January 2015

LASSO-OPTIMAL SUPERSATURATED DESIGN AND ANALYSIS FOR FACTOR SCREENING IN SIMULATION EXPERIMENTS

DADI XING
Purdue University

Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Dadi Xing

Entitled
LASSO-OPTIMAL SUPERSATURATED DESIGN AND ANALYSIS FOR FACTOR SCREENING IN SIMULATION EXPERIMENTS

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Hong Wan	
Chair	
Michael Yu Zhu	
Co-chair	
Bruce A. Craig	
Andrew Lu Liu	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Hong Wan

Approved by: Abhijit Deshmukh 12/3/2015

Head of the Departmental Graduate Program Date

LASSO-OPTIMAL SUPERSATURATED DESIGN AND ANALYSIS FOR FACTOR SCREENING IN SIMULATION EXPERIMENTS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Dadi Xing

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2015

Purdue University

West Lafayette, Indiana

To my wife Yanfang, my kids Bailee and Kaylee and my parents.

ACKNOWLEDGMENTS

I thank my parents for their seless love, without which I would not even have the opportunity to write this dissertation. I appreciate my wife's tolerance, patience and encouragement, especially during the time when I was on crutches. I own a big debt of gratitude to Professor Hong Wan for her endless help and patient in supervising my research. In the past five years, Professor Hong Wan not only inspired me in research, but also provided me with abundant encouragement whenever I was frustrated and under pressure. I am also grateful to Professor Yu Zhu for his technical guidance, patience and extra hours he spent on me. I would like to thank all of my committee members for their generous help, without which I would not be able to finish my Ph.D. research. My colleagues Akash Agarwal, Wenyu Wang and Qin Dong also helped me a lot which I do appreciate.

PREFACE

Complex systems such as large-scale computer simulation models typically involve a large number of factors. When investigating such a system, screening experiments are often used to sift through these factors to identify a subgroup of factors that most significantly influence the interested response. The identified factors are then subject to further and more carefully investigation. A good screening experiment can efficiently allocate the experimental resources to those important factors, and can therefore greatly expand the ability of analysts and decision makers to gain insights from a complicated system with a reasonable amount of cost. In our work, we propose a unified framework for design and analysis of screening experiments with application in large scale simulation models. In particular, we use supersaturated designs as the general design strategy for the experiment of the screening process and further to use the lasso based variable selection methods as the general approach for the analysis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Xing, Dadi Ph.D., Purdue University, December 2015. LASSO-OPTIMAL SUPERSATURATED DESIGN AND ANALYSIS FOR FACTOR SCREENING IN SIMULATION EXPERIMENTS . Major Professor: Hong Wan and Michael Yu Zhu.

Complex systems such as large-scale computer simulation models typically involve a large number of factors. When investigating such a system, screening experiments are often used to study these factors to identify a subgroup of factors that most significantly influence the interested response.

A typical screening procedure consists of two steps: the experiment step and the analysis step. With a large number of factors, both of these two steps can be extremely challenging in practice. Currently there exists a disparity between the experimental designs and the analysis methods used for screening experiments. To address this disparity, our work focuses on analysis-method-directed optimal supersaturated designs for high-dimensional screening experiments. We use supersaturated designs and Lasso for the two steps of screening experiments, respectively; and we propose to optimize supersaturated designs toward the best performance of the Lasso. Specifically, we studied the variable selection performances of the lasso under finite samples and establish the exact relationship between the performances and the design matrix; we then developed the optimality criteria for constructing supersaturated designs that guarantee optimal variable selection performance, we studied the theoretical properties of the proposed criteria, and further developed efficient algorithms to construct optimal supersaturated designs. We also developed a software package capable of constructing optimal supersaturated designs and analyzing data generated from supersaturated experiments, and applied the package to study mission critical computer simulation models. We expect the research to advance the experimental design frontier and have significant impact on analysis of large systems.

1. INTRODUCTION

1.1 Screening Experiment

1.1.1 General Introduction

Complex systems such as large scale computer simulation models typically involve a large number of factors. Here ”factors” refers to parameters or input variables that may influence the performance of a system [1], and their levels can be continuous or discrete. When investigating such a system, screening experiments are commonly used to sift through the factors to identify a subgroup that significantly affects the system based on a certain performance measure. The identified factors are then subject to further investigation. The sparsity of effects principle is often used to justify screening experiments, which states that in general only a small number of factors are responsible for most of the variation in a given response [2]. A successful screening experiment can help investigators allocate experimental resources to important factors and gain insights from a complicated system at a reasonable amount of cost.

The demand for efficient and effective methods for design and analysis of screening experiments arises in a variety of application areas. One such area is the analysis of large-scale simulated systems. Despite the prevalence of using simulated systems as a decision-support tool, there exists a huge gap between the complexity of the systems and the availability of proper methods that can be used to investigate and analyze these systems. A simulation system may take months or years to develop and has literally thousands of components (factors) with unknown effects on the system's performance. For example, drug screening, the computer simulation of nuclear test and the Future Combat Systems Family(FCS) from the United States Department of Defense, the . A single simulation experiment takes many hours or days to run with as many as 100,000 user selectable input values [3]. Therefore efficient and effective analytical methods for such a large experiments become critical. The lack of these methods severely limits investigators or decision makers' capability to gain insights about the systems. The primary goal of this work is to develop a general framework with effective statistical methods for design and analysis of screening experiments to bridge the afore-mentioned gap.

1.1.2 Two Step of Screening Experiment

A typical screening process consists of two steps: the experiment step and the analysis step. During the experiment step, the factors under study are systematically varied and the response variable of interest is measured; and during the analysis step, the generated data are analyzed to identify the important factors. Due to the presence of a large number of factors, both of these two steps can be extremely challenging in practice. For the experiment step, proper experimental design strategies are needed to manipulate the tens of thousands of factors, whereas for the analysis step, efficient and effective analysis methods are required in order to identify the important factors with limited data.

1.1.2.1 Experiment Step

To overcome the challenge in the experiment step, various techniques have been proposed in the literature and used in practice, which include sequential strategies and group designs. Sequential design updates the experimental design while collecting observations in a sequential manner, typically using Bayesian analysis and proper optimization criteria [4] [5] [6]. It is based on the idea that more observations result in more information and further lead to better selection of design points. To handle large-scale screening experiments, the most common sequential approach is group design. Group design is to arrange factors into groups and then treat each group as a factor and design the experiment accordingly. One popular method for group design is sequential bifurcation and its stochastic version, controlled sequential bifurcation [7] [8]. For factorial group designs, see Dean and Lewis [9]. While the group design strategy can significantly reduce the number of factors in an experiment, its disadvantage is also obvious.

When factors with positive and negative effects are grouped together, they will cancel each other. Therefore, the group design strategy relies on prior knowledge of the system. Furthermore, a group design become inefficient when important factors are not grouped together and dispersed into different groups [10].

An alternative design strategy for large scale screening experiments is to use supersaturated designs. A supersaturated design is a fractional factorial design with more factors than experimental runs, which does not have sufficient degrees of freedom for estimating all main effects of the factors. Supersaturated designs were first formally introduced by Booth and Cox [11]. The papers by Lin [12] and Wu [13] has stimulated the interest and research effort in supersaturated designs, and since then, various approaches or criteria, theoretical or numerical, have been proposed for constructing supersaturated designs in the literature. The most compelling motivation for the supersaturate design strategy is that it shares the same basic principle (i.e. the sparsity of effects principle) as screening experiments. The second motivation is that the number of runs required by a supersaturated design is much smaller than the number of factors, therefore, it is most suitable for screening experiments involving a large number of factors. And the third motivation is that recent advances on variable selection in high dimensional data analysis not only provide effective tools for identifying important factors from data generated from supersaturated screening experiments but also help unify the theory and methods used in both the experiment and analysis steps of these experiments.

1.1.2.2 Analysis Step

As discussed previously, the analysis step of a screening process is to identify a subgroup of important factors; and thus it is essentially a variable selection step. Classical variable selection methods include stepwise regression, best subset selection, Mallow's C_p [14], Akaike Information criterion (AIC) [15], Bayesian Information Criterion (BIC) [16], and cross validation [17]. Although popular in practice and based on straightforward heuristics, stepwise regression is known to be unstable, can lead to erroneous selection results, and its statistical properties are still not well understood [18]. The other variable selection methods requires comparing all possible subsets of factors or sub-models, and thus are computationally intensive and can quickly become infeasible when the total number of factors increases. When the number of observations is much smaller than the number of factors, which is referred to as the large p small n scenario, the classical variable selection methods become ineffective.

In the past decade, advances in sciences and technology has brought about an explosion of high dimensional data in a variety of areas, and many of these data sets fit into the large p and small n scenario. The demand for high dimensional data analysis tools motivated researchers especially statisticians to develop effective, stable and computationally feasible solutions. The Least Absolute Shrinkage and Selection Operator (lasso) proposed by Tibshirani [19] is the pioneer of a family of methods that use L_1 penalty to regularize regression and perform variable selection in high dimensional data analysis. The lasso become extremely popular after efficient computing algorithms have been developed [20] [21]. Motivated by the lasso, other L_1 penalty based methods have also been proposed, including the smoothly clipped absolute deviation (SCAD) method by Fan and Li [22], adaptive lasso [23], and the Dantzig Selector by Candes and Tao [24]. Some of these methods were used to analyze data generated from supersaturated experiments. Li and Lin [25] proposed to use SCAD to identify important factors and showed that it outperformed stepwise regression in simulation. Yuan *et al.* [26] proposed to use the lasso to select variables when analyzing designed experiments, and Phoa *et al.* [27] used the Dantzig selector to analyze supersaturated designs. These works have demonstrated the potential of using the lasso and other variants as variable selection methods for analyzing supersaturated experiments.

1.1.2.3 Disparity between the Two Steps

In a review paper on factor screening via supersaturated designs, Gilmour [28] pointed out that there exists a disparity between the construction of supersaturated designs and their analysis in the literature, that is, most existing optimality criteria for constructing supersaturated designs appear to be unrelated to the methods used for analyzing the generated data. For example, the minimum r and $E(s^2)$ criteria [11] are the two most popularly used and studied optimality criteria for supersaturated designs. Here r and $E(s^2)$ are the maximum and average correlation between any two factors in a supersaturated design, respectively. However, it is not clear how r and $E(s^2)$ affects the variable selection results in the analysis step when classical variable selection methods are used, and this issue has never been carefully investigated in the literature. One of the major causes for this disparity is that the impacts of the design matrix on the performance of most classical variable selection methods are difficult to understand and assessed.

The adoption of the lasso and other L_1 penalty based variable selection methods for supersaturated design can potentially resolve the disparity mentioned above. Unlike classical variable selection results, the lasso and other L_1 penalty based methods can be cast as penalized least squares problems and are continuous in nature. Therefore, it is possible to understand the role the design matrix plays in variable selection when the lasso and other L_1 penalty methods are used, and furthermore, it is also possible to construct a supersaturated design that is optimal in terms of delivering best variable selection performance. In order to materialize these two possibilities, the exact relationship between the variable selection performance of the lasso and the design matrix needs to be established, especially under a finite sample. (Design of experiment is usually a finite sample problem). Although there have been a large amount of research results on the variable selection performance of the lasso reported in the literature, they are mostly asymptotic results and not directly applicable for design of experiment. Fortunately, thanks to the penalized least squares setup, the properties of the lasso under finite samples can also be studied and established.

1.2 Supersaturated Designs (SSD)

For ease of discussion, we assume that all the factors have two levels with $-$ and $+$ denoting the low and high levels, respectively, in the rest of this dissertation. All the discussion can be extended to supersaturated designs involving factors with more than 2 levels. For a SSD with n runs and p factors with $n < p$, denoted by $d_{n,k}$, it can be represented by a $n \times p$ array of $-$'s and $+$'s with rows as runs and columns as the settings of the factors. For example, given below is a 6×10 supersaturated design with 10 factors and 6 runs:

$$\begin{array}{cccccccccc}
+ & + & - & + & + & + & - & - & - & + \\
+ & + & + & - & - & - & + & - & + & + \\
- & - & + & - & + & + & - & + & + & + \\
- & + & - & + & + & - & + & + & + & - \\
+ & - & + & + & - & + & + & + & - & - \\
- & - & - & - & - & - & - & - & - & -
\end{array}$$

In general, X_1, X_2, $\ldots$, and X_p are used to denote the factors as well as their corresponding columns in $d_{n,p}$. Let $s_{ij} = X_i' X_j$ for $1 \leq i, j \leq p$ where prime represents the transpose, and let $S = (s_{ij})$. If $s_{ij} = 0$, then the two columns X_i and X_j are orthogonal to each other. Because $n < p$, it is not possible to make all s_{ij}'s equal to zero, that is, orthogonal supersaturated designs do not exist. The nonzero s_{ij}'s reflect the departure of a supersaturated design from orthogonality. Based on them, Booth and Cox [11] proposed two criteria, the minimum r and minimum $E(s^2)$ criteria, for constructing supersaturated design. Let

$$r = \max_{1\leq i<j\leq p} s_{ij}, \text{ and } E(s^2) = \frac{1}{p(p-1)} \sum_{1\leq i<j\leq p} s_{ij}^2.$$

Both of these two criteria are based on correlations between the design columns or factors. A supersaturated design is said to be a minimum r supersaturated design if it attains the minimum r among all SSDs with same n and p. Similarly, a SSD is said to be a minimum $E(s^2)$ design if it attains the minimum $E(s^2)$ among all the SSDs with same n and p. As the measure of the nonorthogonality, the former one is the the maximum correlation between two factors and latter one is the expected squared correlation between any two factors.

To reflect the estimation efficiency and multi-factor orthogonality, Wu ([13]) proposed the D_f- and A_f- criteria for selecting optimal SSDs. For $k \geq 2$, let $s = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, p\}$ and $X_s = \{X_{i_1}, \ldots, X_{i_k}\}$. Define

$$D_k = \binom{p}{k}^{-1} \sum_{|s|=k} |n^{-1}X_s'X_s|^{1/k},$$

and

$$A_k = \binom{p}{k}^{-1} \sum_{|s|=k} k^{-1}\text{trace}(n^{-1}X_s'X_s)^{-1},$$

Where D_k and A_k are the average $D-$ and $A-$ optimality of the k-factor projections. The optimal SSDs could be achieved by maximizing the D_k criterion and Minimizing the A_k criterion [13].

Given an optimality criterion, there are two major approaches to constructing optimal SSDs. One approach is to develop systematic methods to construct optimal SSDs, and the other is to develop efficient search algorithms to search SSDs. In the literature, both of these two approaches have been used.

The construction methods based on the Hadamard matrices was first investigated by Lin and Wu ([29]), Tang and Wu ([30]) and Cheng ([31]); Then the construction methods based on balanced incompleted block designs have been discussed by Nguyen

([32]), Cheng ([31]), Bulutoglu and Cheng ([33]); The cyclic methods of construction have been explored also, for example, by Liu and Zhang ([34]), Eskridge et al. ([35]), Liu and Dean ([36]), and Georgiou ([37]). Theoretical study of SSDs has been focused on establishing the bounds of the optimality criteria. For given n and p, various lower bounds for $E(s^2)$ has been obtained, such as the Nguyen-Tang-Wu bound and the improved lower bound derived by Butler, Mead, Eskridge and Gilmour ([38]). For some recently obtained bounds, see Bulutoglu and Cheng ([33]) and Bulutoglu and Ryan ([39]).

The construction methods always require certain relationship between n and p, therefore they may fail for certain design problems. For example, the method to use a half fraction of a Hadamard matrix (HFHM) ([12]) can only construct SSD of size $(n, p) = (2t, 4t - 2), (t \geq 3)$; The $Es2$ optimal design provided by Tang and Wu [30] using Hadamard matrices require p=q(n-1) factors, q is integer; The optimal designs generated by BIBD or cyclic methods can makeup where some Hadamard matrix methods fails but still are restricted by some constrains of the values of n and p.

To generate the optimal SSD with any value of n and p, the computer algorithmic methods have been employed. For example, Johnson and Nachtsheim ([40]) propose the k-exchange method to iteratively change the row of design to subsequently achieve optimal designs for given criterion. Lin ([41]) develop the algorithm by randomly permuting the signs of the column candidates; Nguyen (1996) using the idea of Near-Orthogonal Arrays (NOA) to achieve the optimal designs by pairwise permuting the signs in columns; Wu and Li ([29]) improved the algorithm by using columnwise-pairwise (CP) exchange method.

However, the above searching algorithms may not be always reliable, particularly as the number of experimental runs n increases. For example, Cheng ([31]) showed that the design found by search algorithm and reported by Nguyen ([32]) for $n = 18$ and $m = 36$ is not in fact $Es2$ optimal designs. Moreover, these algorithms are all employing random permuting or exchanging, therefore they lead to a great amount of computation for large numbers of runs such as $n = 46$ or $n = 98$. These approach usually suffers from low efficiency due to the potential large number of low quality columns generated. Moreover, most construction or searching methods are based on $Es2$, as well as the theoretical study on establishing the bounds of the optimality criteria. For given n and p, various lower bounds for $E(s^2)$ has been obtained; for some recently obtained bounds, see [33] and [39].

As discussed in the introduction, there exists a disconnection between the optimality criteria for constructing SSDs and the ultimate purpose of using such designs for variable selection or factor screening.

For example, whether an optimal SSD in the sense of the minimum $E(s^2)$ criterion can lead to optimal results in factor screening is generally unclear.

The major cause for the disconnection is due to the lack of a unified and systematic method for analyzing data generated from supersaturated experiments. When classical variable selection methods such as AIC are used, the connection between the supersaturated design matrix and the final variable selection results is generally unclear. Therefore, optimal SSDs do not guarantee best variable selection performances. On the other hand, although the asymptotic behaviors of most classical methods are understood [42], they do not provide insights about how to best construct designs or supersaturated designs when collecting only finite samples.

Recently, as discussed in the introduction, the lasso and other L_1 penalty based methods become popular for parameter estimation and variable selection in high dimensional data analysis. Simulation studies and applications show that these methods are much more effective than classical methods [43]. Because these methods are framed to solve penalized least square or penalized likelihood problems, their properties can be carefully studied, and it is possible to relate the property of a selected designs to its performance in variable selection. The relationship can then be used to develop criteria for constructing optimal design for the purpose of variable selection.

1.3 Lasso

The least absolute shrinkage and selection operator (lasso) proposed by Tibshirani ([19]) is the forerunner of a family of methods that use L_1 penalty to regularize parameter estimation and perform variable selection. Some recent extensions of lasso include fused-lasso [44], group-lasso [45] and adaptive lasso [23]. We focus on the original lasso and its performance as a variable selection method in this article. Nevertheless, the concepts and methods we develop in this article can be extended to the aforementioned extensions of lasso.

Consider the linear regression model

$$Y = X\beta^0 + \epsilon, \tag{1.1}$$

where $Y = (Y_1, Y_2, \ldots, Y_n)'$ is an n dimensional vector of observed responses, X is an $n \times p$ design matrix, $\beta^0 = (\beta_1^0, \beta_2^0, \ldots, \beta_p^0)'$ is a p dimensional vector of true regression coefficients, and $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)'$ is the vector of experiment errors that are independent and identically distributed as $N(0, \sigma^2)$. Denote the p columns of X as X_1, X_2, ..., and X_p, respectively. The lasso estimate of β^0, denoted by $\hat{\beta^\lambda}$, is the solution of the following L_1-penalized least squares problem,

$$\hat{\beta^\lambda} = \arg \min_{\beta \in R^p} \{\|Y - X\beta\|_2^2 + \lambda \sum_{i=1}^{p} |\beta_i|\}, \tag{1.2}$$

where $\sum_{i=1}^{p} |\beta_i|$ is the L_1 norm penalty function of β, and λ is the tuning parameter that controls the amount of penalty.

The lasso estimate $\hat{\beta^\lambda}$ depends on λ, and the trajectory of $\hat{\beta^\lambda}$ as λ decreases from a certain large value to zero is referred to as the lasso solution path. Efron [21] showed that the lasso solution path is piecewise linear and proposed an algorithm based on Least Angle Regression (LARs) to efficiently compute the lasso solution path. The LARs algorithm gives lasso a computational advantage over other conventional variable selection methods. Rosset and Zhu ([46]) gave a concise characterization of the lasso solution path. Faster algorithms such as the coordinate decent algorithm [47] have been further developed for computing the lasso solution path. As λ decreases, the components of $\hat{\beta}^{\lambda}$ change from zero to nonzero or from nonzero to zero in a one by one fashion. At a fixed λ, we refer to the model that includes only the variables with nonzero coefficients as the model selected by lasso at λ. Therefore, lasso adds or removes variables from the selected model and generates a sequence of models as λ changes. To obtain the final model for use in practice, one needs to properly determine the magnitude of the tuning parameter λ. Zou and Tibshirani [48] investigated the application of cross validation (CV), Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) in determining λ. We refer to lasso combined with CV, AIC, or BIC as CV-lasso, AIC-lasso, or BIC-lasso, respectively, in this article.

The performance of lasso can be assessed by two different types of criteria, which are lasso's accuracy in estimation and lasso's accuracy in variable selection, respectively. In our research, we only focus on lasso's variable selection accuracy, particularly under finite samples, which is consistent to the purpose of the experiment designs.

1.4 The Contribution and Outline

Our goal in this dissertation is to develop a unified framework for design and analysis of screening experiments with application in large scale simulation models. In particular, we use supersaturated designs as the general design strategy for the experiment step of the screening process and further to use the lasso based variable selection methods as the general approach for the analysis step. In chapter 2, we studied the variable selection performances of the lasso under finite samples scenarios and derive explicit formulas for the probability that lasso selects all true variables at any fixed λ. We further develop a self-voting procedure to approximate the optimal λ and propose to use the resulting approximate value as the selected tuning parameter in practice. In chapter 3, we proposed optimality criteria for constructing supersaturated designs that guarantee optimal variable selection performance based on the results from chapter 2, we study the theoretical and practical properties of the proposed criteria, and further develop efficient algorithms to construct optimal supersaturated designs. We develop a software package capable of constructing optimal supersaturated designs and analyzing data generated from supersaturated experiments. One real example are discussed in chapter 4 to demonstrate how to apply the framework proposed to the real world problem and concluding remarks are made in chapter 6.

2. FINITE SAMPLE PROPERTIES OF LASSO

2.1 Probability of Sign-Correct Selection

In the linear model (1.1), X_1, X_2, $\ldots$, and X_p are used to denote the columns of X. For ease in discussion, X_1, X_2, $\ldots$, and X_p are also used to denote the independent variables that correspond to the columns of X. For $1 \leq i \leq p$, X_i is said to be a true variable if $\beta_i^0 \neq 0$; otherwise, X_i is said to be an untrue variable. Let $A = \{i : \beta_i^0 \neq 0\}$ and $C = \{i : \beta_i^0 = 0\}$ be the collections of the indexes of the true and untrue variables, respectively. For any fixed λ, define $\hat{A}^\lambda = \{j : \hat{\beta}_j^\lambda \neq 0\}$, which is the collection of the indexes of the variables selected by lasso at λ, and $\hat{C}^\lambda$ the complement of $\hat{A}^\lambda$. As mentioned in the Introduction, when λ is sufficiently large, all the entries of $\hat{\beta}^\lambda$ are equal to zero, and $\hat{A}^\lambda = \emptyset$; As λ decreases, variables will be added into or removed from $\hat{A}^\lambda$ step by step, and $\hat{\beta}^\lambda$ will change in a piece-wise linear fashion [21]

The magnitude of λ determines how many and which will be selected by lasso. For given λ, we say lasso achieves ordinary correctness (OC) in variable selection if $\hat{A}^\lambda = A$. Let $sgn(\cdot)$ be the sign function such that $sgn(x) = 1$, if $x > 0$; $= 0$, if $x = 0$; and $= -1$, if $x < 0$. We say lasso achieves sign correctness (SC) in variable selection if $sgn(\hat{\beta}^\lambda) = sgn(\beta^0)$, that is, lasso not only selects all true variables (i.e., $\hat{A}^\lambda = A$) but also estimates the coefficients of the true variables with correct signs. Note that sign correctness implies ordinary correctness, but the converse is not generally true. For given X, we define the probability that lasso achieves OC in variable selection at λ as $P_{OC}(\lambda) = P(\hat{A}^\lambda = A)$ and the probability that lasso achieves SC in variable selection at λ as $P_{SC}(\lambda) = P(sgn(\hat{\beta}^\lambda) = sgn(\beta^0))$. Clearly, $P_{OC}(\lambda) \geq P_{SC}(\lambda)$.

The two types of variable selection correctness defined above are related to lasso's variable selection consistency and sign consistency discussed in the literature of lasso;

however, a key difference exists. We discuss the difference between sign correctness and sign consistency as an example. Sign consistency is an asymptotic property that holds under the irrepresentable condition [49] and a properly selected sequence of λ as n goes to infinity, whereas sign correctness is a sampling property of lasso under a fixed sample of size n and a fixed value of λ. Therefore, we can discuss the probability that lasso achieves sign correctness in variable selection, whereas it does not make sense to discuss the probability that lasso is sign consistent. A connection between sign correctness and sign consistency of lasso does exist. The sign consistency of lasso can be stated as follows. Under the irresponsible condition, there exists a sequence of λ values, denoted as $\{\lambda_n\}$, such that lasso's sign correctness probability $P_{SC}(\lambda_n) \to 1$ as $n \to \infty$. In this article, we are interested in lasso's variable selection performance under finite samples, that is, we are interested in $P_{SC}(\lambda)$ under sample size n rather than the limit of $P_{SC}(\lambda_n)$ as $n \to \infty$.

