (3) as

$$\min_{\mathbf{x}\in X=\mathcal{U}} f_{outer}(\mathbf{x}). \tag{5}$$

It is clear that to calculate the fitness value $f_{outer}(\mathbf{x})$ is equivalent to solve the maximization problem defined in (4). By defining

$$f_{inner}(\mathbf{y}) = -g(\mathbf{x}, \mathbf{y}), \tag{6}$$

the maximization problem can be rewritten as

$$\min_{\mathbf{y}\in\mathcal{Y}=\mathcal{V}} f_{inner}(\mathbf{y}). \tag{7}$$

Therefore, we have the following equivalent optimization problems:

$$\min_{\mathbf{u}\in\mathcal{U}} \max_{\mathbf{v}\in\mathcal{V}} g(\mathbf{u}, \mathbf{v}) \equiv \min_{\mathbf{x}\in X=\mathcal{U}} f_{outer}(\mathbf{x}) \equiv \min_{\mathbf{x}\in X=\mathcal{U}} \left(\min_{\mathbf{y}\in\mathcal{Y}=\mathcal{V}} f_{inner}(\mathbf{y}) \right). \tag{8}$$

Consequently we can solve the nested minimization problem defined in (8) by the standard PSO as described in Algorithm 3. Note that Algorithm 3 is identical to Algorithm 1 in terms of solving the minimization problem $\min_{\mathbf{x}\in X=\mathcal{U}} f_{outer}(\mathbf{x})$, except the calculations of fitness values in Steps (1.2) and (2.3) that involves solving another minimization problem $\min_{\mathbf{y}\in\mathcal{Y}=\mathcal{V}} f_{inner}(\mathbf{y})$.

Algorithm 3 ***Nested PSO for the minimax problem*** (3).

(1) Initialize *particles*

- *(1.1)* Choose *initial positions* $\mathbf{x}_i$ *and velocities* $\mathbf{v}_i$*, for* $i = 1, \ldots, n$.
- *(1.2)* Calculate *fitness values* $f_{outer}(\mathbf{x}_i)$ *by solving* (7) *by Algorithm 1.*
- *(1.3)* Determine *local and global best positions* $\mathbf{p}_i = \mathbf{x}_i$ *and* $\mathbf{p}_g$.

(2) Repeat until *stopping criteria are satisfied.*

- *(2.1)* Calculate *particle velocity according Eq.*(2a).
- *(2.2)* Update *particle position according Eq.* (2b).
- *(2.3)* Calculate *fitness values* $f_{outer}(\mathbf{x}_i)$ *by solving* (7) *by Algorithm 1.*
- *(2.4)* Update *best positions* $\mathbf{p}_i$ *and* $\mathbf{p}_g$ *and best values* $lbest_i$ *and* $gbest$.

(3) Output $\mathbf{p}_g = \arg\min f_{outer}(\mathbf{x})$ *with* $gbest = f_{outer}(\mathbf{p}_g)$.

While Algorithms 2 and 3 can be used to solve the minimax problem (3), we recommend using the latter. This suggestion is not only supported by our numerical