Next, we obtain the explicit formulas for $P_{SC}(\lambda)$ and $P_{OC}(\lambda)$ under finite sample size n. Let X_A and β_A^0 be the restrictions of X and β^0 to the true variables in A, respectively. Let X_C and β_C^0 be the complements of X_A and β_A^0 in X and β^0, respectively. Define $P = X_A(X_A'X_A)^{-1}X_A'$, $R = X_C'X_A(X_A'X_A)^{-1}sgn(\beta_A^0)$, and $D = \frac{\lambda}{2}(X_A'X_A)^{-1}sgn(\beta_A^0)$. The following theorem presents the exact formula for the probability that lasso achieves SC in variable selection under given sample size n, design matrix X, and tuning parameter λ. Without loss of generality, we assume there are q true variables and further the q true variables are $X_1, X_2, ..., X_q$. In other words, we assume $A = \{1, 2, \ldots, q\}$ and $C = \{q+1, q+2, \ldots, p\}$.

Theorem 1. *Assume $U = (U_1, U_2, \ldots, U_{p-q})'$ follows the normal distribution $N(R, \frac{\sigma^2}{\lambda^2}X_C'(I-P)X_C)$, and $V = (V_1, V_2, \ldots, V_q)'$ follows the normal distribution $N(D, \sigma^2(X_A'X_A)^{-1})$. The Probability that lasso achieves sign correctness in variable selection at λ, denoted as $P_{SC}(\lambda)$, is*

$$P_{SC}(\lambda) = P(\mathcal{E}_1)P(\mathcal{E}_2), \tag{2.1}$$

where $\mathcal{E}_1$ is the event of

$$\{-1 \leq U \leq 1\}, \tag{2.2}$$

and $\mathcal{E}_2$ is the complement of

$$\bigcap_{i=1}^{q}\{\min(0, V_i) \leq \beta_i^0 \leq \max(0, V_i)\}. \tag{2.3}$$

The proof of Theorem 1 is given in the Appendix. Theorem 1 is in the same spirit as Proposition 1 in [49], but the two results are different. Theorem 1 gives the exact probability that lasso achieve SC in variable selection, whereas Proposition 1 in [49] provides a lower bound of the same probability. Because [49] focused on lasso's asymptotic behavior, the lower bound turns out to be sufficient. In this article, we instead consider lasso's finite sample behavior, therefore, it is critical to obtain the exact probability. The two events $\mathcal{E}_1$ and $\mathcal{E}_2$ in Theorem 1 represent the necessary and sufficient conditions for lasso to choose the true variables with correct signs. $\mathcal{E}_1$ focuses on the selection of the true variables, while $\mathcal{E}_2$ is to ensure that the signs of the chosen variables are correct.

With fixed A, β^0, λ and σ^2, the probabilities of $\mathcal{E}_1$ and $\mathcal{E}_2$ depend on the mean vectors and covariance matrices of U and V, which are R, $\frac{4\sigma^2}{\lambda^2}X_C'(I-P)X_C$, $\frac{\lambda}{2}D$, and $\sigma^2(X_A'X_A)^{-1}$, respectively. Using $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$, under the irrespresentable condition $|R| < 1$, it can be shown that lasso is asymptotically sign consistent, that is, $P_{SC}(\lambda) = P(\mathcal{E}_1)P(\mathcal{E}_2)$ goes to 1 as n goes to infinity and λ is chosen such that $\lambda/n \to 0$ and $\lambda/n^{(\frac{1+c}{2})} \to \infty$ with $0 < c < 1$. Therefore, asymptotically, $P_{SC}(\lambda)$ is determined by the magnitude of R. This however is not true when the sample size n is finite.

When n is finite, $P_{SC}(\lambda)$ depends on all four terms listed in the previous paragraph through the two independent normal distributions U and V given in Theorem 1. We first exam the dependence of $P_{SC}(\lambda)$ on β^0. Notice that $\mathcal{E}_1$ only depends on the sign but not the magnitude of β_A^0, while $\mathcal{E}_2$ depends on both the sign and magnitude of β_A^0. With fixed sign, when β^0 increases in magnitude, that is, the strength of signal increases, $P(\mathcal{E}_1)$ does not change while $P(\mathcal{E}_2)$ increases, which makes $P_{SC}(\lambda)$

increase. Second, we exam the dependence of $P_{SC}(\lambda)$ on λ. Notice that λ appears in the variance of U as well as in the mean of V. As λ decreases to zero, $P(\mathcal{E}_1)$ decreases to zero while $P(\mathcal{E}_2)$ increases to 1; and as λ increases to infinity, $P(\mathcal{E}_1)$ increases to 1 while $P(\mathcal{E}_2)$ decreases to zero. Therefore, a global maximum of $P_{SC}(\lambda)$ exists. We refer to the value of λ that attains the maximum of $P_{SC}(\lambda)$ as the optimal tuning parameter and denote it as λ_{opt}, that is,

$$\lambda_{\text{opt}} = \text{argmax}_{\lambda} P_{SC}(\lambda) = \text{argmax}_{\lambda} P(\mathcal{E}_1)P(\mathcal{E}_2). \tag{2.4}$$

At $\lambda = \lambda_{\text{opt}}$, lasso achieves the largest probability $P_{SC}(\lambda_{opt})$ that it selects all true variables with correct signs. When X and β^0 are known, λ_{opt} can be calculated at least numerically by solving the optimization problem (2.4). In practice, however, β^0 is unknown and needs to be estimated from data. In Section 2.3, using (2.4), we will propose an iterative procedure to approximate λ_{opt}, and the result will be further used for lasso to select the true variables.

As noted before, sign correctness in variable selection implies ordinary correctness, because ordinary correctness only requires $\hat{A} = A$ but not $sgn(\hat{A}) = sgn(\beta^0_A)$. In other words, ordinary correctness only requires that lasso select all of the true variables (i.e. $\hat{A} = A$), but the signs of $\hat{\beta}^{\lambda}_i$ for $i \in \hat{A}$ can be arbitrary. Therefore, $sgn(\hat{\beta}^{\lambda}_A)$ can be any q-dimensional vector of +1's and -1's. The total number of such vectors is 2^q, which we denote as $\delta^{(1)}, \delta^{(2)}, ..., \delta^{(2^q)}$, respectively. Without loss of generality, we assume $\delta^{(1)} = sgn(\beta^0_A)$. For $1 \leq k \leq 2^q$, define $R^{(k)} = X'_C X_A (X'_A X_A)^{-1} \delta^{(k)}$ and $D^{(k)} = (X'_A X_A)^{-1} \delta^{(k)}$. Let $U^{(k)} = (U^{(k)}_1, U^{(k)}_2, \ldots, U^{(k)}_q)'$ and $V^{(k)} = (V^{(k)}_1, V^{(k)}_2, \ldots, V^{(k)}_q)'$; and we assume $U^{(k)}$ and $V^{(k)}$ follow the normal distributions $\text{N}(R^{(k)}, \frac{4\sigma^2}{\lambda^2} X'_C(I-P)X_C)$ and $\text{N}(D^{(k)}, \sigma^2 (X'_A X_A)^{-1})$, respectively. Assume further that $U^{(k)}$'s and $V^{(k)}$'s are independent for $1 \leq k \leq 2^q$. The probability that lasso selects all of the true variables with or without correct signs, denoted by $P_{OC}(\lambda)$, is given in the following Theorem.

Theorem 2. *The probability that the lasso achieves ordinary correctness in variable selection is*

$$P_{OC}(\lambda) = P_{SC}(\lambda) + \sum_{k=2}^{2^q} P(\mathcal{E}^{(k)}_1)P(\mathcal{E}^{(k)}_2), \tag{2.5}$$

where $\mathcal{E}_1^{(k)}$ *is the event*

$$\left\{-1 \leq U^{(k)} \leq 1\right\}, \tag{2.6}$$

$\mathcal{E}_2^{(k)}$ *is the event*

$$\{(\bigcap_{i:\delta_i^{(k)}=-\delta_i^{(0)}} \mathcal{E}_{V_i}^{(k)})\bigcap(\bigcap_{i:\delta_i^{(k)}=\delta_i^{(0)}} \mathcal{E}_{V_i}^{(k)c})\}, \tag{2.7}$$

$\mathcal{E}_{V_i}^{(k)}$ *is the event* $\{min(0, V_i) \leq \hat{\beta}_i^{(k)} \leq max(0, V_i)\}$*, and* $\mathcal{E}_{V_i}^{(k)c}$ *is the complement of* $\mathcal{E}_{V_i}^{(k)}$ *for* $k = 1, 2, ..., 2^q$ *and* $i = 1, 2, ..., q$*.*

The proof of Theorem 2 is given in the Appendix. Obviously, $P_{OC}(\lambda) \geq P_{SC}(\lambda)$. It can also be shown that the difference of $P_{OC}(\lambda)$ and $P_{SC}(\lambda)$ will decrease to 0 when n increases to infinity.

2.2 Evaluation of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ in high dimensions

The direct evaluation and optimization of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ with respect to β, λ or X involves high dimensional integration, which can become computationally infeasible as the dimension p increases. In this section, we propose two different approaches to approximating $P(\mathcal{E}_1)$and $P(\mathcal{E}_2)$ in high dimensional cases, which use lower bounds and Monte Carlo simulation, respectively.

2.2.1 Lower bounds of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$

One approach to alleviating the computational burden in evaluating $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ is to consider their lower bounds that do not involve high dimensional integration. For the purpose of selecting the optimal tuning parameter λ_{opt}, these bounds sometimes can be sufficient. In what follows, we report such bounds we have obtained.

Recall the two probabilities $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ are defined by the random vectors U and V, respectively, with U and V following $\text{N}(R, \frac{4\sigma^2}{\lambda^2}X_C'(I-P)X_C)$ and

$N(\frac{\lambda}{2}D, \sigma^2(X_A'X_A)^{-1})$. Let $R = (R_1, R_2, \ldots, R_{p-q})'$, and $(\sigma^2_{U_1}, \sigma^2_{U_2}, \ldots, \sigma^2_{U_{p-q}})'$ the vector of the diagonal elements of $4\sigma^2 X_C'(I-P)X_C$. Let $D = (D_1, D_2, \ldots, D_q)'$, and $(\sigma^2_{V_1}, \sigma^2_{V_2}, \ldots, \sigma^2_{V_q})'$ the vector of the diagonal elements of $\sigma^2(X_A'X_A)^{-1}$. Let $\mathbf{I}(\cdot)$ be the indicator function such that $\mathbf{I}(\mathcal{E}) = 1$ if $\mathcal{E}$ is true; and $= 0$, otherwise. Let $\Phi(\dot{)}$ be the cumulative distribution function of the standard normal distribution. The next proposition provides lower bounds for $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$, respectively.

Proposition 1. A lower bound for $P(\mathcal{E}_1)$, denoted as $P(\mathcal{E}_1)_{low}$, is

$$P(\mathcal{E}_1)_{low} = \max\left(0, 1 - \frac{1}{\lambda\sqrt{2\pi}} \sum_{j=1}^{p-q} \sigma_{U_j} \left(\mathbf{I}(|R_j| < 1)Q_1(j) + \mathbf{I}(|R_j| > 1)Q_2(j) + \mathbf{I}(|R_j| = 1)Q_3\right)\right) \tag{2.8}$$

where

$$Q_1(j) = [\frac{1}{(1+R_j)}\exp\{\frac{-\lambda^2(1+R_j)^2}{2\sigma^2_{U_j}}\} + \frac{1}{(1-R_j)}\exp\{\frac{-\lambda^2(1-R_j)^2}{2\sigma^2_{U_j}}\}],$$

and

$$Q_2(j) = [\frac{1}{\sqrt{2\pi}(1+R_j)}\exp\{\frac{-\lambda^2(1+R_j)^2}{2\sigma^2_{U_j}}\} + \frac{1}{\lambda} - \frac{(R_j - 1)}{\sqrt{2\pi}((1-R_j)^2 + \frac{\sigma^2_{U_j}}{\lambda^2})}\exp\{\frac{-\lambda^2(1-R_j)^2}{2\sigma^2_{U_j}}\}],$$

and

$$Q_3 = 0.5 + \Phi(\frac{-2 - R_j}{\sigma_{U_j}}).$$

And a lower bound for $P(\mathcal{E}_2)$, denoted as $P(\mathcal{E}_2)_{low}$, is

$$P(\mathcal{E}_2)_{low} = \max\left(0, 1 - \sqrt{\frac{2}{\pi}} \sum_{i=1}^{q} \sigma_{V_i}[\frac{\beta_i^0}{|\beta_i^0|}\frac{1}{(2\beta_i^0 - \lambda D_i)}\exp\{\frac{-(2\beta_i^0 - \lambda D_i)^2}{8\sigma^2_{V_i}}\}]\right). \tag{2.9}$$

Notice that the expressions of the lower bounds above do not involve high dimensional integration. The product of these two lower bounds leads to a lower bound for $P_{SC}(\lambda)$, which is $P_{SC}(\lambda)_{low} = P(\mathcal{E}_1)_{low}P(\mathcal{E}_2)_{low}$. The evaluation and optimization of $P_{SC}(\lambda)_{low}$ are much easier than the evaluation and optimization of the original probability $P_{SC}(\lambda)$. When the lower bounds are sufficiently tight, the maximizers of $P_{SC}(\lambda)_{low}$ and $P_{SC}(\lambda)$ with respect to λ are expected to be close to each other. For given β^0, X and σ^2, the maximizer of $P_{SC}(\lambda)_{low}$ with respect to λ, denoted as λ_{lopt}, is defined as follows.

$$\lambda_{lopt} = \arg\max_{\lambda} P(\mathcal{E}_1)_{low} P(\mathcal{E}_2)_{low} \tag{2.10}$$

Next, we use an example to demonstrate the various probabilities and their lower bounds obtained above. The example shows that λ_{lopt} and λ_{opt} can be close to each other.

Example 1

Consider the linear model $Y = X\beta + \epsilon$, where $\beta = (a, 0, a, 0, 0, 0, 0, 0, a, 0, 0, 0, 0, 0, 0, 0)'$, X is the 12×16 supersaturated design given in Table 4 of [29], and ϵ follows $N(0, \sigma^2 I_{16})$. The true variables in this model are X_1, X_3, and X_9. The values of $P(\mathcal{E}_1)$, $P(\mathcal{E}_2)$, $P(\mathcal{E}_1)_{low}$ and $P(\mathcal{E}_2)_{low}$ for any $\lambda > 0$ under the model can be calculated using Theorem 1 and Proposition 1.

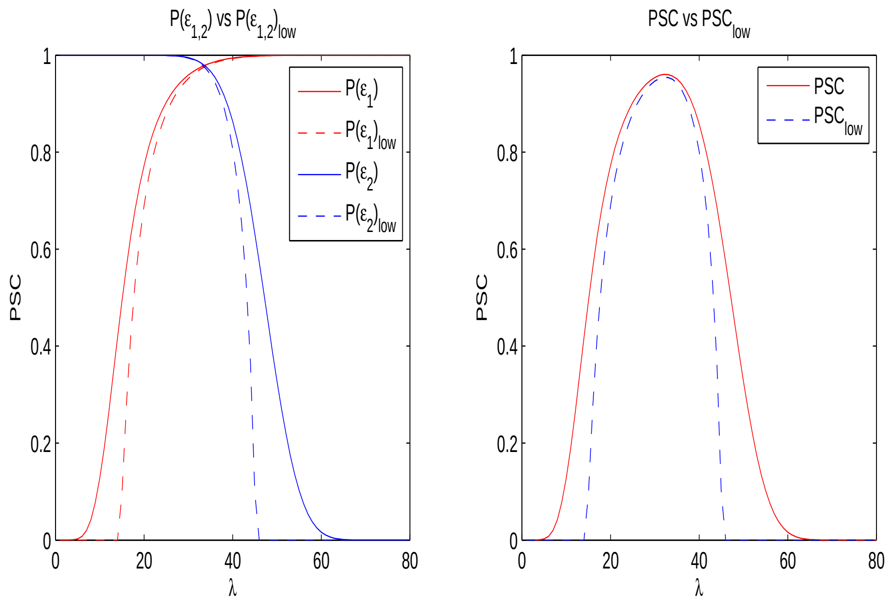


Figure 2.1.: The plots of $P(\mathcal{E}_1)$, $P(\mathcal{E}_1)_{low}$, $P(\mathcal{E}_2)$, $P(\mathcal{E}_2)_{low}$, $P_{SC}(\lambda)$ and $P_{SC}(\lambda)_{low}$ against λ with $a = 2$

For $a = 2$ and $\sigma^2 = 1$, the plots of $P(\mathcal{E}_1)$, $P(\mathcal{E}_1)_{low}$, $P(\mathcal{E}_2)$ and $P(\mathcal{E}_2)_{low}$ against λ are presented in the left panel of Figure 2.1, and the plots of $P_{SC}(\lambda)$ and $P_{SC}(\lambda)_{low}$ against λ are presented in the right panel of Figure 2.1. From Figure 2.1, the curves of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ versus λ demonstrate the expected patterns discussed in Section 2.1. As λ increases from 0 to 80, $P(\mathcal{E}_1)$ increases from 0 to 1 while $P(\mathcal{E}_2)$ decreases from 1 to 0. The probability of $P_{SC}(\lambda)$, which is the product of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$, increases in $0 \leq \lambda \leq 30.287$ and then decreases in $30.287 \leq \lambda \leq 80$. The maximum of $P_{SC}(\lambda)$ is attained at $\lambda = 30.287$, that is, $\lambda_{\text{opt}} = 30.287$ and $P_{SC}(\lambda_{opt}) = 96.1\%$.

As expected, the curves of $P(\mathcal{E}_1)_{low}$ and $P(\mathcal{E}_2)_{low}$ versus λ are always below the curves of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ versus λ, respectively. Similarly, the curve of $P_{SC}(\lambda)_{low}$ versus λ is below that of $P_{SC}(\lambda)$ versus λ. In this example, although $P_{SC}(\lambda)_{low}$ is not a tight lower bound for $P_{SC}(\lambda)$, the former captures the overall pattern of the latter. Furthermore, the maximizer of $P_{SC}(\lambda)_{low}$ is $\lambda_{lopt} = 30.312$ and the attained maximum probability is $P_{SC}(\lambda_{lopt})_{low} = 96.2\%$, which are close to those of $P_{SC}(\lambda)$, respectively.

2.2.2 Monte Carlo Approximation

The lower bounds approach in the previous section is computationally more convenient than the original formulas of $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ and are reasonably tight when the number of variables p is small or moderate. When p is large, however, numerical results show that the lower bounds become less accurate. In this article, we propose to use the Monte Carlo method to approximate $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$. We randomly draw m observations from U and m observations from V. Let m_1 be the number of times that $\mathcal{E}_1$ is true, and m_2 the number of times that $\mathcal{E}_2$ is true. Then, $P(\mathcal{E}_1)$ and $P(\mathcal{E}_2)$ are approximated by m_1/m and m_2/m, respectively, and further denoted by $P(\mathcal{E}_1)_{Monte}$ and $P(\mathcal{E}_2)_{Monte}$

Similar to (12), we can then define the optimal λ based on the Monte Carlo method, which is denoted as λ_{mopt} as follows,

$$\lambda_{mopt} = \arg\max_{\lambda} P(\mathcal{E}_1)_{Monte} P(\mathcal{E}_2)_{Monte} \quad . \tag{2.11}$$

In summary, when β^0 is known, there are three approaches to calculating or approximating $P_{SC}(\lambda)$ and λ_{opt}, which are the direct method provided by Theorem 1, the lower bounds method, and the Monte Carlo method. In practice, however, β^0 is unknown, and it is impossible to directly calculate or approximate λ_{opt}. Nevertheless, the relationship between λ_{opt} and β^0 can be utilized to approximate λ_{opt} in an iterative manner as will be shown in the next section.

2.3 Self-Voting Procedure for determining λ_{opt}

In this section, we propose an iterative procedure to approximate λ_{opt} when β^0 is unknown. Our procedure is different from existing methods for selecting λ for lasso in the literature, such as CV-Lasso, AIC-Lasso and BIC-Lasso. Most existing methods select λ based on lasso's performance in prediction, whereas our procedure selects λ based on lasso's performance in variable selection, which is lasso's probability of selecting all of the true variables.

Our procedure consists of four steps. Firstly, we start with an initial value of λ, denoted as λ^0, and calculate the lasso estimate $\hat{\beta}^{\lambda^0}$. Let $\hat{A}^{\lambda^0} = \{i : \hat{\beta}_i^{\lambda^0} \neq 0\}$. In other words, $\hat{A}^{\lambda^0}$ is the collection of variables selected by lasso with $\lambda = \lambda^0$. Secondly, we generate the ordinary least squares (OLS) estimate of β by regressing the response against the variables in $\hat{A}^{\lambda^0}$ only, and denote the resulting estimate as $\hat{\beta}_{\hat{A}^{\lambda^0}}$. Thirdly, we treat $\hat{A}^{\lambda^0}$ as the collection of the true variables and $\hat{\beta}_{\hat{A}^{\lambda^0}}$ their true coefficients and use $P_{SC}(\lambda)$, $P_{SC}(\lambda)_{low}$ or $P_{SC}(\lambda)_{Monte}$ to calculate or approximate λ_{opt}; we denote the result as λ^*. Fourthly, we check if λ^* is equal to λ^0. If λ^* and λ^0 are equal, we stop the procedure and output λ^* as the final estimate of λ_{opt}; otherwise, replace λ^0 by λ^* and go back to the first step.

We call the procedure proposed above as the Self-Voting (SV) procedure for selecting the tuning parameter λ for lasso. The heuristics underlying the SV procedure come from the self-voting principle used for tuning parameter selection in smoothing splines regression [50]. Given λ^0, the proposed procedure can be considered a voting process for λ^0 to choose the best tuning parameter, that is, the resulting update λ^* is the optimal tuning parameter chosen by λ^0. The self-voting principle states that an optimal tuning parameter must vote for or choose itself. This principle can be validated using an asymptotic argument.

Recall that $Y = X_A\beta_A^0 + \epsilon$, where A is the collection of true variables. If we start the SV procedure from λ_{opt} (i.e., $\lambda^0 = \lambda_{opt}$), then lasso has high probability to select the true model, that is, $\hat{A} = A$ with high probability. The OLS estimate of $\beta_{\hat{A}}$ at λ_{opt}, denoted as $\hat{\beta}_{\hat{A}}^{ols}$, is $\hat{\beta}_{\hat{A}}^{ols} = \beta_A^0 + (X_A'X_A)^{-1}X_A\epsilon$. As the sample size n increases, $\hat{\beta}_{\hat{A}}^{ols}$ converges to β_A^0 at the rate of root n. Therefore, when n is large, $\hat{\beta}_{\hat{A}}^{ols}$ and β_A^0 are close to each other, which implies that the optimal tuning parameter under $\hat{\beta}_{\hat{A}}^{ols}$ (i.e. λ^*) and the optimal tuning parameter under β_A^0 (i.e. λ_{opt}) should be close to each other. Asymptotically, λ^* should converge to λ_{opt}, that is, λ_{opt} should be self-voting. Furthermore, when the SV procedure starts with an initial value in a neighborhood of λ_{opt}, the subsequent updates are expected to get closer to λ_{opt}. These heuristics have been confirmed in our intensive simulation study.

Example 2

We use an example to demonstrate the SV procedure. Consider the first case ($a = 2$) of the linear model discussed in Example 1, in which the true variables are x_1, x_3 and x_9. The optimal tuning parameter λ_{opt} is equal to 32, which was obtained with the knowledge of the design, true variables and their coefficients. In this simulation study, we pretend that the true model is unknown, and generate a sample of observations from the model.

In the first step, we set $\lambda^0 = 50$. With $\lambda^0 = 50$, the lasso estimates are $\hat{\beta}_1^{\lambda^0} = 0.0924$, $\hat{\beta}_9^{\lambda^0} = 0.4644$, and $\hat{\beta}_i^{\lambda^0} = 0$ for $i \neq 1, 9$. Hence, $\hat{A}^{\lambda^0} = \{1, 9\}$ as shown in

Figure 2.2. In the second step, we regress the response against x_1 and x_9 and obtain the OLS estimates $\hat{\beta}_1^{ols,\lambda^0} = 1.6549$ and $\hat{\beta}_9^{ols,\lambda^0} = 2.0269$. In the third step, we treat $\hat{A}^{\lambda^0} = \{1, 9\}$ as the collection of all true variables and $\hat{\beta}_1^{ols,\lambda^0}$ and $\hat{\beta}_2^{ols,\lambda^0}$ the true coefficients, and calculate the optimal tuning parameter that maximizes $P_{SC}(\lambda)$; and the result is $\lambda^* = 29.424$. In the fourth step, because $\lambda^0(= 50)$ and $\lambda^*(= 29.424)$ are different, we update λ^0 by setting $\lambda^0 = \lambda^*$, return to the first step, and start the next or second round of iteration.