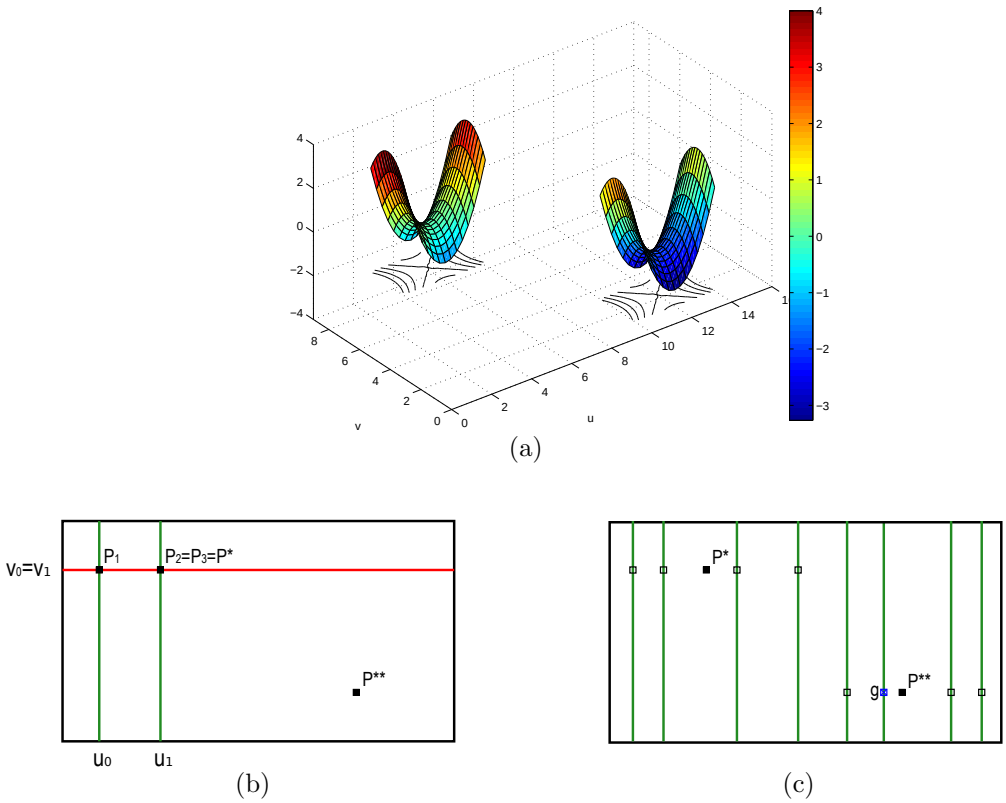


Figure 5: (a) Plot of a conceptual surface and the corresponding contours that contain two saddles with different heights. (b) Starting with the initial guess $\mathbf{u_0}$, Algorithm 2 goes from P_1 to P_3 and stops at P^*. (c) The first iteration is performed by Algorithm 3. The fitness values (also the $lbest_i$ in the first iteration) are indicated by the squares. The current *gbest* is indicated by $\times$ and labeled by "g". Note that in (b) and (c), the two saddle points are indicated by P^* and P^{**}.

experience that Algorithm 3 is more robust and efficient numerically, but there is also some rationale in itself. Algorithm 2 finds the solution point-by-point, in the sequence of $(\mathbf{u_j}, \mathbf{v_{j-1}})$, $(\mathbf{u_j}, \mathbf{v_j})$, $(\mathbf{u_{j+1}}, \mathbf{v_j})$,..., etc but it does not take advantage of the simultaneous searches done by all the particles. PSO is simply used to solve the minimization and maximization problems to move $\mathbf{u}$ and $\mathbf{v}$, respectively. In contrast, Algorithm 3 lets multiple particles calculate the fitness values $f_{outer}(\mathbf{x}_i)$ simultaneously. Each of the particle then solves one inner minimization problem by PSO so that the outer particles have better chance to explore the whole domain.

Figure 5 conceptually illustrates how Algorithms 2 and 3 search for the optimal points. The figure shows why Algorithm 3 has a better chance to find the global optimal solution. Part (a) of the figure defines a surface containing two saddles. The left-top saddle has a larger function value at the saddle point $P^* = (\mathbf{u}^*, \mathbf{v}^*)$.