In the first step of the second round of iteration, $\lambda^0 = 29.424$, the lasso estimates are $\hat{\beta}_1^{\lambda^0} = 0.754$, $\hat{\beta}_3^{\lambda^0} = 0.812$, $\hat{\beta}_9^{\lambda^0} = 1.221$, and $\hat{\beta}_i^{\lambda^0} = 0$ for $i \neq 1, 3, 9$, and hence $\hat{A}^{\lambda^0} = \{1, 3, 9\}$. In the second step, we regress the response against x_1, x_3 and x_9, and the resulting OLS estimates are $\hat{\beta}_1^{ols,\lambda^0} = 0.82$, $\hat{\beta}_3^{ols,\lambda^0} = 0.93$, and $\hat{\beta}_9^{ols,\lambda^0} = 1.52$. In the third step, we treat $\hat{A}^{\lambda^0} = \{1, 3, 9\}$ as the collection of all true variables and the current OLS estimates as the true coefficients, and calculate the optimal tuning parameter that maximizes $P_{SC}(\lambda)$, which is $\lambda^* = 30.287$. In the fourth step, because λ^* and λ^0 are different, we update λ^0 by setting $\lambda^0 = \lambda^* = 30.287$, return to the first step, and start the third round of iteration.

In the first step of the third round of iteration, $\lambda^0 = 30.287$, the lasso estimates are $\hat{\beta}_1^{\lambda^0} = 0.7263$, $\hat{\beta}_3^{\lambda^0} = 0.7757$, $\hat{\beta}_9^{\lambda^0} = 1.1893$, and $\hat{\beta}_i^{\lambda^0} = 0$ for $i \neq 1, 3, 9$, and hence $\hat{A}^{\lambda^0} = \{1, 3, 9\}$. In the second step, we regress the response against x_1, x_3 and x_9, and the resulting OLS estimates are $\hat{\beta}_1^{ols,\lambda^0} = 0.82$, $\hat{\beta}_3^{ols,\lambda^0} = 0.93$, and $\hat{\beta}_9^{ols,\lambda^0} = 1.52$. In the third step, we treat $\hat{A}^{\lambda^0} = \{1, 3, 9\}$ as the collection of all true variables and the current OLS estimates as the true coefficients, and calculate the optimal tuning parameter that maximizes $P_{SC}(\lambda)$, which is $\lambda^* = 30.287$. In the fourth step, because λ^* and λ^0 are the same, we stop the procedure and output λ^* as the estimate of λ_{opt}.

In this example, it takes three rounds of iteration for the SV procedure to stop, and the final tuning parameter is $\lambda^* = 30.287$, which is self-voting. Although the selected tuning parameter is different from the optimal tuning parameter $\lambda_{opt} = 32.371$, which is calculated under the knowledge of β^0, both lead to the selection of the true model. Note that the last round of iteration could stop after its first step, because in the first step, the lasso estimates lead to the same model (i.e. $\{1, 3, 9\}$) as the previous round, the remaining steps must be the same as their counterparts in the previous round, which implies that λ^0 must also be self-voting.

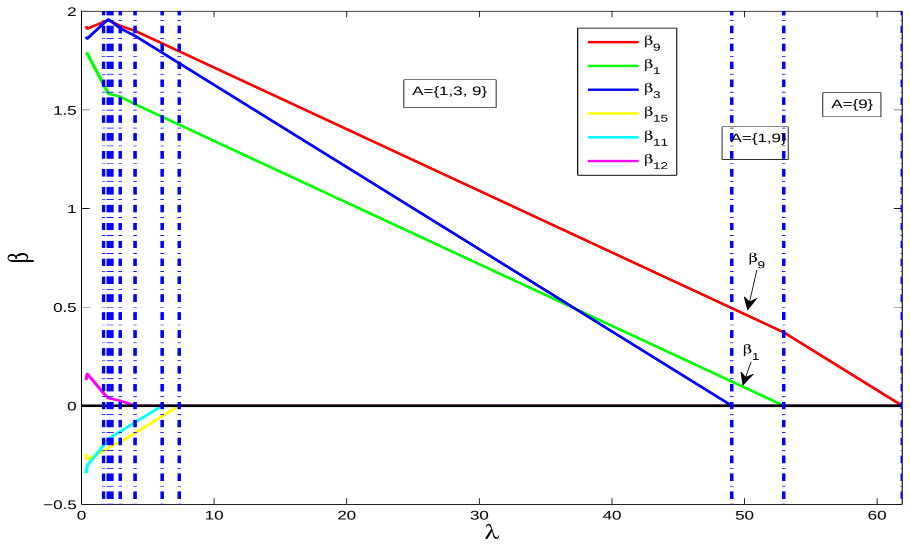


Figure 2.2.: The lasso solution path for Example 2

Each iteration round of the SV procedure can be considered a mapping from a given λ^0 to λ^*, which can be denoted as $\lambda^* = g(\lambda^0)$. It turns out that the mapping g can be easily obtained due to the fact that the selected model $\hat{A}^\lambda$ only changes over the 'transition points' of λ. A value of λ is said to be a transition point if a variable must be removed from or added to $\hat{A}^\lambda$ when λ passes this value. Suppose the transition points of λ are $\lambda_1 < \lambda_2 < ... < \lambda_m$. For any fixed $i \in \{1, 2, ..., m-1\}$ and $\lambda \in (\lambda_i, \lambda_{i+1}]$, $\hat{A}^\lambda$ remains the same, the OLS regression of Y against $\hat{A}^\lambda$ gives the same estimate $\hat{\beta}^{ols,\lambda}_{\hat{A}^\lambda}$, and hence $g(\lambda)$ also remains the same for $\lambda_i < \lambda \leq \lambda_{i+1}$. Therefore, $g(\lambda)$ is a step function of λ with jumps only at the transition points. In order to generate $g(\lambda)$ for all λ, we only need to select one point from each interval of λ. For example, for any fixed $i \in \{1, 2, ..., m-1\}$, we can select the middle point from $(\lambda_i, \lambda_{i+1}]$, calculate $g(\frac{\lambda_i+\lambda_{i+1}}{2})$, and then set $g(\lambda) = g(\frac{\lambda_i+\lambda_{i+1}}{2})$ for $\lambda_i < \lambda \leq \lambda_{i+1}$.

After $g(\lambda)$ is calculated, we can plot $g(\lambda)$ against λ. The self-voting values of λ are those that satisfy $g(\lambda) = \lambda$. To identify these self-voting values, we can add the line $y = \lambda$ and the intersections of the two curves $y = g(\lambda)$ and $y = \lambda$ are self-voting. We refer to the plot that shows those two curves and their intersections as the self-voting plot, or in short the SV-plot. As a matter of fact, the self-voting procedure can be performed graphically in this plot. We use the next example to demonstrate the self-voting procedure using the SV-plot.

Example 3

We calculate $g(\lambda)$ and generate the SV-plot for Example 2. The transition points of λ are 3.4, 4.5, 7, 8, 49.3, 53, and 62. The function $g(\lambda)$ is given as follows: $g(\lambda) = 37$ for $\lambda \in (53, 62)$; $= 29.4$ for $\lambda \in (49.3, 53)$; $= 30.3$ for $\lambda \in (8, 49.3)$; $= 14$ for $\lambda \in (7, 8)$; $= 11$ for $\lambda \in (4.5, 7)$; and $= 9.5$ for $\lambda \in (3.4, 4.5)$. The SV-plot is presented in Figure 2.3. Notice that $y = g(\lambda)$ is a step function represented by solid lines in the plot, and the function $y = \lambda$ is represented by the dash line.

The two functions $y = g(\lambda)$ and $y = \lambda$ have one intersection in this plot, which is $(30.287, 30.287)$. The SV-plot also shows the steps of a SV procedure, which starts from the middle point of the interval (53, 62) (i.e. $\lambda^{0,1} = 57.5$) and eventually converges to the intersection $(30.287, 30.287)$ (i.e. $\lambda^{*,3} = 57.5$). As shown in Figure 2.3, $\lambda^{0,1}(= 57.5)$ is the starting point, the corresponding $g(\lambda^{0,1}) = 37$, that is, $\lambda^{*,1}$ is equal to 37; then update the starting λ by $\lambda^{*,1}$ (i.e. $\lambda^{0,2} = 37$), and find the corresponding $g(\lambda^{0,2}) = 30.287(\lambda^{*,2} = 30.287)$ for current $\lambda^{0,2}$; keep updating the starting λ by $\lambda^{*,2}$ (i.e. $\lambda^{0,3} = 30.287$) and it will converge to the intersection $(30.287, 30.287)$ finally.

From Figure 2.3, we can find that $g(\lambda) < \lambda, (for \lambda > 30.287)$ and $g(\lambda) > \lambda, (for \lambda < 30.287)$. Then, if $\lambda^0 > 30.287$, when we apply the self-voting procedure, each step will result a smaller λ^* ($\lambda^* < \lambda^0$), which means λ^* is closer and closer to λ_{opt}; if $\lambda^0 < 30.287$, the applying of the self-voting procedure will result a larger and larger λ^* ($\lambda^* > \lambda^0$), which meas λ^* is also closer and closer to λ_{opt}. This implies that the

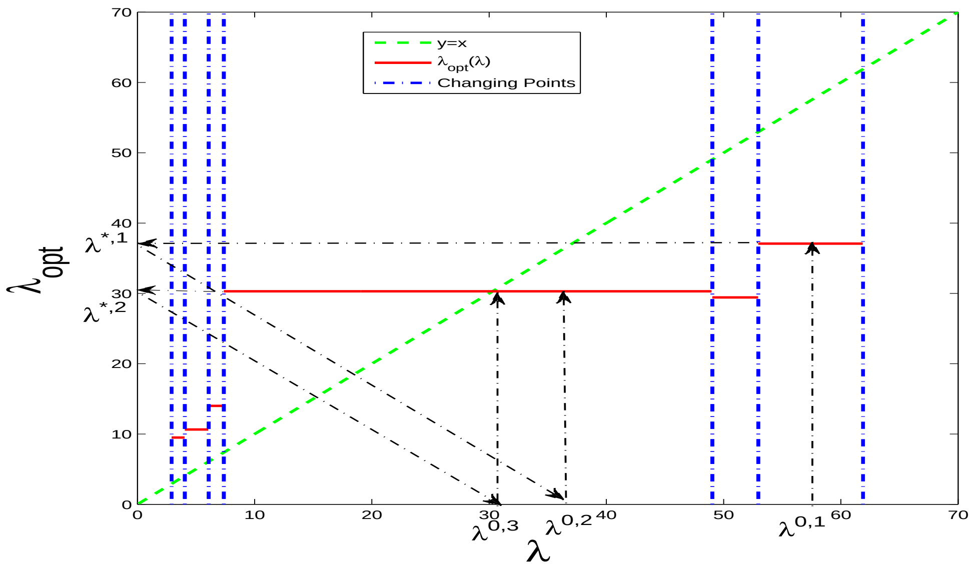


Figure 2.3.: $y = g(\lambda)$ and $y = \lambda$ Single Intersection

self-voting procedure tends to self-correct by iteration, as discussed before. It satisfy the self-voting principle and the intensive simulation study made by us also support it. This is in fact true in general, when there exists an unique intersection of the two curves $y = g(\lambda)$ and $y = \lambda$, the self-voting procedure starting from any initial point will converge and it must converge to this intersection.

The scenario discussed in Example 3 has the unique intersection. Sometime, it can have multiple intersections and none intersection. We discuss these two scenarios separately in the following two examples. We will demonstrate the two scenarios using examples and follow each of the example we give the rule of thumb to deal with the determine of the λ_{opt}.

Example 4:

We still use the same linear model in Example 1 with $a = 1$. In this model, the true variables are still x_1, x_3 and x_9. One sample generated from the model leads to the SV-plot with three intersections, as shown in Figure 2.4. The three intersections are at $\lambda = 12.58$, $\lambda = 14.65$ and $\lambda = 17.59$ respectively, denoted as λ_1^{in}, λ_2^{in} and λ_3^{in} separately. In this case, due to the presence of the three intersections with different starting values, the self-voting procedure will converge to different values. We can find that if the initial point $\lambda^0 \leq 12.58$, the self-voting procedure will converge to 12.58; if the initial point $12.58 \leq \lambda^0 < 14.12$, the self-voting procedure will converge to 12.58 also; if the initial point $14.12 < \lambda^0 \leq 14.65$, the self-voting procedure will converge to 14.65; if the initial point $14.65 \leq \lambda^0 < 14.90$, the self-voting procedure will converge to 14.65 also; if the initial point $14.90 < \lambda^0 \leq 17.59$, the self-voting procedure will converge to 17.59; if the initial point $17.59 \leq \lambda^0$, the self-voting procedure will converge to 17.59; As discussed previously, self-voting is necessary condition for choosing the optimal λ. Therefore to distinguish these three self-voting values, additional criteria must be used.

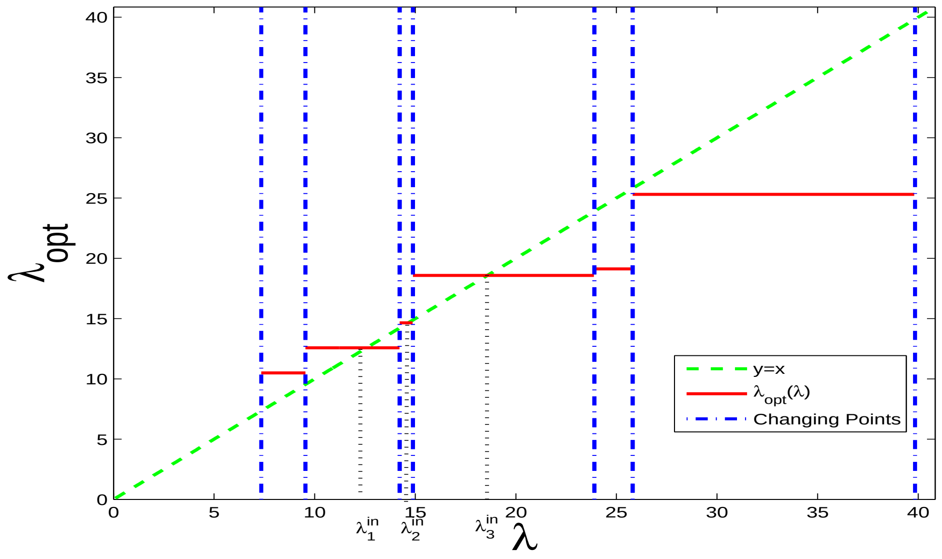


Figure 2.4.: $y = g(\lambda)$ and $y = \lambda$ Multiple Intersections

Example 4 shows the SV-plot with 3 intersections. In general, the SV-plot may have multiple k intersections sometimes. Suppose these intersections are at λ_1^{in}, λ_2^{in}, ..., λ_k^{in}, where $\lambda_1^{in} < \lambda_2^{in} <, ..., < \lambda_k^{in}$. Then starting from different initial values, the self-voting procedure will converge to different intersections. When $\lambda_1^{in} < \lambda^0 < \lambda_k^{in}$, suppose the two adjacent intersections for λ^0 are λ_i^{in} and λ_{i+1}^{in}, ($i \in [1, k-1]$), then it can be proved that if $g(\lambda^0) > \lambda^0$, the self-voting procedure will converge to λ_{i+1}^{in}; if $g(\lambda^0) < \lambda^0$, the self-voting procedure will converge to λ_i^{in}; if $g(\lambda^0) = \lambda^0$, the self-voting procedure will converge to λ^0 itself. Similarly, when $\lambda^0 > \lambda_1^{in}$ or $\lambda^0 < \lambda_k^{in}$, it can be proved that the self-voting procedure will always shrinkage to the nearest intersection.

Noticed that for each of the intersection, there is a corresponding $P_{SC}(\lambda)$ in the self-voting procedure. Intuitively, P_{SC} can be considered as the self-confidence of the current λ voting for itself. Therefore, we only need to compare the values of $P_{SC}(\lambda)$s and choose the λ_{opt} with largest corresponding $P_{SC}(\lambda_i^{in}), 1 \leq i \leq k$. That is,

Selection Rule for Multiple Intersections Scenario:

$$\lambda_{opt} = \arg \max_{\lambda_i^{in}: 1 \leq i \leq k} P_{SC}(\lambda_i^{in}). \tag{2.12}$$

Example 4 continued:

In the Example 4, the self-voting procedure starting from any point could be shrinkage the three intersections $(12.58, 12.58)$, $(14.65, 14.65)$ and $(17.59, 17.59)$. And the corresponding $P_{SC}(\lambda)$ for the three intersections are 13.13%, 17.87% and 42.99%, separately. According to the above rule, we choose 17.59 as the approximation of the λ_{opt}. The variables selected when $\lambda = 17.59$ turns out to be exactly the same with the true ones.

Example 5:

Still use the same linear model in Example 1 with $a = 0.5$. The true variables are still x_1, x_3 and x_9. One sample generated from the model lead to the SV-plot with no intersections, as shown in Figure 2.5, where, the transition points are

$6.9, 11.8, 12.1, 13.2, 14.6, 14.8, 16, 18.2, 18.8, 19.7$. $g(\lambda) = 16.95$ when $\lambda \in (18.8, 19.7)$; $g(\lambda) = 15.4746$ when $\lambda \in (18.2, 18.8)$; $g(\lambda) = 13.6242$ when $\lambda \in (16, 18.2)$; $g(\lambda) = 12.5099$ when $\lambda \in (14.8, 16)$; $g(\lambda) = 11.6012$ when $\lambda \in (14.6, 14.8)$; $g(\lambda) = 9.7974$ when $\lambda \in (13.2, 14.6)$; $g(\lambda) = 9.0820$ when $\lambda \in (12.1, 13.2)$; $g(\lambda) = 7.6443$ when $\lambda \in (11.8, 12.1)$; $g(\lambda) = 6.2069$ when $\lambda \in (6.9, 11.8)$;

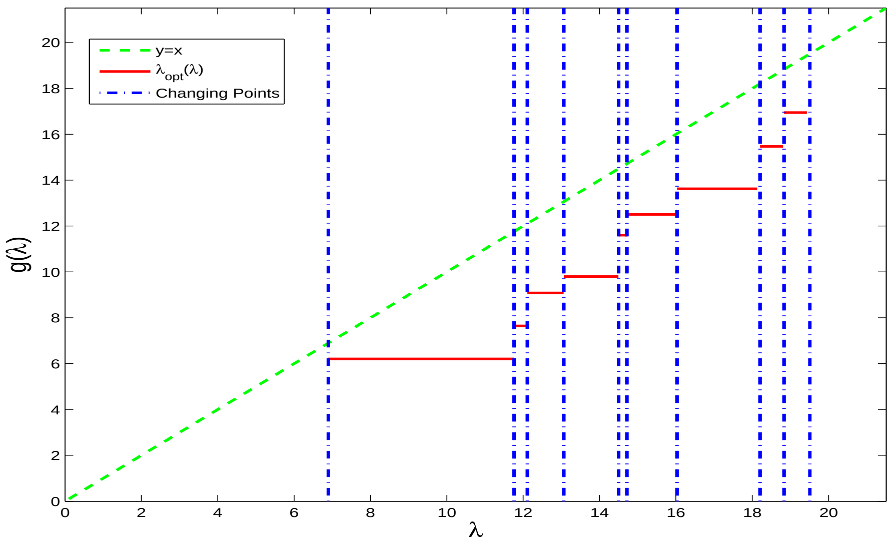


Figure 2.5.: $y = g(\lambda)$ and $y = \lambda$ No Intersection

As discussed before, the lasso path has been separated as several intervals by the 'changing points' of λ. Within each interval, there is a corresponding $g(\lambda)$ calculated by the SV procedure, defined as λ_i^*, $(i \in (1, s))$, where, s is the number of the intervals along the lasso path. Define the corresponding bound of the λ as λ_{i1} and λ_{i2}, (λ_{i1}¡λ_{i2}). Let $D_i = min(|\lambda_i^* - \lambda_{i1}|, |\lambda_i^* - \lambda_{i2}|)$, which is the smaller one for the two distances between λ_i^* and λ_{i1}, and λ_i^* and λ_{i2}, at the ith interval. Intuitively, the smaller the D_i, the more likely the corresponding model in the lasso path will be selected by the self-voting. Therefore we suggest to choose the model for none intersections scenario as follows:

Selection Rule for None Intersection Scenario:

$$k = \arg \max_{i:1\leq i\leq s} D_i, \tag{2.13}$$

where, k is the index of the interval in the lasso path and the corresponding model is the one selected by the self-voting for none intersection scenarios.

Example 5 Continued:

The corresponding D_i is $(1.88, 2.73, 2.40, 2.20, 2.9, 3.27, 3.03, 4.12, 0.699)$ for each of the interval in the lasso path. According to the selection rule for none intersection scenario, we suggest to choose the kth ($k = 9$) as the final selected model, which contains the true variables (x_1, x_3, x_9) but also some redundant variables $(x_2, x_5, x_7, x_10, x_12, x_16)$. Actually, when the none intersection scenarios happen, usually there doesn't exist an interval which contains the exact true variables without non-true ones. For example, in this example, when λ is within the range of the 3th interval of the lasso path, the variables in the corresponding model is x_1, x_9, x_10, and another true variable x_3 is added in until λ reduce to the 9th interval of the lasso path. Heuristically, a model tend to be self-voted when it contains all the true variables. Since in practice, people would rather to select an oversize model to avoid missing the true variables, therefore, the above rule can help to find the proper model which contains all the true variables with the size as smaller as possible.

In the above example, the none intersection scenarios are mostly caused by the low signal noise ratio, which means there is no enough information to decide the true model. In general, if Y is sampled from Equation 1 for fixed X and β, all these three scenarios can be possible. When the signal noise ratio is sharp, the first two scenarios are more likely to happen, and the model selected by the self-voting with the rules can guarantee a high probability to select the true models; when the signal noise is low, the last scenarios are more likely to happen, and the performance of the self-voting method with the rules will decrease. We use the next example to compare the frequency of the three scenarios for the samples from a fixed linear system, and the performance of the self-voting when the signal noise ratio changing.

Example 6:

We still use the same linear model in Example 1, where the true variables are still x_1, x_3 and x_9 and a varies from from 0.25 to 4. For each a, we generated $10,000$ samples of Y, and count the frequencies for the three scenarios and the correct selection for each of them by self-voting method with the rules. The results is shown in Table 2.1, where a represents the signal noise ratio, *Single*, *Multiple* and *None* represents the three scenarios discussed above, and *Total* represents the summation of the three scenarios. a is varying from 4 to 0.25, for the other four columns, each entry formed by three numbers, which indicate the number of the correct cases selected by self-voting, the number of the corresponding scenarios occurs, and the ratio of them, separately. For example, when $a = 2$, for all the $10,000$ samples of Y, there are 5166 cases will lead to the single intersection scenario, furthermore, among these 5166 cases, the self-voting method with the rules can select 4841 cases correctly, therefore the percentage of correct selection for single intersection scenario is 93.71%; similarly, the number of the multiple intersections scenario is 4831 and 4594 cases can be selected correctly, therefore the percentage for the multiple intersections scenario is 95.09%; and there are no none intersection scenario. The total number of the correct selection cases is 9435, which is the summation of 4841, 4594 and 0, and the overall percentage of the self-voting method with the rules is 94.35%.

From Table 2.1, as a decreases from 4 to 0.25, the percentage of the correct selection for the three scenarios are all decreases, which is consistent with the previous discussion. Meanwhile, the numbers of the cases of the *Single* and *Multiple* scenarios also decrease but the number of the cases of the none intersection increases. When a is larger than 1.5, there is barely no none intersection scenarios, but when a reduce to 0.25, the number of the none intersection scenarios jumps to 4760, but the percentage of the correct selection is very low (0.21%).

a	Total	Single	Multiple	None
4.00	9999/10000 (99.99%)	4977/4977 (100%)	5022/5023 (99.99%)	0/0 (0%)
3.00	9981/10000 (99.81%)	4862/4873 (99.77%)	5119/5127 (99.84%)	0/0 (0%)
2.75	9944/10000 (99.44%)	4919/4958 (99.21%)	5025/5042 (99.66%)	0/0 (0%)
2.50	9880/10000 (98.80%)	4804/4860 (98.85%)	5076/5139 (98.77%)	0/0 (0%)
2.25	9730/10000 (97.30%)	4867/5040 (96.57%)	4863/4960 (98.04%)	0/0 (0%)
2.00	9435/10000 (94.35%)	4841/5166 (93.71%)	4594/4831 (95.09%)	0/0 (0%)
1.75	8871/10000 (88.71%)	4716/5299 (89.00%)	4155/4701 (88.39%)	0/0 (0%)
1.50	7917/10000 (79.17%)	4374/5382 (81.27%)	3543/4604 (76.95%)	0/14 (0%)
1.25	6382/10000 (63.82%)	3573/5159 (69.26%)	2809/4811 (58.39%)	0/30 (0%)
1.00	4125/10000 (41.25%)	2411/5007 (48.15%)	1714/4888 (35.07%)	0/105 (0%)
0.75	1699/10000 (16.99%)	1004/4761 (21.09%)	669/4566 (14.65%)	26/673 (3.86%)
0.50	345/10000 (3.45%)	167/4016 (4.16%)	129/3530 (3.65%)	49/2454 (2.00%)
0.25	26/10000 (0.26%)	9/3254 (0.28%)	7/1986 (0.35%)	10/4760 (0.21%)

Table 2.1: Distributions of the Single, Multiple and No intersections cases with a

2.4 Simulation Studies

In this section, we run some numerical simulations to compare the performance of the proposed SV method with those of AIC-Lasso, BIC-Lasso, and CV-Lasso in determining the tuning parameter of lasso for variable selection. The index we use to quantify the performance of a method is the probability that the method selects the true model. We refer to this index as the P-index. Notice the calculation of AIC-Lasso, BIC-Lasso, and CV-Lasso all involve the same term $|Y - \beta X|_2^2$, where Y is the response, X is the design matrix, and β is the regression coefficient vector. Two different types of estimates of β are used, which are the lasso estimate and the OLS estimate, respectively.