The right-bottom saddle has a smaller function value at the the saddle point $P^{**} = (\mathbf{u}^{**}, \mathbf{v}^{**})$. Figure 5(b) shows how Algorithms 2 works. Starting from the initial guess $\mathbf{u_0}$, Algorithms 2 solves for $\mathbf{v_1} = \arg\max_{\mathbf{v}\in\mathcal{V}} g(\mathbf{u_0}, \mathbf{v})$ (i.e. with conditional on the vertical line passing $\mathbf{u_0}$) and $\mathbf{u_1} = \arg\max_{\mathbf{u}\in\mathcal{U}} g(\mathbf{u}, \mathbf{v_1})$ (with conditional on the horizontal line passing $\mathbf{v_1}$) to reach $P_1 = (\mathbf{u_1}, \mathbf{v_1})$. In the next iteration, the algorithm finds P_2. Such process is continued to reach the fixed-point P^*. Figure 5(c) shows how Algorithm 3 works. Starting from eight initial guesses on the $\mathbf{u}$-axis, eight particles move around the $\mathbf{u}$-axis to find the solution of $\min_{\mathbf{x}\in X=\mathcal{U}} f_{outer}(\mathbf{x})$ simultaneously. Each particle solves $\min_{\mathbf{y}\in\mathcal{Y}=\mathcal{V}} f_{inner}(\mathbf{y})$ in the corresponding domain (identified by the vertical lines in the figure) to calculate the fitness function values. It is very possible that some particles are close to the global optimal point P^{**} (e.g. the *gbest* point "g" in the first iteration) with smaller f_{outer} value. Algorithm 3 thus can gradually move toward P^{**}. The figure also suggests that the final result of Algorithm 2 depends on the initial choice and thus can be considered as a local method. Algorithm 3, however, uses multiple particles to search the whole $\mathcal{U}$ domain and thus can be considered as a global method.

In summary, we made two amendments of the PSO method to find the minimax optimal design and we have better experience and success with the Nested PSO method. We therefore recommend Algorithm 3 to find minimax optimal designs for regression problems. All our minimax optimal designs in this paper are obtained using Algorithm 3.

To apply the Nested PSO method to the minimax design problems, suppose $\Phi(\xi)$ is the maximum value for the inner optimization problem. To fix ideas, consider the minimax design problem for the heteroscedastic regression model. We set $\Phi(\xi) = \max_{z\in Z} d(z, \xi)$ and solve the outer optimization problem, $\min_\xi \Phi(\xi)$ in PSO method by treating each particle $\mathbf{x}$ as a design ξ. So $\mathbf{x}$ can be represented as $\mathbf{x} = (x_1, \ldots, x_k, p_1, \ldots, p_k)^\top$, where $x_i, i = 1, \ldots, k$ are the support points in a pre-specified design space and $p_i,\ i = 1, \ldots, k$ are the corresponding weights with $1 > p_i \geq 0$ and $\sum_{i=1}^k p_i = 1$.

5 Tuning Parameters in the PSO Method for a Specific Example

There are two non-random numbers c_1 and c_2 in the velocity equations and they have default values set equal to 2. This choice seems to work well generally in the literature and it is the case for our work here as well. If necessary, other values may be tried out to improve the speed. Acceleration of the performance of the PSO method can also be achieved by using parallel computers. For example, Hung and Wang (ress) suggested a Graphics Processing Unit (GPU) based PSO and reported up to 280X accelerations with respect to Central Processing Unit (CPU) based PSO while solving high-dimension complicated minimization problems.

The PSO method also requires that we specify upfront the number of support

points required in the optimal minimax design. Because we prefer non-singular optimal designs, our initial choice for the number of support points for each particle in the flock is the number of the unknown parameters in the model. After the optimal design is found, one may use the equivalence theorem to verify its optimality; otherwise, we increase the number of support points by a unit each time until the generated design found by the PSO method is confirmed to be optimal.

We now briefly investigate the effect of other tuning parameters in the PSO method for finding a heteroscedastic G-optimal design for a specific case study. As a start, we considered the simple linear model on the interval $X = [-1, 1]$ with different types of efficiency functions: (i) $\lambda(x) = x + 5$, (ii) $\lambda(x) = \exp(-5x^2)$, (iii) $\lambda(x) = 0.5x^2 + 1$, (iv) $\lambda(x) = 4 + x - x^2$, (v) $\lambda(x) = x^4 + 1 + \sin^2(4x)$ and (vi) $\lambda(x) = \exp(-1-x)/(1+\exp(-1-x))^2$. We varied the flock size using 32, 64 and 128 and 50 or 100 number of iterations for the outer problem to search for the optimal design. For the inner optimization problem, we used 32, 64 and 128 for the flock size and 50 or 100 inner iterations for the inner problem to search for μ^*. For each of these $3 \times 3 \times 2 \times 2$ configurations, we applied the PSO method 3 times and evaluate the proximity of the generated design to the optimum. We do this by reporting the lower bound of the G-efficiency of the generated optimal design. This is possible because the function is convex; see Pázman (1986), Wong and Cook (1993) for example. This simple set up proved not useful for our purpose here because the PSO method was very efficient and in all cases, generated a design with at least 99% efficiency.

We next proceeded to look for a situation where varying tuning parameters in the PSO procedure has appreciable effect on the quality of the generated design. After a relatively extensive search, we present here a case study of the impact of tuning parameters in the PSO method for finding the minimax optimal design using a cubic polynomial model with efficiency function (v) on the interval $X = [-1, 1]$. The efficiency function (v) is contrived and just complicated enough with global peaks at ± 1 and two local peaks near ± 0.4 to slow down the PSO procedure and provide us with some insight on how choice of the numbers of inner and outer iterations and the numbers of inner and outer particles impact the search. Table 3 displays the frequency distribution on the lower efficiency bounds from running the PSO 100 times for each of the configuration shown in the table. In addition, we repeat this process two more times to ascertain the stability of the method. Generally the numerical results suggest the following observations.

1. When the numbers of inner particles and outer particles are fixed, designs of higher efficiencies are obtained with more inner and outer iterations.

2. When the inner and outer iterations are fixed, we have higher chance of finding more efficient designs by increasing the numbers of inner and outer particles.

3. Our numerical results generally suggest that designs we obtain more efficient designs when we use more inner particles than outer particles, others things being equal.

4. The numbers of inner and outer particles are a bit more important than the numbers of inner and outer iterations. For example, consider the two cases when we have (inner particles, outer particles, inner iterations, outer iterations) = (128, 64, 50, 50) and (64, 32, 100, 100). The number of inner iterations required for the nested PSO are the same for the two cases, i.e. 128*50*(64*50) = 64*32*100* 100 but we have higher frequency of obtaining a design with at least 90% efficiency for the case of (128, 64, 50, 50) compared with the case when we have (64, 32, 100, 100).

The above conclusions are drawn from a specific case study and may not apply to other design problems. We expect however they are likely reasonable guidelines. The upshot is that all numerical examples that we have worked with showed PSO worked remarkably well for generating minimax optimal designs, and by implication, other optimal designs that are easier to determine such as when the criterion function is differentiable. Indeed a direct application of the PSO method to find a locally D-optimal design for the 2-parameter logistic model with $X = [-1, 1]$ showed results reported on page 72 of the book by Silvey (1980) from Ford's doctorate thesis in (1976) were incorrect. To correct the results, the locally D-optimal designs for case (ii) and case (iii) have to be reversed and we have confirmed our findings using the equivalence theorem. In many of the cases we tried out with different nominal values for the model, the PSO always found the locally D-optimal designs in 1 or 2 seconds of CPU time.

Table 3: Performance of the PSO method for finding heteroscedastic G-optimal designs for a cubic polynomial model on $X = [-1, 1]$ when the efficiency function is $\lambda(x) = x^4 + 1 + \sin^2(4x)$ and the tuning parameters are changed. The numbers in each cell reports the frequency distribution of the efficiency lower bounds for each configuration after running PSO 100 times. A total of three replicates were used to ascertain stability of the PSO method.

Part I. Number of inner particles is 32.

(outer particles, inner particles)	Replicate	lower bound efficiency	(outer iterations, inner iterations) (50, 50)	(100, 50)	(50, 100)	(100, 100)
(32, 32)	1st	< 0.90	52	32	38	13
		$0.90 - 0.95$	17	15	12	17
		$0.95 - 1.00$	31	53	50	70
	2nd	< 0.90	52	35	38	19
		$0.90 - 0.95$	18	10	21	23
		$0.95 - 1.00$	30	55	41	58
	3rd	< 0.80	48	28	38	21
		$0.90 - 0.95$	19	21	18	13
		$0.95 - 1.00$	33	51	44	64
(64, 32)	1st	< 0.90	28	19	11	5
		$0.90 - 0.95$	13	11	17	6
		$0.95 - 1.00$	59	70	72	89
	2nd	< 0.90	24	20	15	14
		$0.90 - 0.95$	18	9	10	11
		$0.95 - 1.00$	58	71	75	75
	3rd	< 0.90	21	19	14	10
		$0.90 - 0.95$	18	8	15	6
		$0.95 - 1.00$	61	73	71	84
(128, 32)	1st	< 0.90	24	17	9	5
		$0.90 - 0.95$	6	5	10	4
		$0.95 - 1.00$	70	78	81	91
	2nd	< 0.90	21	18	3	5
		$0.90 - 0.95$	8	4	6	8
		$0.95 - 1.00$	71	78	91	87
	3rd	< 0.90	14	18	4	2
		$0.90 - 0.95$	7	11	6	8
		$0.95 - 1.00$	79	71	90	90