Because these two different types of estimates can lead to different performances of AIC-Lasso, BIC-Lasso, and CV-Lasso, both are used in the simulation study and their corresponding results are reported. In addition, we apply all the four methods to analyze the well-known diabetes data and compare their results.

Example 7:

In this example, we consider the 12×16 supersaturated design used in Example 1 and the three different specifications of the true variables. Firstly, we designated x_1, x_3 and x_9 to be the true variables. Under this specified true model, $max(|R|) = 0.58$. Secondly, we designated the x_1, x_6 and x_{12} to be the true model, and the resulting $max(|R|) = 1$. Thirdly, we designated the x_5, x_8 and x_{11} to be the true model and the resulting $max(|R|) = 1.0952$.

Note that, the three different designations represent three different scenarios of the irrepresentable condition. In the first one, the irrepresentable condition is satisfied; in the second one, the irrepresentable condition is on the boundary; in the last one, the irrepresentable condition is violated.

Under the first specified model, we set β^0 equal to $(a, 0, a, 0, 0, 0, 0, 0, a, 0, 0, 0, 0, 0, 0, 0)'$ and vary from from 0.1 to 4 with the increment of 0.1. At each level of a, we generate 10,000 independent samples from the model and each of them contain a vector of responses of each design points. For each sample, we apply the SV, CV-Lasso, AIC-Lasso and BIC-Lasso separately to select the true variables. If a method selects are exactly x_1, x_3 and x_9, it's counted as a success or correct selection. The probability that a method can select the true model is estimated by the proportion of the successes the method achieves over the 10,000 samples. We use P_{SV}, P_{CV}, P_{AIC} and P_{BIC} to denote the success proportions of the SV, CV, AIC and BIC, respectively. The simulation results are presented in Table 2.2.

Notice Table 2.2 includes $P_{\lambda_{opt}}$ which is calculated from Theorem 1 for the given β^0s. According to Theorem 1, for any fixed λ, there exists a P_{SV} which selects the true variables. Among them, $P_{\lambda_{opt}}$ is the optimal. The column of P_{CV} is separated into two sub-columns, which are $\hat{\beta}_{OLS}$ and $\hat{\beta}_{Lasso}$ as discussed in the beginning of this section. Similarly, the columns of P_{AIC} and P_{BIC} also separated into two sub-columns. From Table 2.2, we can see that the overall performances of CV is very poor comparatively. Furthermore, for CV, AIC and BIC, the performances based on the OLS estimations are always better than the ones based on the lasso estimations. The performances of SV are comparable with the OLS-based AIC and the OLS-based BIC when $a \leq 0.9$ but becomes dominant when $a \geq 1$.

Specifically, as $a \leq 0.5$, the performances of all these methods are very poor ($\leq 5\%$). Among them, the OLS-based AIC is marginally better in general than the others and SV is comparable with the OLS-based BIC. Under these values of a, all theses methods do not perform as expected because the signals are very weak; As a increases from 0.6 to 0.9, the performances of all these methods continue to improve. Among them, the OLS-based BIC starts to exceed the others from 8.2% at $a = 0.6$ but only reach up to 31.2% at $a = 0.9$. The performances of SV are slightly below the OLS-based AIC when $a \leq 0.8$ but exceed the latter at $a = 0.9$ ($P_{CV} = 24.7\%$ and the OLS-based $P_{AIC} = 24\%$). As a increases from 1.0 to 1.5, P_{SV} increases much faster than the others and dominate. At $a = 1.0$, P_{SV} is equal to 0.425, which is larger than the OLS-based P_{AIC} (0.248) and the OLS-based P_{BIC} (0.342). At $a = 1.5$, P_{SV} is equal to 0.794, which almost doubles the OLS-based P_{BIC} (0.428) and triples the

OLS-based P_{AIC} (0.294). As a increases from 1.6 to 4, P_{SV} keep improving and reach to 100% accuracy in variable selection. However the OLS-based AIC and OLS-based BIC stagnate and do not show any significant improvements. The OLS-based P_{AIC} varies around 30% and up to 32% and OLS-based P_{BIC} varies around 41% and up to 44%. We also plot these simulation results and the similar patterns can be seen in Figure 2.6, where the solid lines are the results based on the OLS estimates and the dash lines are the results based on the lasso estimates.

Under this model, the irrepresentable condition is satisfied. According to the theorem 1, as a increases, which is equivalent to the sample size n increases, there exists a λ that the probability that selects the true variables will go to 1. As shown in Table 2.2, P_{SV} can actually achieve that which is consistent with what the asymptotic results guarantees. But CV, AIC and BIC fail to do that even when a is large.

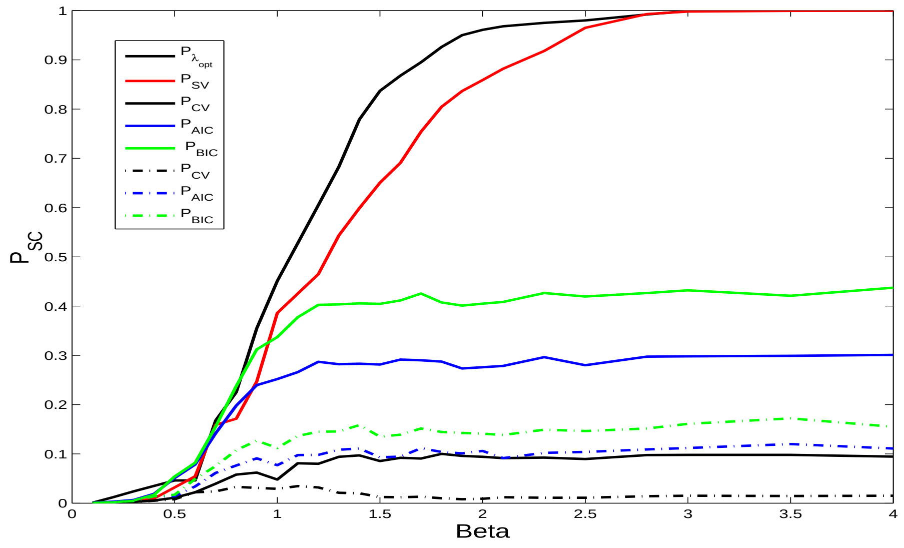


Figure 2.6.: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso $n = 12, p = 16, max(|R|) = 0.58$

The same simulation scheme is applied to the second model (x_1, x_6 and x_{12} are the true variables) and third models (x_5, x_8 and x_{11} are the true variables). The results

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.10	0.001	0.000	0.001	0.001	0.001	0.001	0.001	0.001
0.20	0.012	0.003	0.001	0.000	0.003	0.003	0.002	0.002
0.30	0.024	0.003	0.003	0.002	0.007	0.005	0.006	0.005
0.40	0.035	0.010	0.006	0.005	0.019	0.014	0.017	0.014
0.50	0.046	0.031	0.010	0.009	0.040	0.024	0.037	0.028
0.60	0.059	0.055	0.022	0.022	0.079	0.034	0.082	0.049
0.70	0.168	0.137	0.040	0.024	0.142	0.062	0.156	0.076
0.80	0.225	0.172	0.058	0.033	0.198	0.077	0.214	0.108
0.90	0.355	0.247	0.062	0.031	0.240	0.091	0.312	0.127
1.00	0.451	0.425	0.067	0.043	0.248	0.094	0.342	0.136
1.10	0.528	0.426	0.081	0.035	0.266	0.098	0.378	0.137
1.20	0.605	0.465	0.080	0.032	0.287	0.098	0.403	0.145
1.30	0.683	0.544	0.094	0.021	0.282	0.109	0.404	0.146
1.40	0.779	0.599	0.097	0.020	0.283	0.111	0.406	0.159
1.50	0.837	0.794	0.099	0.032	0.294	0.124	0.428	0.171
1.60	0.868	0.691	0.092	0.012	0.292	0.095	0.412	0.139
1.70	0.895	0.754	0.091	0.013	0.290	0.112	0.426	0.152
1.80	0.926	0.805	0.100	0.010	0.288	0.104	0.408	0.145
1.90	0.950	0.837	0.096	0.008	0.274	0.101	0.401	0.143
2.00	0.961	0.950	0.100	0.024	0.299	0.101	0.426	0.140
2.10	0.968	0.882	0.092	0.012	0.279	0.092	0.409	0.139
2.30	0.975	0.918	0.093	0.011	0.297	0.102	0.427	0.149
2.50	0.980	0.987	0.105	0.030	0.310	0.137	0.436	0.186
2.80	0.992	0.993	0.098	0.014	0.298	0.109	0.427	0.152
3.00	0.999	0.995	0.098	0.028	0.290	0.110	0.420	0.154
3.50	1.000	1.000	0.098	0.015	0.299	0.120	0.421	0.172
4.00	1.000	1.000	0.095	0.015	0.301	0.111	0.438	0.155

Table 2.2: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso $n = 12, p = 16, max(|R|) = 0.58$

are reported in Table 2.3 and Table 2.4, respectively. Due to the space limitation, we only include the selected results that a varies from 0.25 to 4. The definitions of columns are the same as in Table 2.2.

The similar patterns can be found in Table 2.3 and Table 2.4. The overall performances of CV is comparatively poor than the others, and the performances based on the lasso estimation are always lower than the performances which are based on OLS-based estimations. In Table 2.3, when $a \leq 1$, the OLS-based BIC performs slightly better than SV and OLS-based AIC, but reach up to 17.7%. When $a \geq 2$, SV starts to performs better than the others and reach up to 50%, while the OLS-based AIC stagnate around 14% and the OLS-based BIC stagnate around 20%. Notice P_{SV} can not surpass 50% because the second model is just on the boundary of the irrepresentable condition. According to Theorem 1, the upper bound of the P_{SV} will be 50% as a increases, which is consistent with what the asymptotic results dedicates.

Similarly, in Table 2.4, when $a \leq 1$, the OLS-based BIC still performs better than SV and OLS-based AIC, but can only reach up to 12.4%. When $a \geq 2$, SV become superior and reach to 25.6% at $a = 4$, which is higher than OLS-based P_{BIC} (21.2%) and OLS-based P_{AIC} (16.4%). According to Theorem 1, the upper bound of $P_{\lambda_{opt}}$ is equal to 31% as a increases. Due to the violation of the irrepresentable condition, the probability to select the true model under asymptotic scenarios should be 0. But this example shows that even the irrepresentable condition is violated, there still exists certain chances to select the true variables even in the asymptotic scenarios.

Example 8

In this example, we consider three larger supersaturated designs with the dimensions as 12×22, 24×46 and 48×94, separately. The three designs are the E_{S^2} optimal designs which achieve the E_{S^2} bounds. These designs are given in the supplementary documents. Comparing to the 12×16 design used in the previous example, these designs are more challenging for applying the SV algorithm because the calculation of λ_{opt} in SV algorithm involves a higher dimensional integration. Thus, instead

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.25	0.001	0.002	0.000	0.000	0.002	0.001	0.002	0.002
0.50	0.019	0.011	0.006	0.001	0.023	0.002	0.024	0.011
0.75	0.088	0.060	0.020	0.002	0.087	0.011	0.108	0.033
1.00	0.214	0.175	0.034	0.004	0.131	0.023	0.177	0.044
2.00	0.492	0.462	0.042	0.005	0.140	0.035	0.203	0.056
3.00	0.503	0.496	0.040	0.006	0.136	0.041	0.195	0.066
4.00	0.502	0.498	0.049	0.010	0.140	0.051	0.201	0.071

Table 2.3: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso $n = 12, p = 16, max(|R|) = 1$

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.25	0.001	0.001	0.001	0.000	0.002	0.001	0.001	0.001
0.50	0.009	0.005	0.004	0.009	0.014	0.018	0.015	0.021
0.75	0.040	0.021	0.015	0.008	0.048	0.014	0.060	0.018
1.00	0.099	0.061	0.031	0.022	0.088	0.027	0.114	0.037
2.00	0.301	0.250	0.073	0.010	0.140	0.029	0.183	0.040
3.00	0.310	0.253	0.081	0.004	0.149	0.034	0.193	0.044
4.00	0.310	0.256	0.101	0.004	0.164	0.035	0.212	0.044

Table 2.4: Comparison with CV-Lasso, AIC-Lasso, BIC-Lasso, $n = 12, p = 16, max(|R|) = 1.0952$

of using the exact integration calculation, we apply the lower bound in Section 2.2 (Equation 2.10) or Monte Carlo method proposed in Section 2.3 (Equation 2.11) to approximate λ_{opt} for different designs. As discussed before, the lower bound approximation has the highest computing efficiency but the least accuracy comparing to the exact integration calculation and Monte Carlo approximation. On the other hand, the Monte-Carlo approximation could offers acceptable accuracy but take more computational burden.

Similar with the previous example, the responses are generated from the linear model, $Y = X\beta + \epsilon$. Here, β is the coefficients of the variables. For the non-true variables, the value of the corresponding coefficients are all equal to 0; for the true variables, they are all equal to a. In this examples, we consider four levels of a which is 0.5, 1, 2 and 3, separately. ϵ follows $N(0, \sigma^2 I_n)$, where, n is the sample size of the designs. For each level of a, we generate 10,000 independent samples from the model and each of them contain a vector of responses of each design points.

For each of the designs, we have to further specified a set of true variables. Here, for the 12×22 design, we choose x_2, x_{12} and x_{14}) with $max(|R|) = 0.5$; for the 24×46 design, we choose x_1, x_{11}, x_{31} and x_{37} with $max(|R|) = 0.5$; for the 48×94 design, we choose $x_{29}, x_{34}, x_{37}, x_{52}, x_{80}$ and x_{91} with $max(|R|) = 0.6678$.

We apply all the four methods to select the true model. As for the SV algorithm, we apply the lower bound method to approximate λ_{opt} for the 12×22 designs and the Monte Carlo method to approximate λ_{opt} for the 24×46 designs and 48×94 designs, respectively. The results are shown in Table 2.5, Table 2.6 and Table 2.7.

For Table 2.5, when $a \leq 0.75$, all the performances is very poor ($\leq 5\%$). When $a = 1$,The OLS-based BIC performs slightly better than SV and OLS-based AIC, but can only reach up to 12%. When $a \geq 2$, SV become the dominant and reach to 94% at $a = 4$, which is significantly higher than OLS-based P_{BIC} (38.4%) and OLS-based P_{AIC} (27.6%); Similarly, for Table 2.6, when $a \leq 0.75$, the OLS-based BIC performs slightly better than SV and OLS-based AIC, but capped at 17.3%. When $a \geq 1$, SV becomes the dominant and achieves to 100% at $a = 4$, while the OLS-based P_{BIC}

stagnate around 30% and OLS-based P_{AIC} stagnate around 7%; For Table 2.7, when $a \leq 0.75$, the OLS-based BIC still performs better than SV and OLS-based AIC, but reach up to 19.7%. When $a \geq 1$, SV become the dominant and reach to 100% at $a = 4$, while the OLS-based P_{BIC} stagnate around 25% and OLS-based P_{AIC} stagnate around 1%.

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.50	0.004	0.001	0.001	0.001	0.004	0.001	0.004	0.001
0.75	0.013	0.008	0.004	0.002	0.035	0.004	0.012	0.005
1.00	0.053	0.039	0.017	0.011	0.099	0.011	0.120	0.015
2.00	0.601	0.602	0.057	0.197	0.249	0.012	0.384	0.019
3.00	0.951	0.940	0.048	0.446	0.276	0.011	0.383	0.021

Table 2.5: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso, $n = 12, p = 22, max(|R|) = 0.6$

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.50	0.134	0.070	0.007	0.018	0.020	0.022	0.040	0.058
0.75	0.431	0.008	0.018	0.035	0.052	0.054	0.173	0.141
1.00	0.722	0.594	0.027	0.020	0.064	0.051	0.269	0.149
2.00	0.997	0.954	0.028	0.022	0.078	0.050	0.279	0.178
3.00	1.000	1.000	0.035	0.018	0.073	0.057	0.300	0.174

Table 2.6: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso, $n = 24, p = 46, max(|R|) = 0.5$

a	$P_{\lambda_{opt}}$	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
			$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
0.50	0.030	0.070	0.000	0.000	0.002	0.000	0.045	0.009
0.75	0.396	0.008	0.001	0.000	0.005	0.001	0.197	0.013
1.00	0.796	0.758	0.001	0.000	0.008	0.000	0.257	0.014
2.00	1.000	0.998	0.001	0.000	0.004	0.000	0.247	0.015
3.00	1.000	1.000	0.001	0.000	0.009	0.000	0.252	0.013

Table 2.7: Comparison of SV, CV-Lasso, AIC-Lasso, and BIC-Lasso, $n = 48, p = 94, max(|R|) = 0.6678$

Example 9

In the two previous examples, the designs are fixed. In this example, we consider two linear models with random designs and power decay correlations. The first model uses the n by 8 design proposed in Zou and Tibshirani (2007). Hence, n is the prespecified run size varying from 100 to 2000 and the number of predictors is 8. The response vectors y are generated from a linear model: $y = X\beta + N(0,1)$ and $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)$. The design columns $x_i, (i = 1, 2, ...8)$ are multivariate normal vectors with pairwise correlation $corr(i,j) = (0.1)^{|i-j|}$ for any pair of x_i and x_j,$(i,j = 1, 2, ...8)$. The performances of the above model selection methods are given in Table 2.8 after repeat 10,000 times. The results show that performances of SV are always better than the others and reach 100% since $n \geq 200$. The performances of OLS-based BIC is comparable with SV but slightly worse than SV; and the performances of OLS-based AIC can only reach up to 51.7% even $n = 2000$.

n	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
		$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
100	0.874	0.476	0.280	0.530	0.226	0.869	0.323
200	0.996	0.479	0.306	0.527	0.252	0.927	0.360
300	1.000	0.467	0.320	0.515	0.248	0.934	0.387
400	1.000	0.475	0.299	0.534	0.236	0.951	0.385
500	1.000	0.468	0.300	0.513	0.227	0.954	0.375
1000	1.000	0.465	0.303	0.505	0.232	0.964	0.406
2000	1.000	0.473	0.322	0.517	0.242	0.979	0.432

Table 2.8: Comparison of SV, CV, AIC-Lasso and BIC-Lasso: Random Design with $p = 8$, $n = 100, 200, 300, 400, 500, 1000, 2000$ (Zou and Tibshirani, 2007)

The second model use the $n \times 100$ random supersaturated designs, and n is the prespecified run size varying from 20 to 80, and the number of the factors is 100. The design matrix Xs are generated each time from $N(0, \Sigma)$, where $\Sigma_{ij} = 0.5^{|i-j|}$ for all

i, j. The coefficients β_j^* follows $N(3,1)$ for $j \in \{1, ..., 6\}$, and $\beta_j^* = 0$ otherwise. The responses y are generated from the linear model $y = X\beta + \sigma\epsilon$, where ϵ follows $N(0, I)$ and $\sigma = 1$. Similarly, we can get the probability of correct selection for all the model selection methods. The results of $10,000$ replications are shown in Table 2.9. The results shows that when $n = 20$, the OLS-based BIC performs slightly better than SV. But when $n \geq 40$, SV becomes dominant and the accuracies of selecting the true model increase with n. When $n = 80$, P_{SV} is equal to 73.4% while the OLS-based P_{BIC} is only 28.3% and the OLS-based P_{AIC} is only 0.92%.

n	P_{SV}	P_{CV}		P_{AIC}		P_{BIC}	
		$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
20	0.075	0.026	0.002	0.054	0.004	0.112	0.009
40	0.496	0.012	0.000	0.051	0.007	0.307	0.074
60	0.663	0.005	0.000	0.074	0.014	0.280	0.134
80	0.734	0.002	0.000	0.092	0.019	0.283	0.189

Table 2.9: Comparison of SV, CV, AIC-Lasso and BIC-Lasso: Random Design with $p = 100$, $n = 20, 40, 60, 80$

Example 10: 'Diabetes' Data

Lastly, we test all these model selection methods on the 'Diabetes' data with 442 samples (Efron, 2004). There are 10 baseline variables, age, sex, body mass index, average blood pressure and six blood serum measurements, which are denoted as "$age, sex, bmi, map, tc, ldl, hdl, tch, ltg, glu$" separately. If only considering the 10 main factors, the lasso path show that when λ decreases, the variables are selected in follows the order "$bmi, ltg, map, hdl, sex, glu, tc, tch, ldl, age$". The optimal model selected by AIC, BIC and CV is "bmi, ltg, map, hdl, sex", which is the same with the optimal model selected by self-voting method.

When considering the interactions of the main factors, that is, 64 factors in total, the model selected by SV contains 11 variables. To verify the feasibility of our method, we randomly divided the 442 sample size into two parts and use the first part to obtain the model and the second part to test the fitness measured by MSE. We try different ratios such as 9 : 1, 7 : 3, 5 : 5 and 3 : 7, separately; and repeat 1,000 times for each of them to compare the average MSE. Furthermore, we use the original lasso path calculated from the full samples to calculate the corresponding MSE too. A_i refer to the model at the ith step along the lasso path. Because A_i is generated from the full data, thus MSE calculated by this fixed A_i will be the least, denoted as MSE_{A_i}. We can see that, comparing with other model selection methods, SV always has the least MSE no matter the ratio of the two parts.

The variables selected at the 4th step is "bmi, ltg, map, hdl", denoted as A_4. Similarly we have A_9 to represent the model selected at the 9th step which is "$bmi, ltg, map, hdl, bmi * map, age * sex, glu^2, bmi^2, age * map$", A_{11} to be "$bmi, ltg, map, hdl, bmi * map, age * sex, glu^2, bmi^2, age * map, age * glu, sex$" and A_{12} to be "$bmi, ltg, map, hdl, bmi * map, age * sex, glu^2, bmi^2, age * map, age * glu, sex, glu$".

Ratio	A_i	MSE_{A_i}	MSE_{SV}	MSE_{CV}		MSE_{AIC}		MSE_{BIC}	
				$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$	$\hat{\beta}_{OLS}$	$\hat{\beta}_{Lasso}$
9 : 1	A_{12}	1.2968	1.3217	1.3335	1.3987	1.4224	1.4176	1.4224	1.4176
7 : 3	A_{11}	3.9767	4.0240	4.0787	4.1786	4.2397	4.2243	4.2397	4.2243
5 : 5	A_9	6.9194	6.9781	7.1260	7.2247	7.4311	7.3800	7.4311	7.3800
3 : 7	A_4	10.019	10.355	10.849	10.924	11.700	11.582	11.700	11.582

Table 2.10: Average MSE value ($\times$ 10^5) of random testify, repeat 1000 times.

3. OPTIMAL SUPERSATURATED DESIGN VIA LASSO

3.1 Lasso-based Optimality Criteria

In the previous chapter, we have studied the finite sample properties of Lasso and have developed an efficient algorithm to select the optimal tuning parameter of Lasso. Then it is feasible to develop the optimality criteria for constructing SSDs for screening experiments in the sense of optimizing the probability of correct selection directly. In this chapter, we will discuss how to generate the SSDs based on the probability of correct selection. Notice that part of this chapter has been published in our paper [51].

An obvious choice is to optimize P_{sc} in (4) directly. As discussed in Section 2.1, P_{sc} depends on X, A, β_A, and λ. To explicitly indicate the dependence, we use the notation $P_{sc}(X, A, \beta_A, \lambda)$ for P_{sc} in the following discussion.

While we do not know the set A, we can usually assume the upper limit of the number of true factor based on prior knowledge, the expert opinion, or the user decision, for example, to focus only on the top 10% of the most important factors. For convenience, the upper limit of the number of possible true factors is still denoted by q. Recall that p denotes the total number of factors. The goal of the screening procedure, therefore, is to find which q of the p factors are true variables. Without imposing additional prior knowledge, any subset of q factors can be the set that includes all true factors. Therefore, when constructing the optimal design, all possible subsets of q factors need to be considered. Let $\mathcal{A}$ be the collection of all possible subsets of q factors, and $C_{pq} = \begin{pmatrix} p \\ q \end{pmatrix}$. Define

$$\widetilde{P_{sc}}(X, \lambda) = C_{pq}^{-1} \sum_{A \in \mathcal{A}} P_{sc}(X, A, \beta_A, \lambda). \tag{3.1}$$

The probability $\widetilde{P_{sc}}(X, \lambda)$ can be interpreted as the average probability for Lasso to select the true variables with correct signs, and it does not depend on each individual A. The evaluation of $\widetilde{P_{sc}}(X, \lambda)$ still depends on the true coefficients vector β_A. One way to handle this is to define a threshold β_0 for the coefficients of the effects of the true factors, and we assume that we are only interested in identifying factors with their absolute regression coefficients above β_0, that is, $|\beta_A| \geq \beta_0$. Then replace β_A as β_{0A} , which is the vector of the thresholds with proper signs for the factors in A for $A \in \mathcal{A}$. For ease of discussion, we assume that the signs of the effect coefficients can be pre-determined. If that is not the case, then all combinations of both positive and the negative effect coefficient of all βs need to be considered, and their corresponding P_{sc} needs to be averaged.