Part II. Number of inner particles is 64.

(outer particles, inner particles)	Replicate	lower bound efficiency	(outer iterations, inner iterations) (50, 50)	(100, 50)	(50, 100)	(100, 100)
(32, 64)	1st	< 0.90	50	21	26	10
		$0.90 - 0.95$	17	18	17	8
		$0.95 - 1.00$	33	61	57	82
	2nd	< 0.90	39	19	30	6
		$0.90 - 0.95$	19	14	15	15
		$0.95 - 1.00$	42	67	55	79
	3rd	< 0.90	37	19	18	3
		$0.90 - 0.95$	19	14	24	10
		$0.95 - 1.00$	44	67	58	87
(64, 64)	1st	< 0.90	9	5	7	1
		$0.90 - 0.95$	19	8	7	4
		$0.95 - 1.00$	72	93	86	95
	2nd	< 0.90	18	5	5	0
		$0.90 - 0.95$	16	8	15	7
		$0.95 - 1.00$	66	87	80	93
	3rd	< 0.9	12	11	10	2
		$0.90 - 0.95$	12	7	8	1
		$0.95 - 1.0$	76	82	82	97
(128, 64)	1st	< 0.90	5	7	0	2
		$0.90 - 0.95$	3	3	4	4
		$0.95 - 1.0$	92	90	96	94
	2nd	< 0.90	8	10	3	0
		$0.90 - 0.95$	6	8	6	3
		$0.95 - 1.00$	86	82	91	97
	3rd	< 0.90	6	5	3	0
		$0.90 - 0.95$	10	7	6	0
		$0.95 - 1.00$	84	88	91	100

Part III. Number of inner particles is 128.

(outer particles, inner particles)	Replicate	lower bound efficiency	(outer iterations, inner iterations) (50, 50)	(100, 50)	(50, 100)	(100, 100)
(32, 128)	1st	< 0.90	17	13	5	3
		$0.90 - 0.95$	26	13	20	9
		$0.95 - 1.00$	53	74	75	88
	2nd	< 0.90	21	6	18	2
		$0.90 - 0.95$	25	12	17	9
		$0.95 - 1.00$	54	82	65	89
	3rd	< 0.90	29	5	8	5
		$0.90 - 0.95$	16	12	18	10
		$0.95 - 1.00$	55	83	74	85
(64, 128)	1st	< 0.90	2	2	2	0
		$0.90 - 0.95$	15	3	8	0
		$0.95 - 1.00$	83	95	90	100
	2nd	< 0.90	6	1	4	0
		$0.90 - 0.95$	10	7	8	2
		$0.95 - 1.00$	84	92	89	98
	3rd	< 0.90	2	1	2	0
		$0.90 - 0.95$	11	3	9	0
		$0.95 - 1.00$	87	96	89	100
(128, 128)	1st	< 0.90	1	0	0	0
		$0.90 - 0.95$	1	1	4	0
		$0.95 - 1.00$	98	99	96	100
	2nd	< 0.90	0	0	0	0
		$0.90 - 0.95$	1	1	2	1
		$0.95 - 1.00$	99	99	98	99
	3rd	< 0.90	1	0	0	0
		$0.90 - 0.95$	3	0	2	0
		$0.95 - 1.00$	96	100	98	100

6 Discussion

In today rapidly rising cost of experimentation, optimal design ideas take on an increasingly important role. A well designed study is able to answer the scientific questions accurately and with minimum cost. It is therefore not surprising that optimal experimental designs continue to find increasingly more applications in different fields and novel applications are continually seen in traditional areas, see Berger et al. (2005), for example.

Recent development in the field includes incorporating multiple objectives in the design criteria as opposed to the traditional way of designing under one criterion and hope that the optimal design is also adequate under the other criteria. To promote use of optimal design ideas in practice, it is helpful to facilitate practitioners have easy access to optimal designs. One way to promote optimal design ideas is to provide a website that readily generates different types of optimal designs for a variety of

models. One such site is at http://www.biostat.ucla.edu/optimal-design housed in the Department of Biostatistics at UCLA. The site has a list of frequently used biological regression models and optimality criteria that the visitor can select from to generate his or her tailor-made optimal design after inputting design parameters for the problem. We have now MATLAB codes based on PSO methodology for finding minimax optimal designs and also codes for visual appreciation of the dynamic search of the optimum by the flock as the iterations proceed. We plan to upload the codes for some minimax design problems. Our long term goal is to allow user change the design criterion and input the information matrix themselves and let PSO does the rest of the work. This way we will not have to supply an endless list of program codes for different models and design criteria for the practitioners.

In summary, we have demonstrated that the PSO method is an effective and powerful tool for generating different types of optimal experimental designs. PSO offers both many exciting and immediate opportunities and applications. One of our next goals is to use the PSO method to find continuous optimal designs for two or more objective simultaneously and explore its applicability to find exact optimal designs. We believe we have only just explored PSO versatility and its ability to generate quickly a variety of optimal experimental designs. We are impressed with the PSO capabilities and believe the introduction of PSO to find optimal designs represents an advance in the field. Clearly, PSO does not require the objective function to be differentiable as shown in all our examples. As may have been suggested in the applications noted in the opening paragraph, one very important property of PSO is that it also does not even require the objective function to be convex! The added bonus of working with convex objective functions is that an equivalence theorem becomes possible.

It is worth repeating that formulae for optimal designs rarely exist and when they do, they are invariably complex enough not to be useful to the practitioners. The practical implication of our work here is that practitioners can now simply write a few lines of codes for the PSO method and generate tailor made optimal designs for their problems quickly and study them before implementation. Our experience based on examples shown here and elsewhere is that the designs generated by the PSO method all attain high efficiencies very quickly. If necessary, the skeptical researcher can still verify the optimality of the generated design using an equivalence theorem when the objective criterion is convex.

References

Berger, M. P. F., King, J., and Wong, W. K. (2000). Minimax D-optimal Designs for Item Response Theory Models. *Psychometrika*, 65(3):377–390.

Berger, M. P. F., Wong, W. K., and Wiley, J. (2005). *Applied Optimal Designs*. Wiley Online Library.

Broniatowski, M. and Celant, G. (2007). Optimality and Bias of Some Interpolation and Extrapolation Designs. *Journal of Statistical Planning and Inference*, 137(3):858–868.

Brown, L. D. and Wong, W. K. (2000). An Algorithmic Construction of Optimal Minimax Designs for Heteroscedastic Linear Models. *Journal of Statistical Planning and Inference*, 85(1-2):103–114.

Chatterjee, A. and Siarry, P. (2006). Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization. *Computers and Operations Research*, 33(3):859–871.

Chen, R. B., Wong, W. K., and Li, K. Y. (2008). Optimal Minimax Designs over a Prespecified Interval in a Heteroscedastic Polynomial Model. *Statistics and Probability Letters*, 78(13):1914–1921.

Clerc, M. (2006). *Particle Swarm Optimization*. Wiley-ISTE.

Dette, H., Melas, V. B., and Pepelyshev, A. (2003). Standardized Maximin E-optimal Designs for the Michaelis-Menten Model. *Statistica Sinica*, 13(4):1147–1164.

Dette, H. and Sahm, M. (1998). Minimax Optimal Designs in Nonlinear Regression Models. *Statistica Sinica*, 8:1249–1264.

Dette, H. and Wong, W. K. (1999). E-optimal Designs for the Michaelis-Menten Model. *Statistics and Probability Letters*, 44(4):405–408.