Finally, we define

$$\overline{P_{sc}}(X, \lambda) = C_{pq}^{-1} \sum_{A \in \mathcal{A}} P_{sc}(X, A, \beta_{0A}, \lambda), \tag{3.2}$$

where β_{0A} is the vector of the thresholds with proper signs for the factors in A for $A \in \mathcal{A}$. Then the optimal design together with the optimal tuning parameter λ, which are denoted as X_{o} and λ_{o}, respectively, can be defined as follows,

$$(X_{\text{o}}, \lambda_{\text{o}}) = \arg\max_{X,\lambda} \overline{P_{sc}}(X, \lambda). \tag{3.3}$$

Now the new criterion ($\overline{P_{sc}}$) become a function of X (the design matrix), and λ (the tuning parameter), which is a two dimensional optimization problem. While the criterion is quite general, there are two major challenges of optimizing $\overline{P_{sc}}$ directly. Firstly, based on our numerical evaluation (which we do not report here), given a design, the optimal λ, the tuning parameter, is highly sensitive to the β_A. Therefore, while we can use the threshold β_0 to approximate the β_A, the resulting optimal design and tuning parameter can be very biased. Furthermore, even with the approximation, the calculation of $\overline{P_{sc}}$ involve high dimensional integrations, which can quickly become computationally prohibitive as the dimension p increases, making the optimization problem not solvable. To address the above two challenges, we want to develop a criterion that focuses on X and relatively easy to calculate, and assume that λ can be properly selected after X is determined and observations have been collected to estimate βs, here, the optimal λ using the P_{sc} criterion will be determined by the self-voting algorithm we proposed in chapter 2.3.

Based on the results demonstrated in Section 2.1, we can see that with fixed A, β, and σ^2, $\mathrm{P}(\mathcal{E}_1)$ and $\mathrm{P}(\mathcal{E}_2)$ depend on the means and variances of U and V, which are R, $4X_C'(I-P)X_C\sigma^2/\lambda^2$, $\frac{\lambda}{2}D$, and $(X_A{}'X_A)^{-1}\sigma^2$, respectively. And all the four terms can be furthermore written as the functions of three terms $X_C{}'X_A(X_A{}'X_A)^{-1}$, $X_A{}'X_A$, and $X_C{}'X_C$. When n goes to infinity, it is easy to show that $\mathrm{P}(\mathcal{E}_2)$ will converge to 1, therefore,

$$P_{sc(n\to\infty)} \to \mathrm{P}(\mathcal{E}_1) \to \mathrm{P}(|R|<1) \tag{3.4}$$

The above result is consistent with the irrepresentable condition of Formula (3) . In other words, when n is large, the probability of the correct selection can be approximated as the probability that the irrepresentable condition fulfills. Therefore, $R = X_C{}'X_A(X_A{}'X_A)^{-1}\mathrm{sgn}(\beta)$ play a pivotal role in both the asymptotic and finite properties of Lasso. Again, we assume that the signs of the effect coefficients can be pre-determined. Intuitively, Each row of $X_C'X_A(X_A'X_A)^{-1}$ or equivalently each column of $(X_A'X_A)^{-1}X_A'X_C$ consists of the estimated regression coefficients generated from regressing an untrue factor against all true factors. Let X_j be an untrue factor, regressing X_j on X_A produces the vector of regression coefficients focusing on $X_C{}'X_A(X_A{}'X_A)^{-1}$ only. Define

$$\gamma_{j,A} = (X_A'X_A)^{-1}X_A'X_j, \tag{3.5}$$

which we refer to as the *representation index vector* of X_j by the true factors X_A. The representation index vector reflects how well X_j can be represented by the true factors, which can be considered as the correlation between X_j and the group of true factors. The larger the absolute values of the γ_j, the more difficult it is to separate X_j from X_A. The irrepresentable condition of Lasso basically requires that the absolute sum of the representation indices of all the untrue factors be bounded from above by

1. The smaller the representation indices, the more efficient is the design matrix X for Lasso to select the true variables. Therefore, the representation indices can be used to define the optimality criteria for Lasso. There are various ways to measure the magnitude of the representation index vector. For example, the L_∞, L_1, and L_2 norms of $\{\gamma_{j,A}, j \in C\}$ can all be considered. Here we only focus on the L_2 norm, which is denoted as RSS(A) and defined as follows,

$$\text{RSS}(A) = \sum_{j \in C} \|\gamma_{j,A}\|_2^2 = trace(X_C{}'X_A(X_A{}'X_A)^{-2}X_A{}'X_C). \tag{3.6}$$

For a given design X, the average of RSS(A) over all possible subsets of q factors is then defined as

$$\overline{\text{RSS}}_q(X) = C_{pq}^{-1} \sum_{A \in \mathcal{A}} \text{RSS}(A), \tag{3.7}$$

where C_{pq} is defined as the same as before. The optimal design X_o is defined as the minimizer of $\overline{\text{RSS}}_q(X)$ over all possible designs. We refer to X_o as the optimal $\overline{\text{RSS}}_q$ design. The subscript q indicates that the optimality criterion is for constructing optimal SSDs under the assumption that the number of true factors does not exceed q. Note that in the special case when $q = 1$, from the definition above, it's easy to show that for any given design X, $\overline{\text{RSS}}_1(X)$ is proportional to the $E(s^2)$ value of X. In other words, the optimization of the $\overline{\text{RSS}}_1$ is equivalent to the optimization of $E(s^2)$ when we assume there is only one important factor:

Theorem 3.

$$\overline{\text{RSS}}_1(X) = \frac{1}{n^2} E(s^2) \tag{3.8}$$

The deduction is given in Appendix 1. When $q > 1$, our proposed criterion is different from the $E(s^2)$ criterion. The numerical evaluation in Section 3.4 will show that the new criterion can generate superior designs with high $\widetilde{P_{sc}}$ values compared to $E(s^2)$ criterion. Compared with optimizing $\widetilde{P_{sc}}$ directly, the calculation of $\overline{\text{RSS}}_q$ is significantly faster, and the new criterion is not dependent on the tuning parameter and unknown coefficient, which makes the optimization more straightforward.

It is worth noting that when both the numbers of factors and possible number of true factors (q) become large, the collection $\mathcal{A}$ will be large and the evaluation of $\overline{\text{RSS}}_q$ can become prohibitively slow. There are various approaches to limit the size of $\mathcal{A}$. One approach is to select a sub-collection of $\mathcal{A}$ and the size of the sub-collection is limited by an upper bound. Furthermore, the sub-collection can be required to possess certain balance properties such as those of balance incomplete block designs, for example all factors and pair of factors should be included in equal number of A's. Another approach is to rank the involved factors with prior knowledge when applicable, and score the subset of q factors with their total ranks and then $\mathcal{A}$ only consists of those subsets with scores above a certain threshold. In other words, only those subsets or models that are most likely will be included in the construction of optimal designs. These two approaches make the criterion applicable in large-scale cases.

3.2 Partial Gradient Algorithms for RSS-Optimal SSDs

An important component of this work is to develop efficient algorithms to construct optimal SSDs based on the criteria proposed in Section 3.2. As discussed in Introduction, there are two major approaches to constructing optimal SSDs, the systematic construction based method and searching algorithms based methods. We focus on searching algorithm-based approach. We employ the idea of the column-wise pair-wise exchange algorithm but using more efficient way to select the candidate instead of randomly permuting or exchanging. We propose a partial gradient column exchange algorithm to generate the optimal designs. This approach is inspired by [52];, where they use the partially projected gradient to efficiently computing nonparametric maximum likelihood estimates of mixing distributions. The intuition of our algorithm is to find the partial gradient of the objective function with respect to each column, and use the gradient information to guide the exchange of the columns.

Despite the discrete of the designs, the partial gradient of the design will always provide some information or directions for approaching optimality.

The proposed $\overline{\mathrm{RSS}}_q$ criterion is a specific fit for this approach since its partial gradient is trackable. The first step of the partial gradient algorithm is to calculate the partial derivative of the RSS_q. Let X^k be an $n \times p$ design matrix where k represents the kth iteration step. The columns of X^k are denoted as $\{{X_1}^k, {X_2}^k, \ldots, {X_p}^k\}$, respectively, and the $\overline{\mathrm{RSS}}_q$ value of X^k is denoted as $\overline{\mathrm{RSS}}_q(X^k)$. The partial derivative of $\overline{\mathrm{RSS}}_q(X^k)$ with respect to ${X_b}^k$ ($b \in (1, 2, ..., p)$), denoted as PG_b^k, can be explicitly derived and the formulas are given in the following propositions.

Proposition 2. *When $q = 1$, The partial derivative of $\overline{RSS}_q(X^k)$ with respect to ${X_b}^k$, denoted as PG_b^k, is*

$$PG_b^k = \frac{\partial}{\partial X_b} \overline{\mathrm{RSS}}_1 = 4C_{p1}^{-1} n^{-2} X_c X_c' X_b, \tag{3.9}$$

Proposition 3. *When $q = 2$, Let $S = {X_j}' X_b, C = {X_j}' X_i, t = {X_b}' X_i$, n is the row number of X. The partial derivative of $\overline{RSS}_q(X^k)$ with respect to ${X_b}^k$, denoted as PG_b^k, is*

$$PG_b^k = \frac{\partial}{\partial X_b} \overline{\mathrm{RSS}}_2 = C_{p2}^{-1} PA_1^{(2)} + PA_2^{(2)} + PB^{(2)} \tag{3.10}$$

where,

$PA_1^{(2)} = 2 \sum\limits_{X_b \in A} \sum\limits_{j \in C} [(1,0) \left(\begin{smallmatrix} n & t \\ t & n \end{smallmatrix}\right)^{-2} \left(\begin{smallmatrix} S \\ C \end{smallmatrix}\right)] X_j,$

$PA_2^{(2)} = -2 \sum\limits_{X_b \in A} \sum\limits_{j \in C} (S, C) [\left(\begin{smallmatrix} n & t \\ t & n \end{smallmatrix}\right)^{-1} \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} n & t \\ t & n \end{smallmatrix}\right)^{-2} + \left(\begin{smallmatrix} n & t \\ t & n \end{smallmatrix}\right)^{-2} \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} n & t \\ t & n \end{smallmatrix}\right)^{-1}] \left(\begin{smallmatrix} S \\ C \end{smallmatrix}\right) X_i;$

$PB^{(2)} = 2 \sum\limits_{X_b \notin A} X_A (X_A' X_A)^{-2} X_A' X_b.$

Proposition 4. *When $q = 3$, Let $S = {X_j}' X_b$, $C_1 = {X_j}' X_{i_1}$, $C_2 = {X_j}' X_{i_2}$, $H_1 = {X_{i_1}}' X_{i_2}$, $t_1 = {X_b}' X_{i_1}$, $t_2 = {X_b}' X_{i_2}$, $n = {X_b}' X_b = {X_{i_1}}' X_{i_1} = {X_{i_2}}' X_{i_2}$, n is the row number of X. The partial derivative of $\overline{RSS}_q(X^k)$ with respect to ${X_b}^k$, denoted as PG_b^k, is*

$$PG_b^k = \frac{\partial}{\partial X_b} \overline{\mathrm{RSS}}_3 = C_{p3}^{-1} PA_1^{(3)} + PA_2^{(3)} + PA_3^{(3)} + PB^{(3)} \tag{3.11}$$

Where,

$$PA_1^{(3)} = 2\sum_{X_b\in A}\sum_{j\in C}((1,0,0)B^{-2}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} X_j');$$

$$PA_2^{(3)} = -2\sum_{X_b\in A}\sum_{j\in C}((S,C_1,C_2)B^{-1}[\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} B^{-1}+B^{-1}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}]B^{-1}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} X_{i_1}');$$

$$PA_3^{(3)} = -2\sum_{X_b\in A}\sum_{j\in C}((S,C_1,C_2)B^{-1}[\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} B^{-1}+B^{-1}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}]B^{-1}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} X_{i_1}');$$

$$PB^{(3)} = 2\sum_{X_b\notin A} X_A(X_A'X_A)^{-2}X_A'X_b;$$

$$B = \begin{pmatrix} n & t_1 & t_2 \\ t_1 & n & H_1 \\ t_2 & H_1 & n \end{pmatrix},$$

Proposition 5. *In general, for any given q, Let* $S = X_j{}'X_b$, $C_1 = X_j{}'X_{i_1}$, $C_2 = X_j{}'X_{i_2}$, $C_{q-1} = X_{i_1}{}'X_{iq-1}$, $t_1 = X_b{}'X_{i_1}$, $t_2 = X_b{}'X_{i_2}$, $n = X_b{}'X_b = X_{i_1}{}'X_{i_1} = X_{i_2}{}'X_{i_2} = \cdots = X_{iq-1}{}'X_{iq-1}$, $H_{1,2} = X_{i_1}{}'X_{i_2}$, $H_{1,q-1} = X_{i_1}{}'X_{iq-1}$, $H_{2,q-1} = X_{i_2}{}'X_{iq-1}$. *n is the row number of X. The partial derivative of* $\overline{RSS}_q(X^k)$ *with respect to* ${X_b}^k$*, denoted as* PG_b^k*, is*

$$PG_b^k = \frac{\partial}{\partial X_b}\overline{\mathrm{RSS}}_q = C_{pq}^{-1}PA_1^{(q)} + \sum_{h=2}^{q} PA_h^{(q)} + PB^{(q)}, \tag{3.12}$$

where,

$$PA_1^{(q)} = 2\sum_{X_b\in A}\sum_{j\in C}((1,0_1,0_2,\ldots,0_{q-1})B^{-2}\begin{pmatrix} S \\ C_1 \\ C_2 \\ \ldots \\ C_{q-1} \end{pmatrix} X_j')$$

$$PA_h^{(q)} = -2 \sum_{X_b \in A} \sum_{j \in C} ((S, C_1, C_2, \ldots, C_{q-1}) B^{-1} [M_q^h B^{-1} + B^{-1} M_q^h] B^{-1} \begin{pmatrix} S \\ C_1 \\ C_2 \\ \ldots \\ C_{q-1} \end{pmatrix} X'_{i_{h-1}}),$$

$PB^{(q)} = 2 \sum_{X_b \notin A} X_A (X'_A X_A)^{-2} X'_A X_b$,

$$B = \begin{pmatrix} n & t_1 & t_2 & \ldots & t_{q-1} \\ t_1 & n & H_{1,2} & \ldots & H_{1,q-1} \\ t_2 & H_{1,2} & n & \ldots & H_{2,q-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ t_{q-1} & H_{1,q-1} & H_{2,q-1} & \ldots & n \end{pmatrix},$$

M_q^h is the q by q matrix in which the element $(h, 1)$ and $(1, h)$ is 1 and all the others are 0. For example,

$$M_q^2 = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}_{q \times q}, M_q^3 = \begin{pmatrix} 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 1 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}_{q \times q}, \text{ and}$$

$$M_q^q = \begin{pmatrix} 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & 0 & 0 & \ldots & 0 \end{pmatrix}_{q \times q}.$$

Proposition 2-5 demonstrate that the partial derivatives of the $\overline{\text{RSS}}_q$ criteria can be calculated for all qs. The proof of the propositions are given in Appendix 2. With these formulas, we now propose the partial gradient-based algorithm to construct $\overline{\text{RSS}}_q$ optimal designs.

Partial Gradient Algorithm.

Since PG_b^k has the same dimension as ${X_b}^k$, where, $b \in [1, p]$, we can define the partial derivative matrix PG^k as $\{{PG_1}^k, {PG_2}^k, \ldots, {PG_p}^k\}$, where, ${PG_1}^k$ is the partial derivative of $\overline{\mathrm{RSS}}_q(X^k)$ with respect to ${X_1}^k$, ${PG_2}^k$ is the partial derivative of $\overline{\mathrm{RSS}}_q(X^k)$ with respect to ${X_2}^k$, and so on. The partial gradient algorithm for minimizing $\overline{\mathrm{RSS}}_q$ is described as follows.

1. Initialization: Set k=0, and select an initial design X^0.
2. Find $(i^*, j^*) = \arg\max\limits_{i,j}\{sgn(X_{ij}) \cdot PG_{ij}^k\}$ and $i^0 = \arg\max\limits_{i \in A_i^*}\{sgn(X_{ij^*}) \cdot PG_{ij^*}^k\}$, where $A_i^* = \{i : sgn(X_{ij^*}) = -sgn(X_{i^*j^*})\}$.
3. Exchange the signs of $X_{i^*j^*}^k$ and $X_{i^0j^*}^k$.
4. Iterate Step 2 and Step 3 until $|\overline{\mathrm{RSS}}_q(X^k) - \overline{\mathrm{RSS}}_q(X^{k-1}))| < \delta$, where δ is the prespecified threshold.

The intuition to consider $\max\{sgn(X_{ij}) \cdot PG_{ij}^k\}$ in Step 2 is as follows. When $sgn(X_{ij}) \cdot PG_{ij}^k > 0$, in other words, X_{ij} and PG_{ij}^k have the same sign, then changing the sign of X_{ij} will decrease $\overline{\mathrm{RSS}}_q(X^k)$. The larger the magnitude of PG_{ij}^k is, the larger the decrease(i.e. the improvement) of $\overline{\mathrm{RSS}}_q(X^k)$ will have. This is how we decide the i^* and j^*. Note that the SSDs we consider in this dissertation are balanced with equal numbers of high $(+1)$ and low (-1) levels. Therefore, to change the sign of $X_{i^*j^*}$, we need to balance the column by changing one opposite sign term in the same column. The selection of i^0 in Step 2 and changing the sign of $X_{i^0j^*}$ in Step 3 are to maintain the properly balance while avoiding the increase of $\overline{\mathrm{RSS}}_q(X^k)$. Here, we use an example to show how the algorithm works when using $\overline{\mathrm{RSS}}_3$ criterion.

Example 11:

The X^0 shown below is a randomly generated initial two level design with $n = 8$ and $p = 12$:

+	−	−	+	−	+	+	+	+	+	−	−
−	+	+	+	−	−	−	−	+	+	+	+
+	−	+	+	+	−	−	−	+	−	−	−
−	+	−	−	+	+	−	−	−	+	+	−
+	−	−	−	+	−	+	+	−	−	−	+
−	+	+	−	+	+	−	+	−	−	−	+
+	+	−	−	−	+	+	−	+	+	+	+
−	−	+	+	−	−	+	+	−	−	+	−

According to the Proposition 4, the partial gradient matrix of X^0 for $\overline{\text{RSS}}_3$ criterion, or PG^0, can be calculated as,

68.35	−79.72	−50.92	41.28	−51.86	30.72	68.35	23.68	51.86	41.00	−23.68	−25.70
−47.36	79.81	73.22	57.63	−59.01	−23.89	−47.36	−64.37	59.01	45.09	64.37	5.45
35.20	−74.62	45.98	69.64	45.57	−47.99	−56.90	−31.68	46.53	−46.37	−56.49	−16.08
−43.74	98.04	−32.39	−66.25	39.11	46.29	−43.74	−39.09	−39.11	49.32	39.09	−19.72
58.99	−73.56	−38.37	−64.28	48.28	1.04	58.99	53.27	−48.28	−86.96	−53.27	17.40
−52.39	57.26	30.58	−64.72	67.82	13.92	−52.39	31.79	−67.82	−60.20	−31.79	25.30
37.85	67.40	−74.08	−42.93	−43.38	27.89	37.85	−30.09	43.38	104.49	30.09	29.43
−56.90	−74.62	45.98	69.64	−46.53	−47.99	35.20	56.49	−45.57	−46.37	31.68	−16.08

Then calculates $sgnX^0. * PG^0$, where, $.*$ is the element-wise product. The result is shown below,

68.35	79.72	50.92	41.28	51.86	30.72	68.35	23.68	51.86	**41.00**(+)	23.68	25.70
47.36	79.81	73.22	57.63	59.01	23.89	47.36	64.37	59.01	**45.09**(+)	64.37	5.45
35.20	74.62	45.98	69.64	45.57	47.99	56.90	31.68	46.53	**46.37**(−)	56.49	16.08
43.74	98.04	32.39	66.25	39.11	46.29	43.74	39.09	39.11	**49.32**(+)	39.09	19.72
58.99	73.56	38.37	64.28	48.28	−1.04	58.99	53.27	48.28	$\underline{\mathbf{86.96}}$(−)	53.27	17.40
52.39	57.26	30.58	64.72	67.82	13.92	52.39	31.79	67.82	**60.20**(−)	31.79	25.30
37.85	67.40	74.08	42.93	43.38	27.89	37.85	30.09	43.38	$\underline{\mathbf{104.49}}$(+)	30.09	29.43
56.90	74.62	45.98	69.64	46.53	47.99	35.20	56.49	45.57	**46.37**(−)	31.68	16.08

The entry with the largest value (104.49) lies in 7th row and 10th column. Then the least favorable column is selected as 10th column, as highlight in the above table. Meanwhile, the entry with largest value (86.96) in the same column with the opposite sign of X^0 lies in 5th row and 10th column. Therefore, $X^0(7, 10)$ and $X^0(5, 10)$ form a pair of entries in the same column with opposite signs.

If we exchange the signs of $X^0(7, 10)$ and $X^0(5, 10)$, as shown in the following table. The new design, denoted as X^1, is still a balanced SSD. (The two exchanged entries are shown by circled symbols)

$$\begin{array}{cccccccccccc}
+ & - & - & + & - & + & + & + & + & + & - & - \\
- & + & + & + & - & - & - & - & + & + & + & + \\
+ & - & + & + & + & - & - & - & + & - & - & - \\
- & + & - & - & + & + & - & - & - & + & + & - \\
+ & - & - & - & + & - & + & + & - & \oplus & - & + \\
- & + & + & - & + & + & - & + & - & - & - & + \\
+ & + & - & - & - & + & + & - & + & \ominus & + & + \\
- & - & + & + & - & - & + & + & - & - & + & -
\end{array}$$

It is easy to verify that the $\overline{\mathrm{RSS}}_3$ values dropped from $238.47(X^0)$ to $196.62(X^1)$. Since all the $\overline{\mathrm{RSS}}_q$ are minimum criteria, the object function is significantly improved with this exchange. Replace X^0 by X^1 and continue the exchange until the $\overline{\mathrm{RSS}}_3$ stagnates, we will obtain the optimal design.

To avoid local stagnation and improve the quality of the final generated designs, a commonly used strategy is to consider multiple initial designs. Therefore, in our numerical experiments, we randomly select a group of initial deigns, run the above algorithm separately, and report the best subset of optimal designs (Section 3.4). The numerical study shows that the proposed algorithm is much more efficient compared to other existing columns exchanging algorithms. Here we demonstrate a comparison of the efficiencies of the proposed partial gradient algorithm and the random columnwise-pairwise algorithm [29] that has been popular in practice. The experiment is conducted on a 64-bit Window 8.1 desktop PC with Intel(R) Core(TM) i7-3517U CPU (1.90GHz) and 8.00 GB RAM. The results are shown in Table 3.1.

In the above table, we select 100 random-generated 12×16 initial designs and apply both the partial gradient algorithm and Columnwise-Pairwise algorithm. In average, it takes 20 iterations and 35 seconds for the partial gradient algorithm to

	RSS_1	Iterations	Time Consumed(second)
Partial Gradient	5.33	20	35
Columnwise-Pairwise	5.33	20,000	7,200

Table 3.1: Partial Gradient Algorithm vs Columnwise-Pairwise Algorithm

converge to the 95% of the RSS_1 lower bound, which is equivalent to the $E(s^2)$ lower bound according to Theorem 3. Meanwhile, when using Columnwise-Pairwise algorithm, it takes more than 20,000 iterations in average and more than two hours of computer time to reach to the similar optimal results. This huge difference shows that the potential of our approach for constructing large- scale designs, which has barely been touched in literature and is badly needed in practice.

3.3 Numerical Evaluations

We study the RSS-optimal SSDs for three scenarios: $(n = 12, p = 16)$, $(n = 24, p = 46)$ and $(n = 48, p = 94)$. Specifically, we randomly generate 50 designs as initial designs for each scenario and applied the partial gradient algorithm to generate the optimal designs. Therefore for each scenario there are 50 optimal designs generated. For $(n = 12, p = 16)$, we pick the best four designs based on $\overline{\text{RSS}}_3$, denote them as X^1, X^2, X^3 and X^4, and report them in Table 3.2 with the ordered $\overline{\text{RSS}}_3$ values(from the smallest to the largest). In other words, design X^1 has the smallest $\overline{\text{RSS}}_3$ whereas X^4 has the fourth smallest $\overline{\text{RSS}}_3$. We also list the $\overline{\text{RSS}}_q$ values for $q = 1$ and 2. Similarly, for $(n = 24, p = 46)$, we pick the six best generated designs based on smallest $\overline{\text{RSS}}_5$, denote them as X^1, X^2, X^3, X^4, X^5 and X^6, and report the $\overline{\text{RSS}}_q$ values for $q = 1$ to 5 in Table 3.3. For $(n = 48, p = 94)$, we pick the six best generated designs based on smallest $\overline{\text{RSS}}_7$, denote them as X^1, X^2, X^3, X^4, X^5 and X^6, and report the $\overline{\text{RSS}}_q$ values of them for $q = 1$ to 7 in Table 3.4.

Design	X^1	X^2	X^3	X^4
$\overline{\text{RSS}}_1$	5.2	5.2	5.2	5.2
$\overline{\text{RSS}}_2$	140.9	141.4	143.7	147.2
$\overline{\text{RSS}}_3$	1927	1950	2016	2035

Table 3.2: $\overline{\text{RSS}}_q$ of 12×16 Designs for $q \in [1, 3]$

Design	X^1	X^2	X^3	X^4	X^5	X^6
$\overline{\text{RSS}}_1$	12.80	12.80	12.80	12.80	12.80	12.80
$\overline{\text{RSS}}_2$	2173	2179	2192	2194	2196	2204
$\overline{\text{RSS}}_3$	99347	99358	101120	101264	101860	102049
$\overline{\text{RSS}}_4$	139613	142964	143629	144193	145204	145826
$\overline{\text{RSS}}_5$	185007	189591	191115	193309	194324	194492

Table 3.3: $\overline{\text{RSS}}_q$ of 24×46 Designs for $q \in [1, 5]$

Design	X^1	X^2	X^3	X^4	X^5	X^6
$\overline{\text{RSS}}_1$	24.77	24.77	24.77	24.77	24.77	24.77
$\overline{\text{RSS}}_2$	8942	8944	8949	8952	8959	8964
$\overline{\text{RSS}}_4$	128596	128633	128778	128804	128848	128897
$\overline{\text{RSS}}_5$	164751	164784	165146	165037	165078	165117
$\overline{\text{RSS}}_6$	202443	202665	203160	203058	203044	203120
$\overline{\text{RSS}}_7$	242481	242700	243203	243028	243238	243394

Table 3.4: $\overline{\text{RSS}}_q$ of 48×94 Designs for $q \in [1, 7]$

From Table 3.2-3.4, we can see that the $\overline{\text{RSS}}_1$ values of the picked designs in each table are the same. Based on Theorem 3, we can deduct the $E(s^2)$ value of the designs from the $\overline{\text{RSS}}_1$ values, and it is straightforward to verify that all $E(s^2)$ values reach the corresponding $E(s^2)$ optimal bound according to Theory 3.1 in [33]. Therefore all these designs are optimal at $E(s^2)$ criterion. However, they have different $\overline{\text{RSS}}_q$ values when $q > 1$, which indicates that the new criterion differentiate designs that are $E(s^2)$ optimal. In other words, given n and p, the RSS-optimal designs are likely different when the number of the true factors (q) are different. This observation is consistent with the fact that $E(s^2)$ is a special case for RSS criterion. We can also see that the orders of $\overline{\text{RSS}}_q$ are consistent for different q values in all the scenarios we have studied.

For example, in Table 3.2, $\overline{\text{RSS}}_2(X^1) \leq \overline{\text{RSS}}_2(X^2) \leq \overline{\text{RSS}}_2(X^3) \leq \overline{\text{RSS}}_2(X^4)$, which is consistent with $\overline{\text{RSS}}_3(X^1) \leq \overline{\text{RSS}}_3(X^2) \leq \overline{\text{RSS}}_3(X^3) \leq \overline{\text{RSS}}_3(X^4)$. Note that this consistency is not always true, but does hold most of the time with very rare exceptions. Majority of time, the order of $\overline{\text{RSS}}_{q_1}$ of the designs implies the order of $\overline{\text{RSS}}_{q_2}$ when $q_1 > q_2$.

We also generate the optimal $\overline{\text{RSS}}_q(q = 1, 2, 3)$ designs for various combinations of $n \times p$ from 10×10 to 50×100 by using the partial gradient algorithm and studied their properties. In Table 3.5 and Table 3.6, selected results are demonstrated. The first two columns represent the run size (n) and the number of the factors (p) of the optimal designs. The third column is the theoretical $E(s^2)$ lower bound [33] of the specified $(n,\ p)$ designs. The last three columns list the value of the corresponding $Es^2 = \overline{\text{RSS}}_1 * n^2$, $\overline{\text{RSS}}_2$ and $\overline{\text{RSS}}_3$ for generated (n,p) RSS-optimal designs. We can see that most of the designs' $E(s^2)$ reach the theoretical $E(s^2)$ lower bound proposed by [33], or are very close to the bound. This demonstrates that our algorithm is effective and the generated designs are valid. To facilitate the application of our proposed criterion and algorithm, We have built a web tool to generate the RSS-based optimal designs for any given dimensions. Interested readers can go `Http://engineering.purdue.edu/Smartdesigns` to learn more details.

We further compared the designs in Table 3.2-3.7 by their corresponding $\overline{P_{sc}}$ values using simulation. The simulation scheme is described as follows. For the given q, design X, and constant μ, we first randomly choose a subset A of q columns of X as the set of true factors, the regression coefficients of the true variables are randomly drawn from $N(\mu, 1)$ and their signs are randomly assigned; and the regression coefficients of all the untrue variables are assigned to be zero. Second, we use the linear model given in (1) to generate the response vector Y. Third, we identify the optimal tuning parameter λ_{opt} by the self-voting algorithm proposed in Section 2.3, and then screen the true factors using Lasso. Denoting the screening outcomes as $\hat{A}$, if $\hat{A} = A$, record 1; otherwise record 0. Repeat the above three steps for 10,000 times, and the proportion of 1's can be used to estimate Lasso's variable selection probability $\overline{P_{sc}}$.

n	p	$E(s^2)$ Bound	$Es^2 = \overline{\text{RSS}}_1 * n^2$	$\overline{\text{RSS}}_2$	$\overline{\text{RSS}}_3$
10	10	4	4	31.25	122.3922902
10	11	4	4	45.83333333	219.1519274
10	12	4	4	62.5	343.0748299
10	13	4.61538	4.820512821	97.25694444	590.6853632
10	14	5.05494	5.054945055	132.4583333	900.2602873
10	15	5.52381	5.523809524	180.1388889	1324.531944
10	16	5.86666	5.866666667	236.5	1889.020772
10	17	5.88235	5.882352941	290.8333333	2526.570913
10	18	5.882352941	5.882352941	350.25	3278.315789
12	12	2.18181	2.181818182	23.25	125.0714286
12	13	3.6923	3.692307692	48.33333333	273.7653061
12	14	4.21978	4.21978022	74.16666667	482.281746
12	15	4.57142	4.571428571	101.5	727.6984127
12	16	5.2	5.2	140.8611111	1082.653424
12	17	5.64705	5.647058824	185.0833333	1527.048073
12	18	5.96078	5.960784314	235.3333333	2085.708401
12	19	6.45614	6.456140351	301.6944444	2840.703798
12	20	6.82105	6.821052632	374.2916667	3735.771882
12	21	6.85714	6.857142857	442.2916667	4710.585952
12	22	6.85714	6.857142857	512.7	5769.171429
14	14	4	4	46.52083333	270.2886831
14	15	4	4	61.17708333	409.2102881
14	16	4	4.533333333	85.95555556	627.9687967
14	17	4.94117	4.941176471	115.42875	918.7705835
14	18	5.6732	5.673202614	159.1252778	1357.686431
14	19	6.05848	6.058479532	203.1261111	1858.895041
14	20	6.35789	6.357894737	251.7636111	2456.942682
14	21	6.66666	6.742857143	310.8609722	3205.960639
14	22	6.90909	6.909090909	370.2925	4046.424909
14	23	7.41502	7.541501976	461.5766667	5272.847761

Table 3.5: Optimal $\overline{\text{RSS}}_q$ Value Table for Different (n, p) Combinations (1)

n	p	E(s^2) Bound	$\overline{\text{RSS}}_1$	$\overline{\text{RSS}}_2$	$\overline{\text{RSS}}_3$
14	24	7.82608	7.826086957	548.3836111	6583.765254
14	25	7.84	8.053333333	641.4093056	8070.064656
14	26	7.84	8.135384615	732.8025	9642.974641
16	16	2.13333	2.133333333	29.25333333	203.2919728
16	17	3.7647	3.764705882	64.84	496.9040363
16	18	4.183	4.183006536	92.45666667	813.6456463
16	19	4.49122	5.14619883	129.985	1168.920641
16	20	5.38947	5.726315789	170.1461111	1624.315073
16	21	6.09523	6.323809524	218.8077778	2205.531807
16	22	6.64935	6.787878788	272.1825	2898.839198
16	23	7.083	7.209486166	332.3780556	3724.531693
16	24	7.42029	7.536231884	397.8761111	4689.602791
16	25	7.68	7.893333333	473.2955556	5838.965735
16	26	7.87692	8.073846154	548.0727778	7080.797771
16	27	8.388	8.478632479	647.2841667	8727.687627
16	28	8.804	8.888888889	757.5872222	10597.30693
16	29	8.828	9.064039409	863.6538889	12592.35619
16	30	8.828	9.048275862	958.5766667	14521.98596
18	18	4.000	4	63.54	503.8183066
18	19	4.000	4.374269006	84.21388889	727.9502754
18	20	4.000	5.010526316	114.3713889	1059.11554
18	21	5.143	5.676190476	152.0304167	1498.719465
18	22	5.801	6.216450216	193.4991667	2020.637987
18	23	6.150	6.782608696	242.7788889	2668.317754
18	24	6.667	7.130434783	292.6106944	3388.02092
18	25	7.200	7.626666667	356.1189909	4324.135102
18	26	7.643	8.036923077	424.3683333	5386.92675
18	27	8.103	8.376068376	497.4444161	6588.223229
18	28	8.487	8.825396825	586.88572	8094.68277
18	29	8.690	9.123152709	676.369915	9693.896475

Table 3.6: Optimal $\overline{\text{RSS}}_q$ Value Table for Different (n, p) Combinations (2)

n	p	$\mathrm{E}(s^2)$ Bound	$\overline{\mathrm{RSS}}_1$	$\overline{\mathrm{RSS}}_2$	$\overline{\mathrm{RSS}}_3$
18	30	8.855	9.370114943	772.6325567	11511.40758
18	31	9.368	9.711827957	886.2484637	13688.00619
18	32	9.807	10	1005.578866	16058.95067
18	33	9.818	10.24242424	1133.553648	18736.29695
18	34	9.818	10.4456328	1268.143912	21663.53864
20	20	2.105	4.042105263	72.86	656.1069299
20	21	3.810	4.571428571	96.60305556	926.3746545
20	22	4.156	5.471861472	134.8978175	1375.741358
20	23	4.427	6.134387352	174.556712	1881.155322
20	24	5.507	6.666666667	218.224263	2485.127534
20	25	6.400	7.04	262.718441	3143.217781
20	26	6.892	7.581538462	319.8430839	4000.465313
20	27	7.293	8.068376068	383.1860998	5003.898995
20	28	7.831	8.465608466	451.4357086	6154.874274
20	29	8.276	8.78817734	523.0467971	7421.64544
20	30	8.644	9.195402299	608.622483	8974.194109
20	31	9.118	9.565591398	699.2870011	10668.44595
20	32	9.516	9.903225806	799.9945011	12659.59627
20	33	9.697	10.27272727	911.1560658	14884.88569
20	34	9.840	10.52406417	1023.874031	17284.1211
20	35	10.353	10.81008403	1151.299099	20079.18315
20	36	10.794	11.0984127	1288.075	23137.07318

Table 3.7: Optimal $\overline{\mathrm{RSS}}_q$ Value Table for Different (n, p) Combinations (3)

We applied the simulation scheme to all three groups of the designs with $q = 3$ for 12×16 optimal designs, $q = 5$ for 24×46 optimal designs and $q = 7$ for 48×94 optimal designs separately. And the value of μ is varying from 0 to 5 with an increment of 0.5. The plots of the estimated $\overline{P_{sc}}$ versus μ for the three group of the designs are presented in the Figures 3.1-3.3. Since D_1 is the best generated design for each of the three scenarios, we further calculate the theoretical $\overline{P_{sc}}$ value of D_1 at each point of μ and plot the curves $D1_{PSC}$ in Figure 3.1-3.3. Notice that when β and σ^2 are known, the optimal tuning parameter λ and subsequently the P_{sc} value can be calculated directly from the equation (4). This curve can serve as an upper bound of the P_{sc} value since the λ is directly optimized based on the true values of all parameters.

In Figure 3.1, $D1, D2, D3, D4$ demonstrates the $\overline{P_{sc}}$ curves of designs X^1 through X^4, respectively, with $q = 3$. Notice that the $\overline{P_{sc}}$ curve of X^1, which is the best design according to the $\overline{\mathrm{RSS}}_3$ values, is above those of the other designs, over all μ values, indicating X^1 outperformed the other designs. In fact, the order of the $\overline{\mathrm{RSS}}_3$ values is the same as the order of the $\overline{P_{sc}}$ curves, indicating that the $\overline{\mathrm{RSS}}_q$ values have successfully discriminated the variable selection performances of the designs when using Lasso was used as the variable selection tool. We can also see that the $\overline{P_{sc}}$ curve of X^1 is quite close to the $D1_{PSC}$ curve. This implies that the achieved P_{sc} value is close to the theoretical value. Similarly, Figure 3.2 and Figure 3.3 show the $\overline{P_{sc}}$ curves of designs X^1 through X^6 for $n = 24, p = 46 (q = 5)$ and $n = 48, p = 94 (q = 7)$, respectively. And we made similar observations as in Figure 3.1.

In the above Figures, we can see that while all the designs we reported are $E(s^2)$ optimal which reach the $E(s^2)$ lower bound, the probability of correct selection of these designs are different, which can not be reflected by $E(s^2)$ criterion but are consistent with their $\overline{\mathrm{RSS}}_q$ values. This simulation study provides evidence that the our proposed RSS criterion is effective in selecting supersaturated designs with high P_{sc}s for Lasso. Specifically, designs with lower $\overline{\mathrm{RSS}}$ values also have better $\overline{P_{sc}}$ values. The design-only criterion we discussed in Section 3.2 works well as shown in the numerical study.

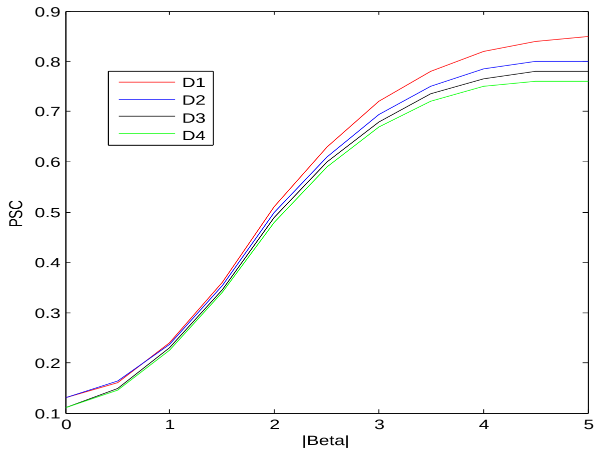


Figure 3.1.: 12 $\times$ 16 Optimal Designs Performance(q=3)

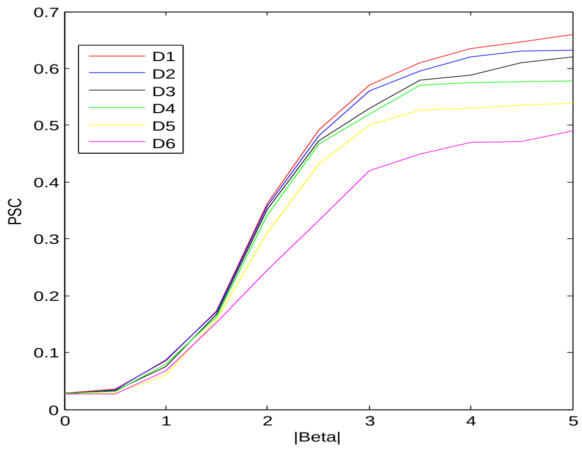


Figure 3.2.: 24 × 46 Optimal Designs Performance(q=5)

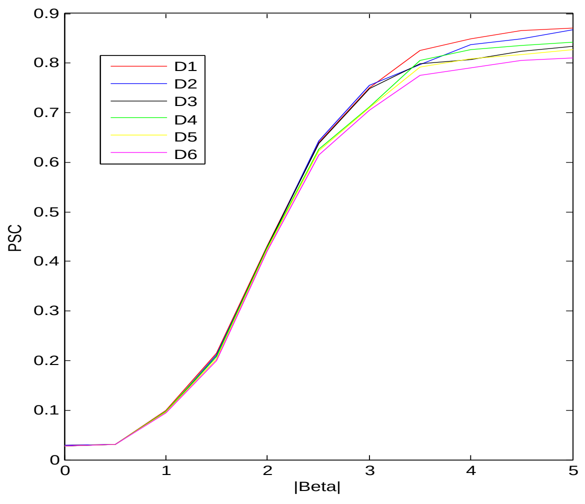


Figure 3.3.: 48 × 94 Optimal Designs Performance(q=7)

4. CASE STUDY

To illustrate the potential benefits of this screening procedure, we apply it to study an agent-based model of maritime escort operations. Notice the content in this chapter has been published in our paper [51]. A brief summary of the context follows. For further details, see [53]: which is part of a multinational, multi-year effort to leverage modeling and simulation in order to better understand trade-offs between cost and operational effectiveness for surface ships, and so improve the ship design process.

4.1 Motivation

In the past two decades, there has been a significant shift in navy missions towards operations other than war. Counter-piracy, search and rescue, maritime interdiction, maritime patrol, naval escort operations, and humanitarian efforts are the main focus of most fleets today, but the vessels that are currently being used in such operations were mainly built for other purposes. For instance, on 17 August 2009, the North Atlantic Council approved "Operation Ocean Shield" to fight piracy in the Gulf of Aden. Among six surface ships that were assigned in January-June 2012 rotation of this NATO mission, one was a destroyer and three were frigates ("NATO - Counter-piracy operations," n.d.). These are sophisticated warships which are capable of anti-surface warfare, anti-air warfare, and anti-submarine warfare. Although these sophisticated multi-mission capable fleets are able to achieve good results in expeditionary warfare against a strong enemy (Murphy 2007), their full capabilities will probably be used in less than 1% of their total life time. Frigates and destroyers are also expensive to build and operate.

Smaller combatants are much cheaper and better suited for the more common modern naval operations due to their flexibility. Therefore, many nations are reshaping their fleets to meet emerging operational demands, by starting to build multi-mission capable combatants such as Offshore Patrol Vessels (OPVs) that are smaller and less sophisticated than frigates and destroyers. These nations are also developing new tactics and counter measures to better deal with the new threats. The main purpose of OPVs is maintaining maritime security inshore and offshore, and they can be deployed globally [54]. Being cheaper yet flexible, OPVs are also great options for those countries that either do not really need or cannot afford sophisticated naval combatants [55].

Unfortunately, the ship design and the acquisition process has not kept pace with the rapid technological advances of the past few decades [56]. Historical cost estimation models, such as those based on cost per ton, often fail to adequately capture new technology-driven requirements. In the past, operational requirements were often not considered until late in the ship design process, and then might be based on limited input by small panels of subject matter experts. Cost effectiveness and operational effectiveness are both important, and it is extremely hard to achieve both when using a traditional ship design process. Advances in modeling and simulation, engineering design software, and massive amounts of data now allow for a much broader and richer range of ship specifications and capabilities to be explored. Moreover, utilizing simulation and analytical models to build decision making tools will ensure collaboration between warfighters and engineers in the early stages of the process. Exploiting technology is paramount for accomplishing the navy objectives and increasing the effectiveness for both cost and operations [57].

4.2 Simulation Scenario Description

Even though the term ”maritime terrorism” has been in the literature for more than a few decades, it was not really spelled out clearly until a series of incidents started in 2000. In January 2000, terrorists tried to attack USS The Sullivans (DDG-68) in Yemen. They failed because the terrorist boat sank right before the attack.

Following this event, the terrorists attacked USS Cole (DDG-67) with a suicide boat, killing 17 of the crew members, in October 2000. Almost two years after these incidents, in October 2002, a boat with explosives hit the French oil tanker Limburg which was close to Yemen coastal waters [58]. These are just a few examples of how, with just a small boat, terrorists can cause damage to multi-million/billion dollar ships, and—more importantly—kill innocent people. Many other terrorist activities which were discovered in their planning phases and prevented. These incidents show that maritime terrorism is a serious problem, and that surface combatants must be ready to fight terrorists at sea.

In June 2002 a group of terrorists, who were planning an attack on two merchant vessels in the Strait of Gibraltar, were caught by Moroccan officials [59]. The fact that the officials arrested those terrorists does not mean that similar plans will not be put into practice by other groups. If a terrorist group fills a boat with explosives and approach a ship in Strait of Gibraltar, it might be really hard to identify that boat as a terrorist boat since many vessels are in the strait at any point in time.

Our specific scenario involves a naval escort mission scenario in an anti-surface warfare environment. It is based on the 2002 Morocco incident, and implemented in an agent-based simulation modeling platform called MANA, designed by the Operational Analysis personnel of the New Zealand Defence Technology Agency (DTA) [60]. The key attributes of MANA which make it a useful tool for military applications are the situational awareness of the agents, advanced communication capabilities within squads and with other agents, the interaction of entities with friends and foes, and the user friendly design of the program. There are four main sets of parameters that form agent behaviors in MANA [61]:

- Personality weightings determine the willingness of agents to perform a particular action;
- Movement constraints modify the basic personality weightings of the agents;

- Intrinsic capabilities determine the physical characteristics of the agents such as sensors, weapons, or fuel level; and
- Movement characteristic adjustments ensure that agent actions change in different terrain conditions and different situations.

Just recently, DTA released MANA-V. The “V” stands for both vector and five. In this version, the programmers replaced the cell-based movement of previous versions with vector based movement. This allows building larger battlefield regions with panning and zooming options, and allows defining users to define distances, and the attributes such as speed and range, in real-world units rather than pixel-based units [62].

There are six types of agents in the scenario: the high value unit (HVU) being escorted, the OPV, the OPV’s helicopter, terrorist boats, known vessels, and unknown vessels. A screenshot from MANA appears in Figure 4.1; note that the sizes of the agents are not to scale, but are magnified for easier visualization by the user. The allegiance of the HVU, OPV, and the helicopter is “friend,” the allegiance of the terrorist boats is “hostile,” and the allegiance of the known and unknown vessels is “neutral.” The OPV’s mission is to escort the HVU through the Strait of Gibraltar and protect the HVU from attacks that can occur in the passage. The OPV has several guns, ranging from high caliber guns to machine guns. It also has a helicopter landing platform. The main role of the helicopter is to detect and classify unknown vessels. Its high speed and maneuverability give the friendly forces an advantage against the hostiles. The helicopter can also be equipped with a machine gun so that, if necessary, it can start firing before the hostile vessels come too close to the HVU.

The terrorist boats are loaded with explosives, and their purpose is to get close enough to the HVU to perform a suicide attack. The only target of the hostile boats is the HVU. They do not attack the OPV, but in some cases they may try to evade it. One of the critical properties of the terrorist boats is that they are initially acquired as unknown vessels in the friendly force radar systems until they are classified as

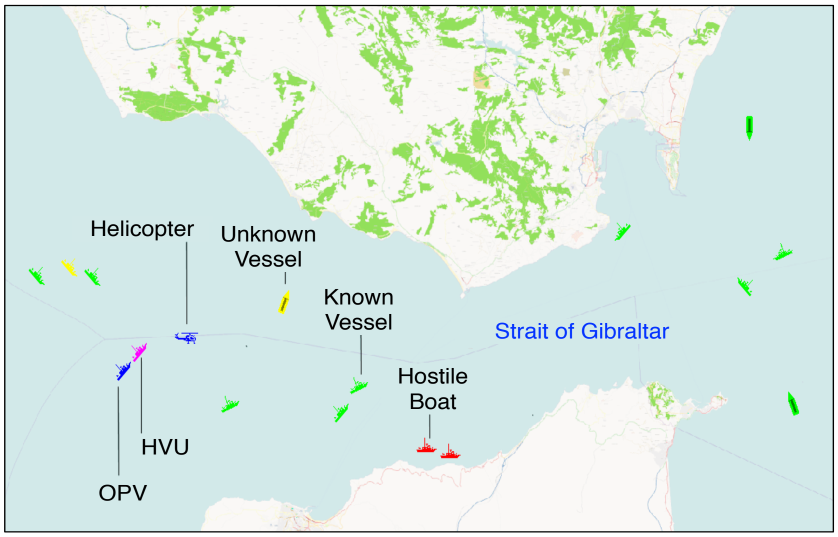


Figure 4.1.: MANA screenshot, showing six types of agents in the Strait of Gibraltar (from Kaymal 2013)

enemy. Therefore, either the helicopter classifies them as enemy, or they get closer to the OPV or HVU, and the friendly ships classify them depending on the range.

The known ships and the unknown ships are both neutral and they do not pose a threat to friendly assets. The only difference between the two is that while the OPV can instantly classify the known neutral ships using an Automatic Identification System (AIS) device, unknown ships cannot be classified when they are initially detected. This occurs for several reasons: some ships may be too small to carry an AIS device, others may have a device that is not working or turned off. This represents the type of marine traffic often found in the Strait of Gibraltar and other similar situations. It complicates matters for the OPV, since the OPV and its helicopter may be unable to fire on hostile boats if neutral ships are close by. Moreover, the OPV and helicopter may need to spend a substantial amount of time and effort identifying unknown ships, even though the vast majority are neutral. These distractions might give terrorists a better chance to approach the HVU without being classified as threats.

The simulation begins with the HVU and OPV approaching the strait. It halts the instant that either the HVU successfully navigates the strait, or it is attacked by a hostile ship.

4.3 Simulation Experiments

We conducted multiple sets of experiments in the process of creating and exploring this simulation model. Those for exploring the trade-offs used nearly orthogonal and balanced mixed designs [63], which are space-filling designs capable of handling a mix of continuous, discrete, and qualitative factors. The initial experiment involved 100 replications of a NOAB design for 38 factors, including one that represented the time step used in the scenario. This directly affects the time it takes to complete the simulation runs, but the time step must be small to avoid model artifacts (Al Rouawaii and Buss ref). Once the choice of time step was settled and some other minor changes were made, the final space-filling experiment involved 200 replications of a NOAB for 35 factors. Consolidating these results into a single file for analysis resulted in 153,600 simulation runs. Kaymal [53] uses a range of statistical techniques, including regression, logistic regression, and partition trees, to model the relationship between the factors and key responses. These models also are used to suggest factor ranges and combinations that make efficient use of resources but achieve high operational effectiveness.