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

Eberhart, R., Shi, Y., and Kennedy, J. (2001). *Swarm Intelligence*. Elsevier.

Fan, S. K. S. and Chang, J. M. (2007). A Modified Particle Swarm Optimizer Using an Adaptive Dynamic Weight Scheme. In *Proceedings of the 1st International Conference on Digital Human Modeling*, pages 56–65. Springer-Verlag.

Fedorov, V. V. (1972). *Theory of Optimal Experiments*. Academic press.

Gaffke, N. and Krafft, O. (1982). Exact D-optimum Designs for Quadratic Eegression. *Journal of the Royal Statistical Society, Ser. B*, 44(3):394–397.

Hung, Y. and Wang, W. (In Press). Accelerating Parallel Particle Swarm Optimization via GPU. *Optimization Methods and Software*.

Kennedy, J. (1997). The Particle Swarm: Social Adaptation of Knowledge. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 303–308. IEEE.

Kiefer, J., Brown, L., and Institute of Mathematical Statistics (Beachwood, E. U. (1985). *Jack Carl Kiefer Collected Papers: Design of Experiments*. Springer.

Kiefer, J. and Wolfowitz, J. (1964a). Optimum Extrapolation and Interpolation Designs I. *Annals of the Institute of Statistical Mathematics*, 16(1):79–108.

Kiefer, J. and Wolfowitz, J. (1964b). Optimum Extrapolation and Interpolation Designs II. *Annals of the Institute of Statistical Mathematics*, 16(1):295–303.

Kiefer, J. and Wolfowitz, J. (1965). On a Theorem of Hoel and Levine on Extrapolation Designs. *The Annals of Mathematical Statistics*, 36(6):1627–1655.

King, J. and Wong, W. K. (1998). Optimal Minimax Designs for Prediction in Heteroscedastic Models. *Journal of Statistical Planning and Inference*, 69(2):371–383.

King, J. and Wong, W. K. (2000). Minimax D-optimal Designs for the Logistic Model. *Biometrics*, 56(4):1263–1267.

Laskari, E., Parsopoulos, K., and Vrahatis, M. (2002). Particle swarm optimization for minimax problems. In *WCCI*, pages 1576–1581. IEEE.

Lazinica, A. (2009). *Particle Swarm Optimization*. InTech.

Levine, A. (1966). A Problem in Minimax Cariance Polynomial Extrapolation. *The Annals of Mathematical Statistics*, 37(4):898–903.

Murty, V. N. (1971). Minimax Designs. *Journal of the American Statistical Association*, 66:319–320.

Ogungbenro, K., Graham, G., Gueorguieva, I., and Aarons, L. (2005). The Use of a Modified Fedorov Exchange Algorithm to Optimise Sampling Times for Population Pharmacokinetic Experiments. *Computer Methods and Programs in Biomedicine*, 80(2):115–125.

Olsson, A. E. (2011). *Particle Swarm Optimization: Theory, Techniques and Applications*. Nova Publications.

Pázman, A. (1986). *Foundations of Optimum Experimental Design*. Springer-Verlag.

Shi, Y. and Eberhart, R. (1998a). A Modified Particle Swarm Optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE.

Shi, Y. and Eberhart, R. (1998b). Parameter Selection in Particle Swarm Optimization. In *Evolutionary Programming VII*, pages 591–600. Springer.

Silvey, S. D. (1980). *Optimal Design*. Chapman & Hall.

Spruill, M. C. (1984). Optimal Designs for Minimax Extrapolation. *Journal of Multivariate Analysis*, 15(1):52–62.

Spruill, M. C. (1990). Optimal Designs for Multivariate Interpolation. *Journal of Multivariate Analysis*, 34(1):141–155.

Wong, W. K. (1992). A Unified Approach to the Construction of Minimax Designs. *Biometrika*, 79(3):611.

Wong, W. K. and Cook, R. D. (1993). Heteroscedastic G-optimal Designs. *Journal of the Royal Statistical Society, Ser. B*, 55(4):871–880.

Yang, X. S. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*. Wiley.