In this paper, we focus on just one of the many potential output measures. Our response Y is the proportion of time that the HVU successfully navigates the strait. We use a 24×69 Lasso-optimal design capable of examining up to 69 factors, and identifying up to 24 important ones. The design is available on request. We augmented Kaymal's list of 35 factors with an additional 30 that had not initially been explored. To maintain the spirit of a screening experiment, we made very tiny changes in factor settings for personality weights and communication characteristics (e.g., ± 2 on a scale of -100 to 100). In addition to serving as an example for the Lasso screen-

Table 4.1: Numerical Evaluation of 24 $\times$ 69 Lasso-optimal design

q	Important Factors	SVOE Lasso Variable Selection
5	**4,5,32,50,56**	(**56**,**50**,**4**,**32**,**5**)
10	**3**,**4**,**6**,**17**,23,**25**,26,**31**,**37**,63	(**37**,**4**,**31**),(**37**,**4**,**31**,**3**),(**37**,**4**,**31**,**3**,59,25,**17**,**6**)
15	**5**,8,9,20,25,**26**,29,**33**,**37**, 44,45,**46**,47,60,63	(3,**37**,39,**46**,**33**,**5**,52,24,48,58,**26**)
20	2,8,9,12,21,**26**,32,34,**39**,40, 46,47,**52**,53,55,57,**58**,62,64,65	(**39**,38,**52**,**26**,**58**)

ing approach, we viewed this as an opportunity to beta-test some aspects of MANA that were not used in Kaymal's experiments. (Over the years, the developers have added many features to MANA in response to requests by NPS students. In turn, the beta-testing by NPS students of MANA's features has been useful for verification and validation efforts.)

To test the performance of the selected design, we ran a quick numerical evaluation of the selected design. Consider a linear model with 65 factors and $q = 5, 10, 15,$ and 20 respectively. The important factors are randomly selected and their effects are set at 3. The other factors all have an effect of 0. The white noise follows a standard normal distribution. We generate one set of responses of the 24 $\times$ 69 Lasso-optimal design for each case, and the results are summarized in Table 4.1. All factors that are both truly important and selected by Lasso are shown in boldface. Note that for the $q = 10$ case, there are multiple optimal λs, therefore multiple models.

Ideally, the SVOE Lasso method would select all of the important factors and none of the unimportant ones. It does this for our $q = 5$ example. As q increases, the number of selected factors remains roughly constant, which means there is a decrease in the proportion of correctly identified factors. Some unimportant factors also show up, although false positives are less of a concern in screening experiments because they can be eliminated in subsequent stages. Clearly, a much more extensive empirical investigation would be needed to provide a true picture of the design's screening capabilities, both in terms of false positives and false negatives. Nonetheless, it is promising to see that when the number of important factors is small or moderate ($q =$ 5 or 10), the SVOE Lasso method selects a majority or all of the important factors with less than half number of runs required by saturated design. This implies that for practical screening problems where a small set of factors can drive the response, SVOE Lasso may work well.

With these promising (though limited) results giving us some confidence of the design, we ran 1000 replications to obtain sample proportions Y_i for each of the 24 design points. This took over 5.25 days of CPU time on our computing cluster. We found that seven of the 24 design points aborted before all replications were complete. After defining a new response $\tilde{Y} \equiv \mathrm{I}(\text{all reps complete})$, we quickly identified which factor was causing the problem. After confirming our findings by running MANA in its GUI mode, we removed this factor from our list and substituted another. This points out a clear benefit of using a screening procedure. We were able to identify a bug in just 5.25 CPU days, rather than the 112 CPU days required for 1000 replications of our 512 design point NOAB, or the 4.9 CPU years required for 1000 replications of an 8196 design point resolution V fractional factorial [64].

Our second screening experiment ran to completion. There were two potential choices for λ_{opt}. The smallest selects two of the first 35 factors and three of the second 30, the largest selects six of the first 35 factors and five of the second 30. The Lasso path for $40 \leq \lambda \leq 220$ appears in Figure 4.2.

These results were somewhat surprising, since we had anticipated that all the important factors would be among the first 35. Further experimentation is underway to determine an appropriate baseline for the study. Whatever these results, they will be beneficial for several stakeholders. As mentioned earlier, the MANA developers have been adding features to expand its capabilities, and they are always interested in results of large-scale beta tests that might indicate either model bugs

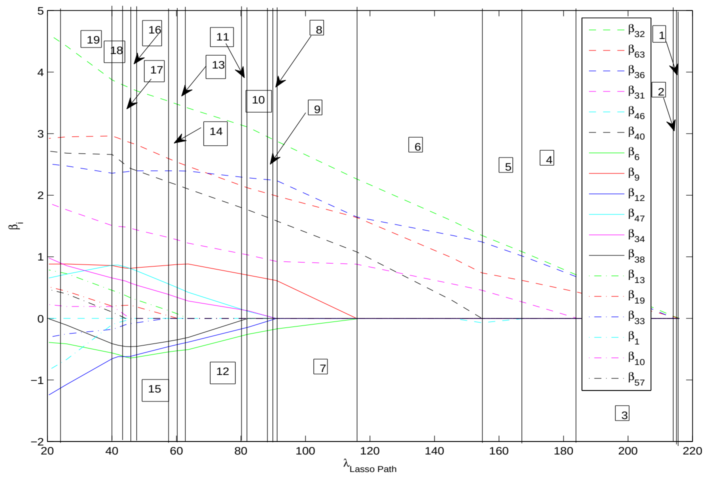


Figure 4.2.: Lasso Path for the 24 65 design

("artifacts"), or else modeling concepts that would benefit from additional clarification in the documentation. Those creating scenarios can use screening experiments early in the model-building process to aid in model verification and help set suitable factor ranges. The international consortium looking at the model-based ship design project will benefit from having information about the robustness of the findings—we advocate that it is better to experiment on a large number of factors and seek a simplified model than to limit the factors from the outset [65]. Finally, by examining simulation models that are being used for decision-making, we will continue to identify challenges that would benefit from further methodological research.

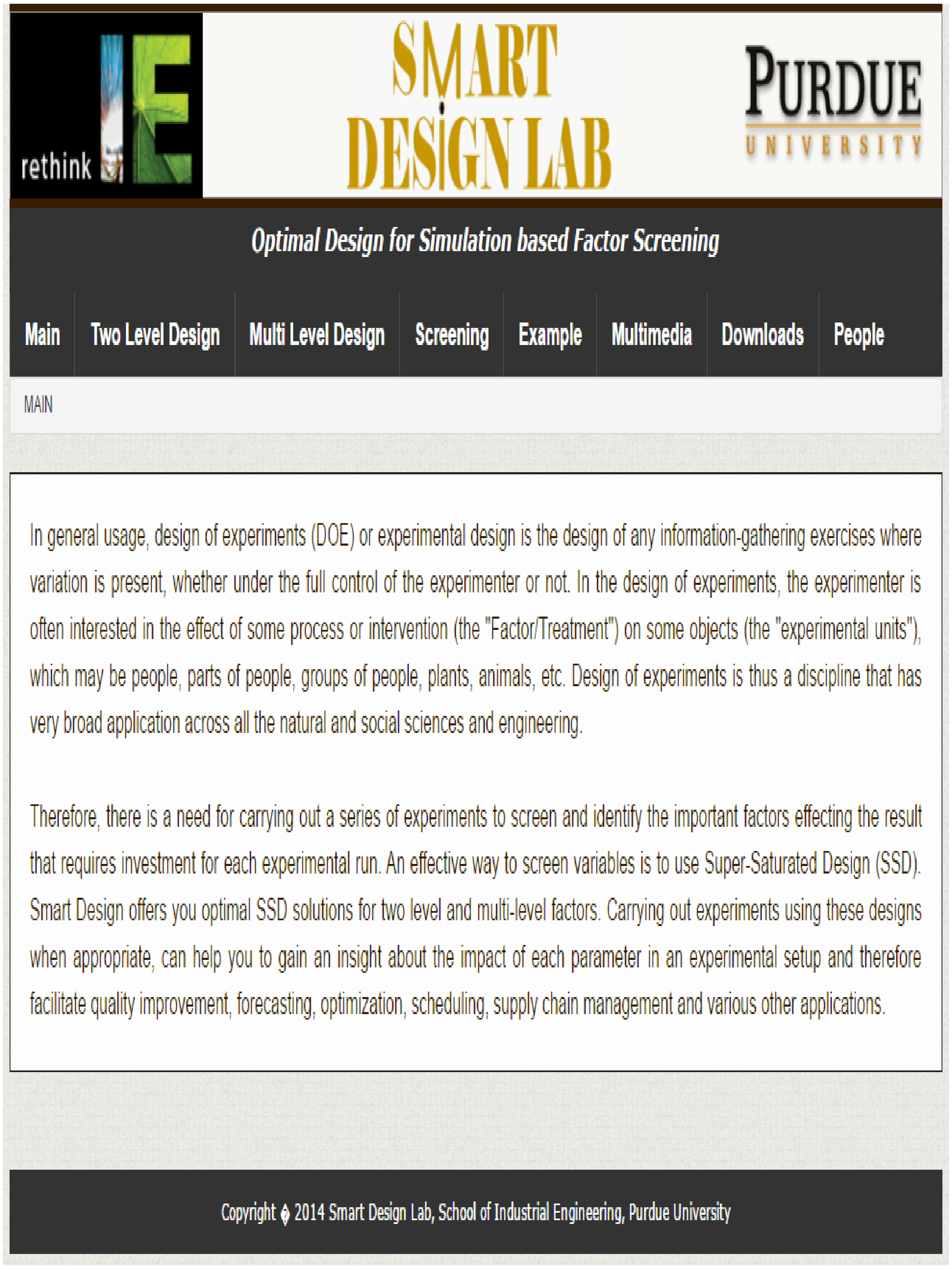
5. WEBSITE OF THE SMART OPTIMAL DESIGN FOR SIMULATION

To facilitate the application of our methods and make it more convenient for academic and industry applications, we created "The Smart Design for Simulation" website ("https://engineering.purdue.edu/smartdesigns") on the cluster of Industrial Engineering department of Purdue University. The website provide an interface for people to generate optimal SSDs using the criteria and algorithm discussed in previous chapters. Given any combination of the run size n and the number of factors p, the Matlab package running on the cluster usually generates the designs within several seconds or minutes, depends on the given n and p.

The main page (Figure 5.1) describes the purpose of the website. By click the tap "Two Level Design" and "Multiple Level Design" on the left top of the main page, users can select the levels of the designs they want to generate (Figure 5.2). As mentioned previous, we also extended our methods into the multiple level designs as well. User can just type in the dimensions (n and p) and select the criterion they prefer and then click the button "Generate" (Figure 5.3). The Matlab package running on the cluster will provide the optimal SSDs for given n and p (Figure 5.4).

After the generated optimal designs have been applied, users can also use this website to analyze the results. As shown in Figure 5.6, users can upload the design matrix and responses from an excel or text file, then select the model selection method they prefer. Besides the AIC-Lasso, BIC-Lasso and CV-Lasso, We also provide the SV-Lasso (Chapter 2.3) as an option to do the model selection. We expect the SV-Lasso can help people to achieve better model selection results comparing to the traditional methods.

We also provide an example on the website (click the tab "Example" as shown in Figure 5.7 and Figure 5.8) to show how the website works and a link to download our packages and data generated.

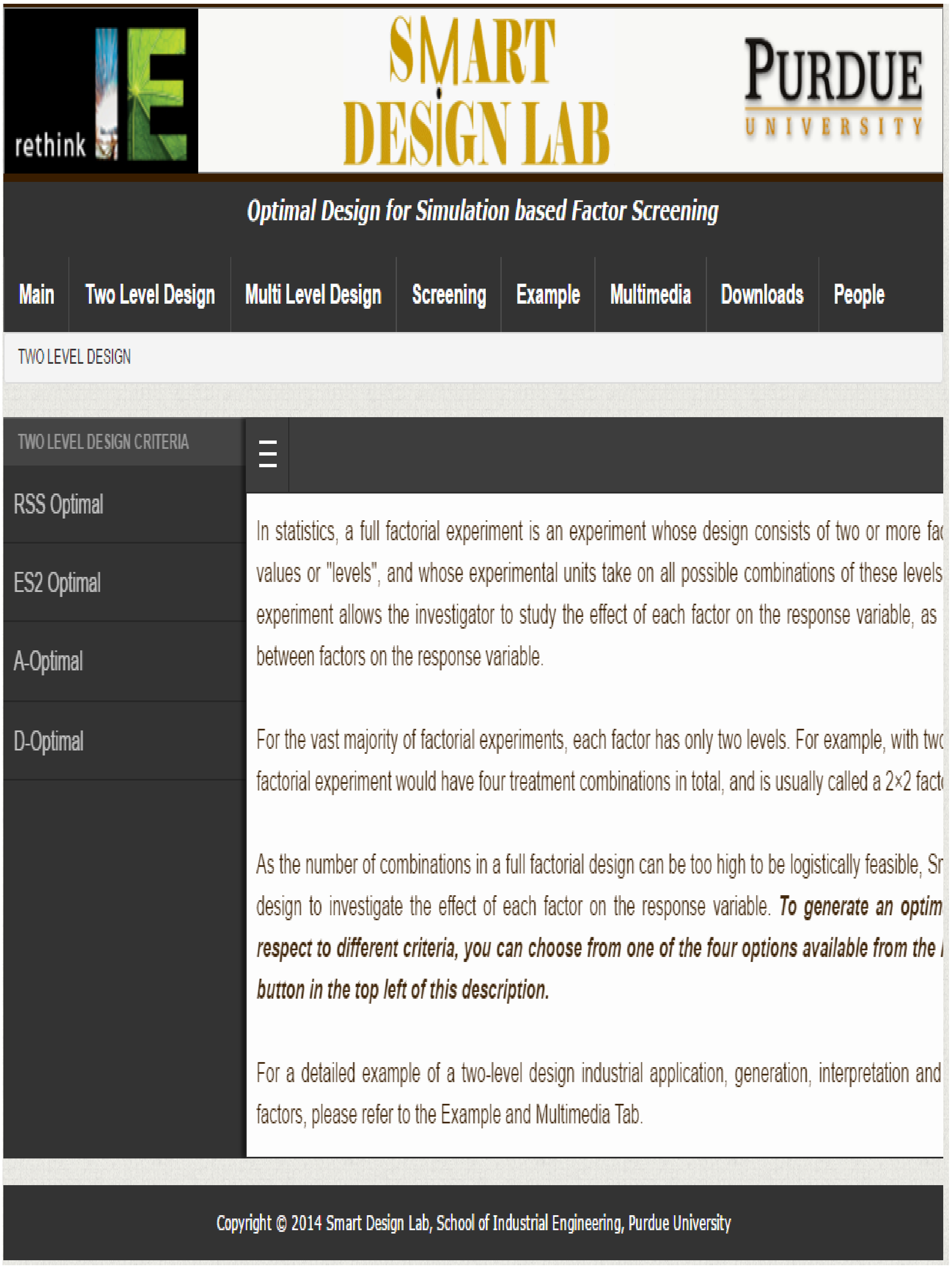


In general usage, design of experiments (DOE) or experimental design is the design of any information-gathering exercises where variation is present, whether under the full control of the experimenter or not. In the design of experiments, the experimenter is often interested in the effect of some process or intervention (the "Factor/Treatment") on some objects (the "experimental units"), which may be people, parts of people, groups of people, plants, animals, etc. Design of experiments is thus a discipline that has very broad application across all the natural and social sciences and engineering.

Therefore, there is a need for carrying out a series of experiments to screen and identify the important factors effecting the result that requires investment for each experimental run. An effective way to screen variables is to use Super-Saturated Design (SSD). Smart Design offers you optimal SSD solutions for two level and multi-level factors. Carrying out experiments using these designs when appropriate, can help you to gain an insight about the impact of each parameter in an experimental setup and therefore facilitate quality improvement, forecasting, optimization, scheduling, supply chain management and various other applications.

Copyright � 2014 Smart Design Lab, School of Industrial Engineering, Purdue University

Figure 5.1.: Main Page



In statistics, a full factorial experiment is an experiment whose design consists of two or more fac
values or "levels", and whose experimental units take on all possible combinations of these levels
experiment allows the investigator to study the effect of each factor on the response variable, as
between factors on the response variable.

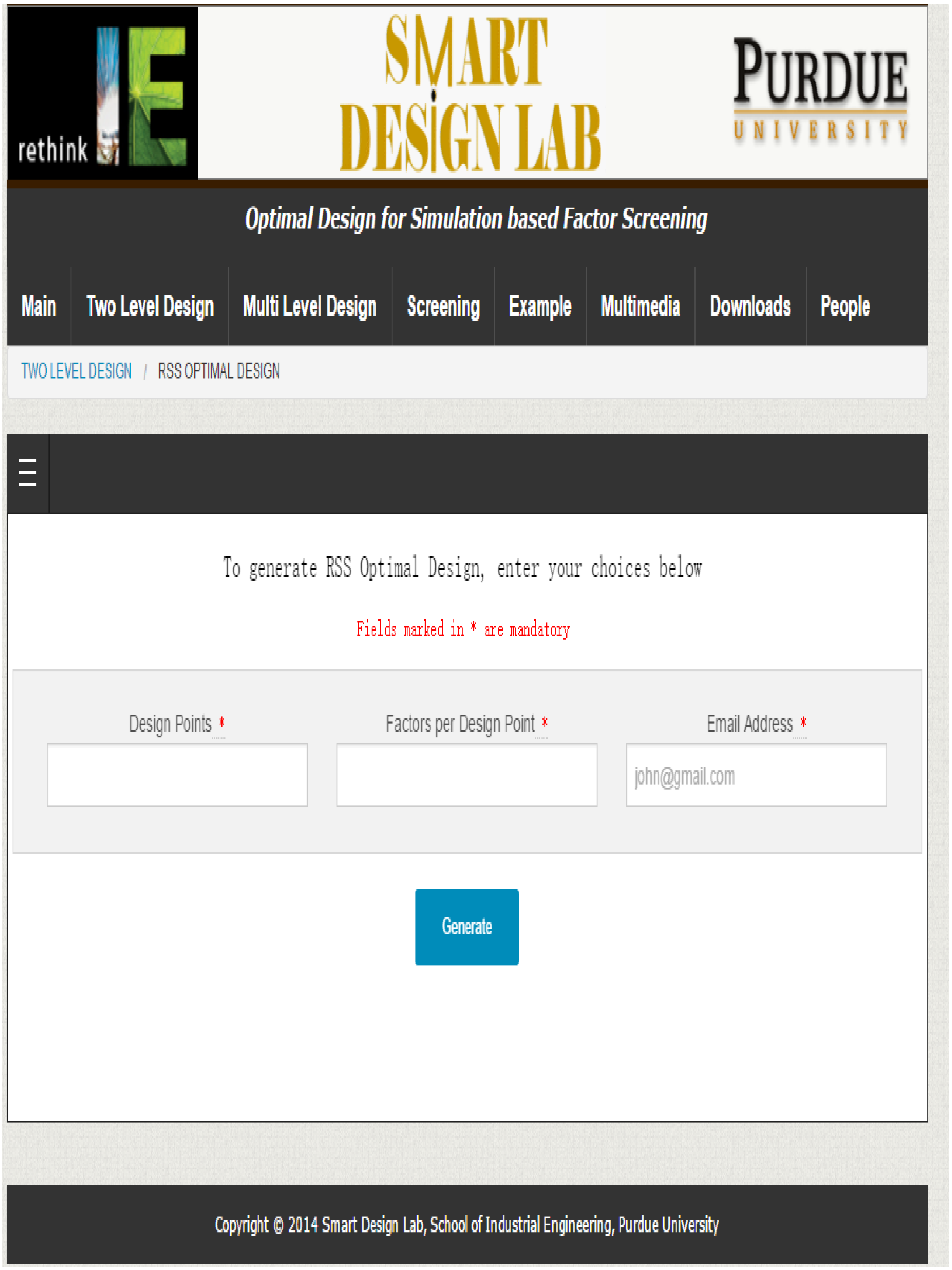
For the vast majority of factorial experiments, each factor has only two levels. For example, with twc
factorial experiment would have four treatment combinations in total, and is usually called a 2×2 facto

As the number of combinations in a full factorial design can be too high to be logistically feasible, Sr
design to investigate the effect of each factor on the response variable. ***To generate an optim***
respect to different criteria, you can choose from one of the four options available from the
button in the top left of this description.

For a detailed example of a two-level design industrial application, generation, interpretation and
factors, please refer to the Example and Multimedia Tab.

Copyright © 2014 Smart Design Lab, School of Industrial Engineering, Purdue University

Figure 5.2.: Two Level Designs



Optimal Design for Simulation based Factor Screening

TWO LEVEL DESIGN / RSS OPTIMAL DESIGN

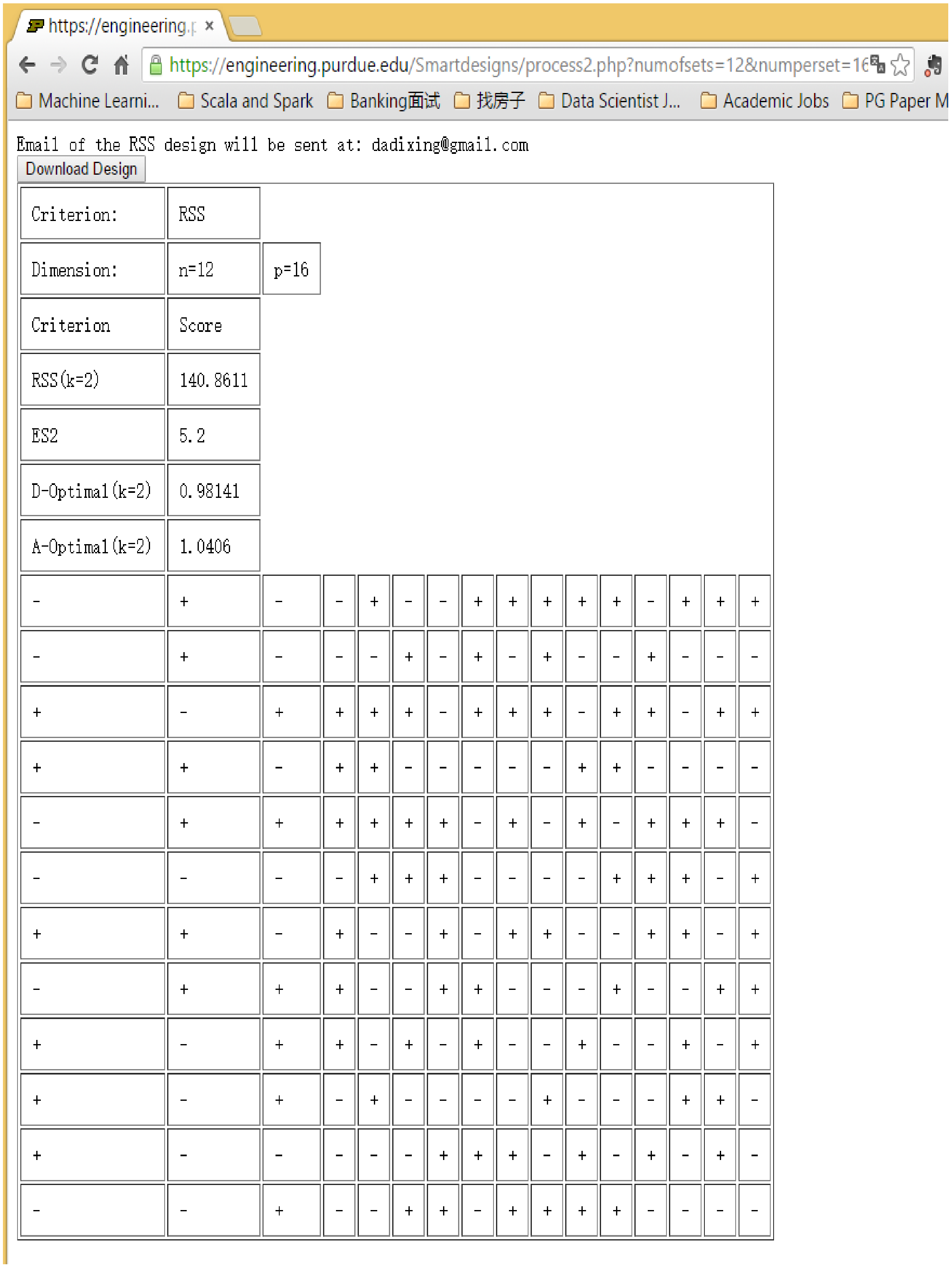
To generate RSS Optimal Design, enter your choices below

Fields marked in * are mandatory

Design Points *	Factors per Design Point *	Email Address *
		john@gmail.com

Copyright © 2014 Smart Design Lab, School of Industrial Engineering, Purdue University

Figure 5.3.: Two Level Designs Input

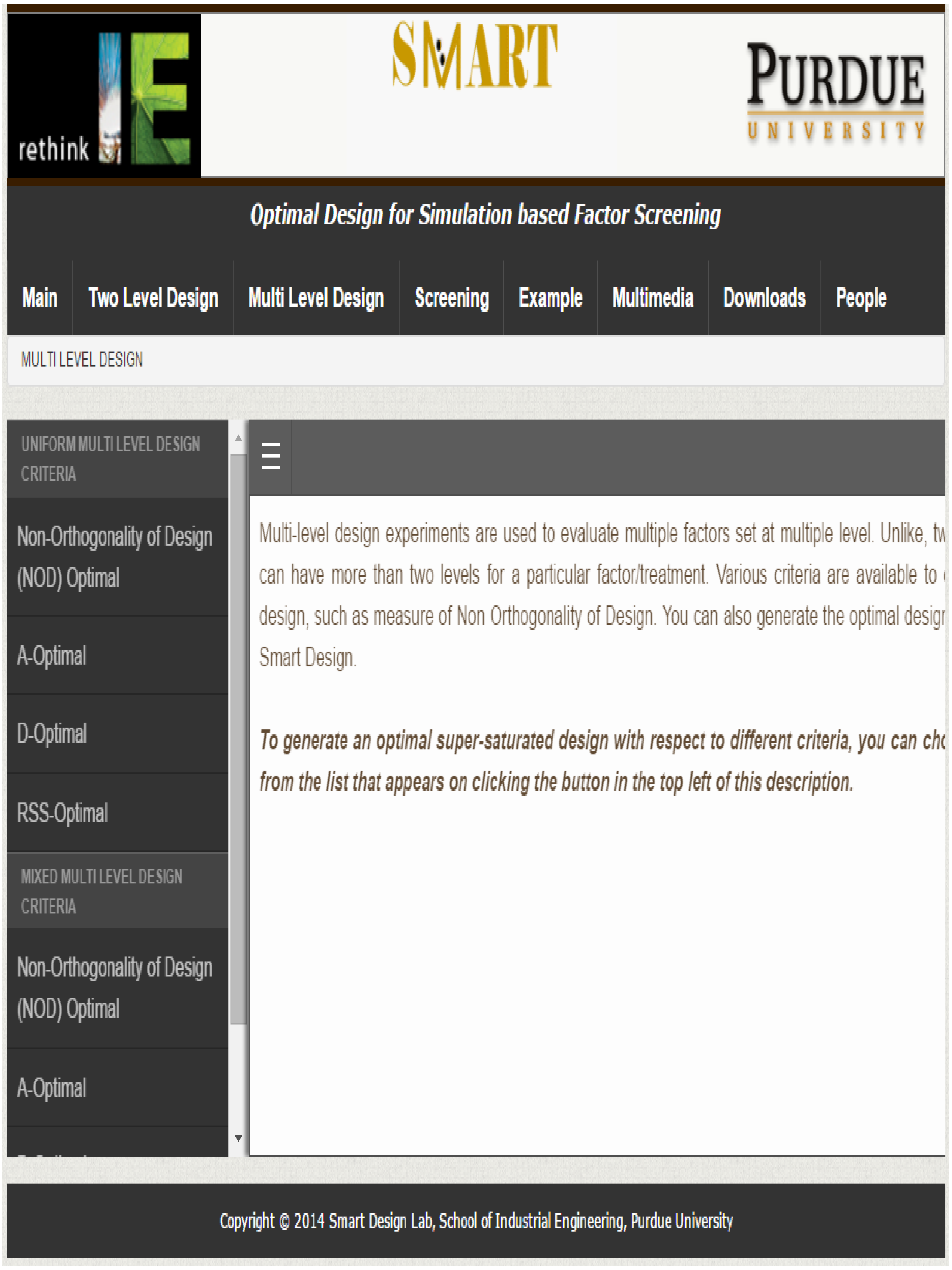


Email of the RSS design will be sent at: dadixing@gmail.com

Download Design

Criterion:	RSS														
Dimension:	n=12	p=16													
Criterion	Score														
RSS(k=2)	140.8611														
ES2	5.2														
D-Optimal(k=2)	0.98141														
A-Optimal(k=2)	1.0406														
-	+	-	-	+	-	-	+	+	+	+	+	-	+	+	+
-	+	-	-	-	+	-	+	-	+	-	-	+	-	-	-
+	-	+	+	+	+	-	+	+	+	-	+	+	-	+	+
+	+	-	+	+	-	-	-	-	-	+	+	-	-	-	-
-	+	+	+	+	+	+	-	+	-	+	-	+	+	+	-
-	-	-	-	+	+	+	-	-	-	-	+	+	+	-	+
+	+	-	+	-	-	+	-	+	+	-	-	+	+	-	+
-	+	+	+	-	-	+	+	-	-	-	+	-	-	+	+
+	-	+	+	-	+	-	+	-	-	+	-	-	+	-	+
+	-	+	-	+	-	-	-	-	+	-	-	-	+	+	-
+	-	-	-	-	-	+	+	+	-	+	-	+	-	+	-
-	-	+	-	-	+	+	-	+	+	+	+	-	-	-	-

Figure 5.4.: Two Level Designs Output

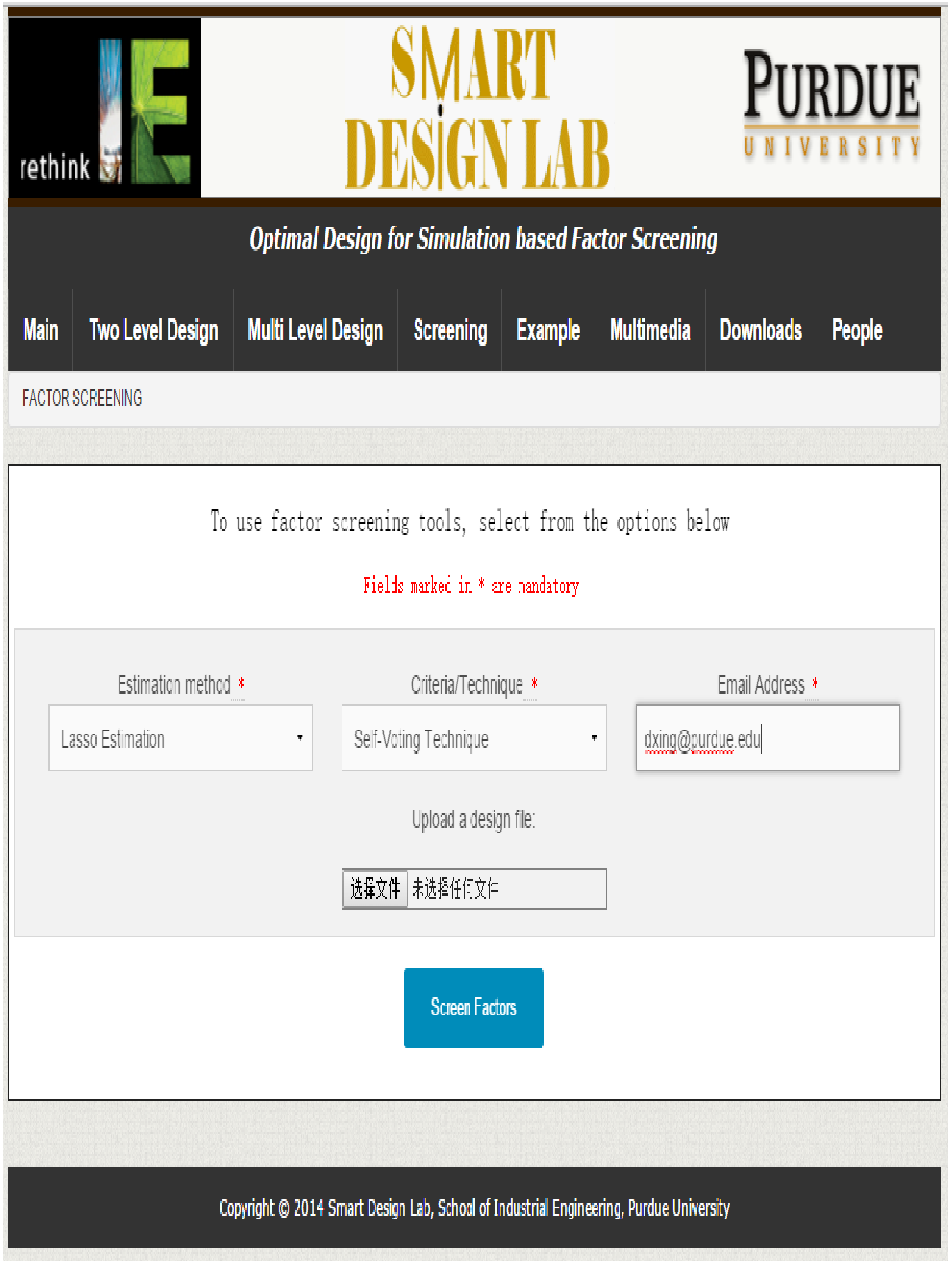

rethink IE
SMART
PURDUE UNIVERSITY
Optimal Design for Simulation based Factor Screening
Main
Two Level Design
Multi Level Design
Screening
Example
Multimedia
Downloads
People
MULTI LEVEL DESIGN
UNIFORM MULTI LEVEL DESIGN CRITERIA
Non-Orthogonality of Design (NOD) Optimal
A-Optimal
D-Optimal
RSS-Optimal
MIXED MULTI LEVEL DESIGN CRITERIA
Non-Orthogonality of Design (NOD) Optimal
A-Optimal

Multi-level design experiments are used to evaluate multiple factors set at multiple level. Unlike, tw
can have more than two levels for a particular factor/treatment. Various criteria are available to
design, such as measure of Non Orthogonality of Design. You can also generate the optimal desig
Smart Design.

*To generate an optimal super-saturated design with respect to different criteria, you can ch
from the list that appears on clicking the button in the top left of this description.*

Copyright © 2014 Smart Design Lab, School of Industrial Engineering, Purdue University

Figure 5.5.: Multiple Level Designs



To use factor screening tools, select from the options below

Fields marked in * are mandatory

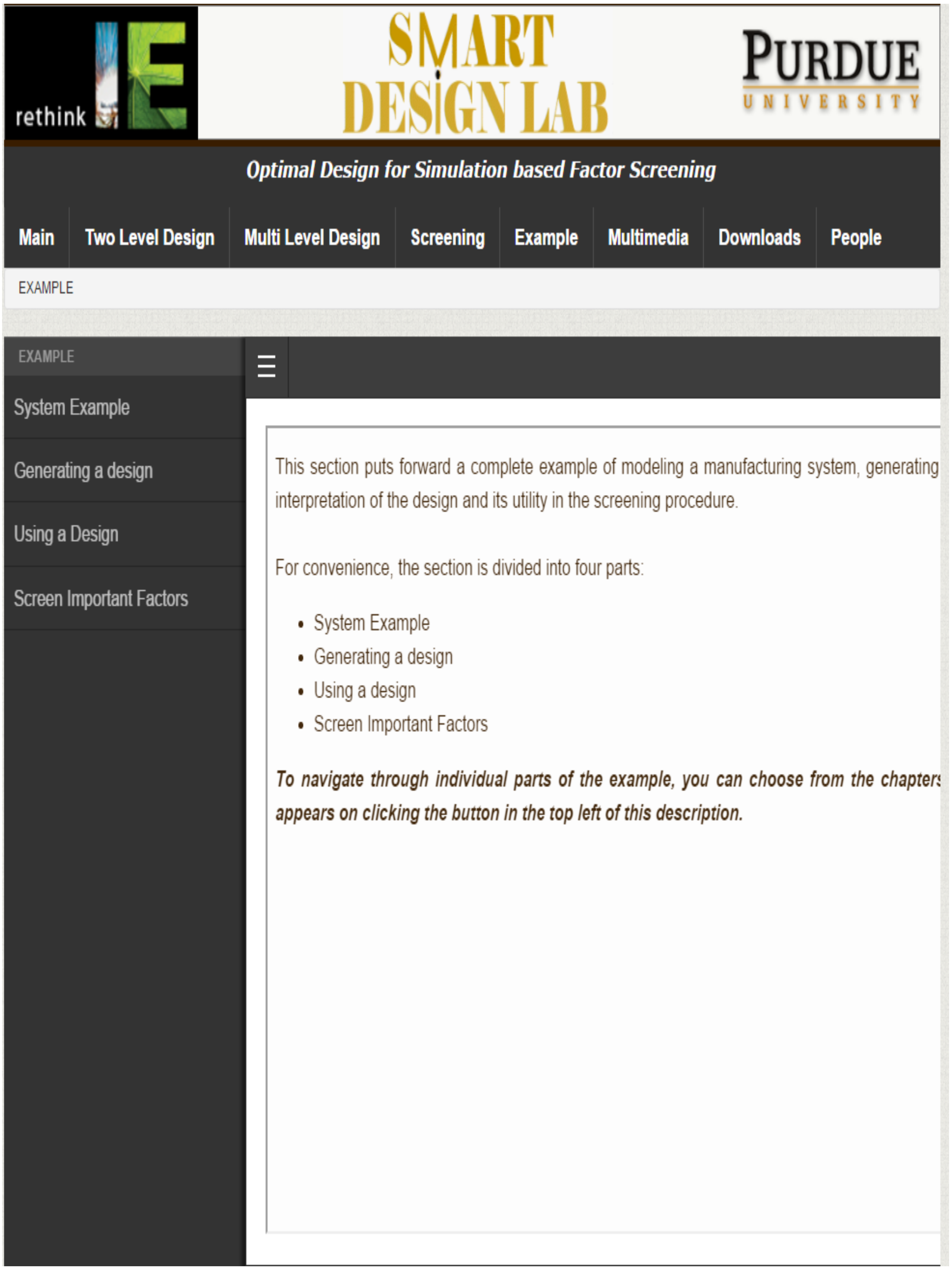
Estimation method *	Criteria/Technique *	Email Address *
Lasso Estimation	Self-Voting Technique	dxing@purdue.edu

Upload a design file:

选择文件 未选择任何文件

Copyright © 2014 Smart Design Lab, School of Industrial Engineering, Purdue University

Figure 5.6.: Screening



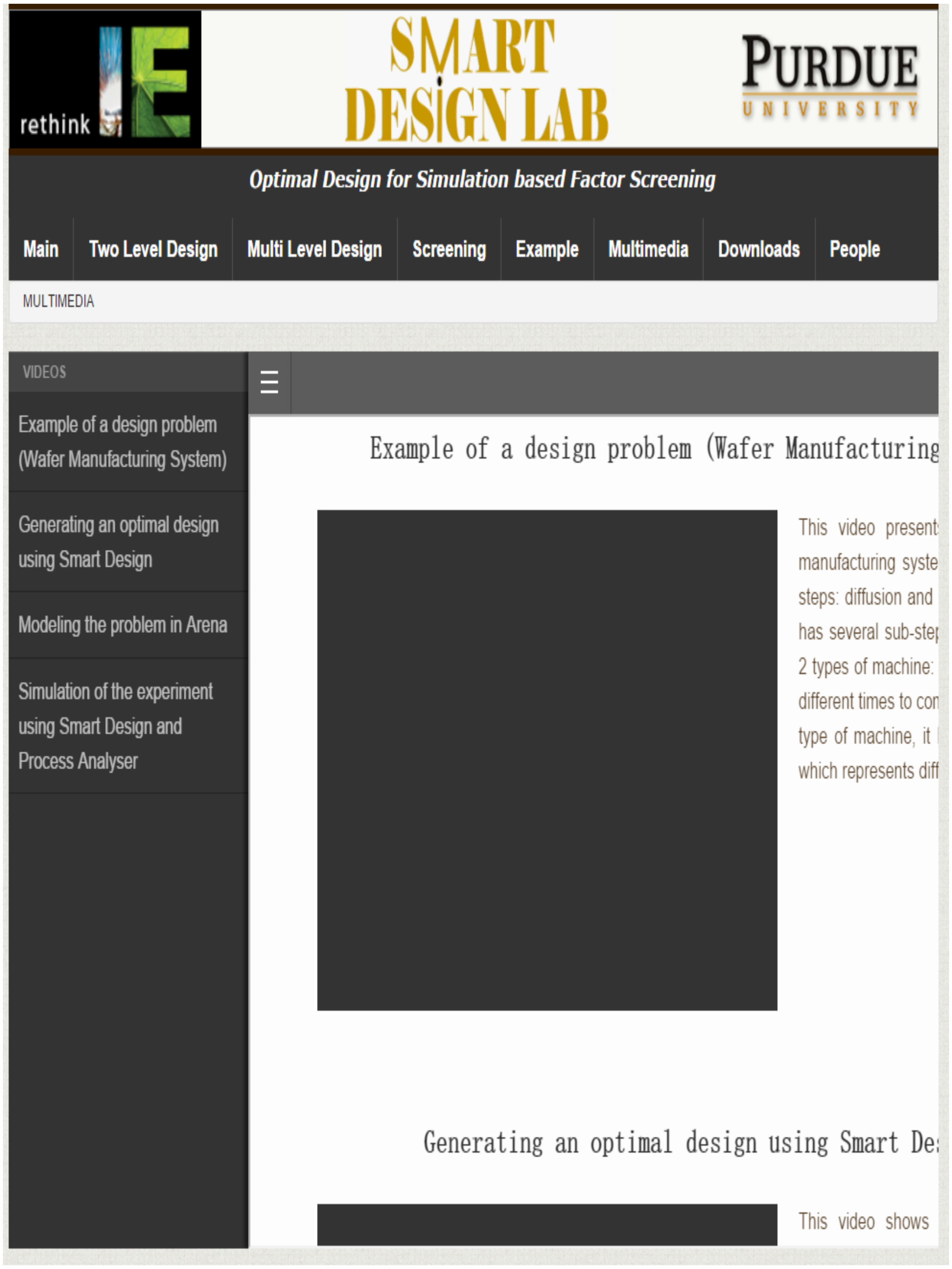
This section puts forward a complete example of modeling a manufacturing system, generating
interpretation of the design and its utility in the screening procedure.

For convenience, the section is divided into four parts:

- System Example
- Generating a design
- Using a design
- Screen Important Factors

To navigate through individual parts of the example, you can choose from the chapters
appears on clicking the button in the top left of this description.

Figure 5.7.: Example


rethink IE
SMART DESIGN LAB
PURDUE UNIVERSITY
Optimal Design for Simulation based Factor Screening
Main
Two Level Design
Multi Level Design
Screening
Example
Multimedia
Downloads
People
MULTIMEDIA
VIDEOS
Example of a design problem (Wafer Manufacturing System)
Generating an optimal design using Smart Design
Modeling the problem in Arena
Simulation of the experiment using Smart Design and Process Analyser

Example of a design problem (Wafer Manufacturing

This video present manufacturing syste steps: diffusion and has several sub-ste 2 types of machine: different times to cor type of machine, it which represents diff

Generating an optimal design using Smart De

This video shows

Figure 5.8.: Multipmedia

6. CONCLUDING REMARKS AND FUTURE WORK

We have developed a general framework with effective statistical methods for design and analysis of screening experiments to bridge the aforementioned gap. We first studied the variable selection performances of the lasso under finite samples and establish the exact relationship between their performances and design matrices. We also discussed the approximations of the lower bounds of the probabilities for the lasso selecting the true model or variables, and developed Monte Carlo procedures for them. Using the probabilities, we developed a data driven iterative procedure to select λ. Simulations show it outperform other existing methods.

Based on that study, we have proposed optimality criteria for constructing supersaturated designs that guarantee optimal variable selection performance, and studied the theoretical properties of the proposed criteria. Furthermore, we develop an efficient partial gradient algorithm and apply it to construct optimal $\overline{\mathrm{RSS}}_q$ SSDs. The numerical results demonstrate the great potential of our methods.

Our research will encourage the practice of design of experiments in study of complex systems. Specifically, the design will be used in large-scale simulation models. The resulting efficiency improvement of simulation experiments will impact research and applications of various science and engineering areas and facilitate the decision-making process.

For the future research, it is possible to extent our work in three directions. First, the iterative procedure of the tuning parameter selection demonstrates excellent performance empirically, that is, it selects λ close to the optimal lambda, but we have not established the result theoretically. We expect that the procedure produces a good estimate of the optimal lambda. Second, we are applying the finite samples analysis on other variants or improvements of the lasso that use more sophisticated penalty function such as the LLA [66]. Thirdly, the idea that generating designs based on optimizing the variable selection performance can be extended to other L_1-based shrinkage methods beyond Lasso to generate new and practically more attractive criteria, and the partial gradient algorithm shows significant improvement of the efficiency and can be used on other criteria.

APPENDICES

APPENDIX A
DESIGNS USED IN THE EXAMPLES

Design: Optimal 12×16 SSD based on $\mathrm{E}(s^2)$

−	−	+	+	+	+	−	−	−	+	+	−	+	+	−	+
−	+	−	+	+	+	−	+	+	−	−	+	−	+	+	+
+	−	+	−	+	−	−	+	+	−	−	−	+	−	+	+
−	+	+	+	+	−	+	+	−	+	−	+	+	−	−	−
−	+	+	−	−	+	−	−	+	+	+	+	+	−	+	−
+	+	−	+	−	−	+	−	+	−	+	+	+	−	−	+
+	−	+	+	−	−	−	+	−	−	+	+	−	+	+	−
+	−	+	−	−	+	+	−	+	+	−	+	−	+	−	+
−	+	−	−	−	−	+	+	−	+	+	−	−	+	+	+
+	−	−	+	+	+	+	+	+	+	+	−	−	−	−	−
+	+	−	−	+	+	+	−	−	−	−	−	+	+	+	−
−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−

Design: Optimal 12×22 SSD based on $\mathrm{E}(s^2)$

+	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	−	−	−	−	−
−	−	+	−	+	+	+	−	−	−	+	+	+	−	+	−	−	−	+	+	+	−
+	+	−	−	−	+	−	−	+	−	+	+	+	+	−	+	−	−	−	+	+	+
−	−	−	+	−	−	+	−	+	+	+	+	−	+	+	−	+	−	−	−	+	+
−	+	−	+	+	+	−	−	−	+	−	+	+	−	+	+	−	+	−	−	−	+
+	−	−	+	−	+	+	+	−	−	−	+	+	+	−	+	+	−	+	−	−	−
+	−	−	−	+	−	−	+	−	+	+	+	+	+	+	−	+	+	−	+	−	−
−	+	+	+	−	−	−	+	−	−	+	+	−	+	+	+	−	+	+	−	+	−
−	−	+	−	−	+	−	+	+	+	−	+	−	−	+	+	+	−	+	+	−	+
−	+	−	−	+	−	+	+	+	−	−	+	−	−	−	+	+	+	−	+	+	−
+	+	+	−	−	−	+	−	−	+	−	+	+	−	−	−	+	+	+	−	+	+
+	−	+	+	+	−	−	−	+	−	−	+	−	+	−	−	−	+	+	+	−	+

APPENDIX B PROOFS

Proof of Theorem 1

Because the lasso target function is convex, the lasso estimate by (2) must satisfy the KKT conditions as follows (Rosset and Zhu,2007):

(X'X- X'Y)$_j = -\frac{\lambda}{2} sgn(\hat{\beta}_j)$ for $\hat{\beta}_j \neq 0;$ (B.1)

$$-\frac{\lambda}{2} \leq (X'X\hat{\beta} - X'Y)_j \leq \frac{\lambda}{2} \quad \text{for } \hat{\beta}_j = 0; \tag{B.2}$$

Note that for the correct selection condition, $\hat{A} = A \iff \hat{\beta}_{\hat{C}} = \beta_C = 0$, put the latter into the KKT conditions, we can get:

$$X_A'X_A\hat{\beta}_A - X_A'Y = -\frac{\lambda}{2} sgn(\hat{\beta}_A) \tag{B.3}$$

$$-\frac{\lambda}{2} \leq X_C'X_A\hat{\beta}_A - X_C'Y \leq \frac{\lambda}{2} \tag{B.4}$$

where, X_A is the set of columns belong to A, X_C is the set of columns belong to C. By linear model, we have $Y = X_A\beta_A + \epsilon$, replace Y in the above formulations, they become:

$$\hat{\beta}_A - \beta_A = (X_A'X_A)^{-1}X_A'\epsilon - \frac{\lambda}{2}(X_A'X_A)^{-1}sgn(\beta_A) \tag{B.5}$$

$$-\frac{\lambda}{2} \leq X_C'X_A(\hat{\beta}_A - \beta_A) - X_C'\epsilon \leq \frac{\lambda}{2} \tag{B.6}$$

Replace $(\hat{\beta}_A - \beta_A)$ in (8) by the right side of (7), we have

$$-\frac{\lambda}{2}(1 + X_C'X_A(X_A'X_A)^{-1}sgn(\hat{\beta}_A)) \leq X_C'(I-P)\epsilon \leq \frac{\lambda}{2}(1 - X_C'X_A(X_A'X_A)^{-1}sgn(\hat{\beta}_A)) \tag{B.7}$$

where $P = X_A(X_A'X_A)^{-1}X_A'$, this is the sufficient condition for correct selecting variables, but the $sgn(\hat{\beta}_A)$ may be not correct with the original ones. To ensure the sign correctness, we have to add the sign condition into the formulations above. The correct sign condition $sgn(\hat{\beta}_A) = sgn(\beta_A)$ is equal with the following inequations:

$$\hat{\beta}_A. * \beta_A > 0; \tag{B.8}$$

where, ".$*$" means the production of corresponding elements in the vectors, which is still a vector. For example, $\hat{\beta}_{\hat{A}}.*\beta_A = diag(\hat{\beta}_{\hat{A}})diag(\beta_A)\tilde{1}$, here $\tilde{1} = (1,1,...1)'_q$, $q =\mid A \mid$ is the size of A.

Obviously, implementing any of them will all achieve the sufficient and necessary condition for sign correctness. Here, we use (10) for mathematical convenience.

"$.*\beta_A$" at the both sides of (7), we have

$$\hat{\beta}_A.*\beta_A = \beta_A.*\beta_A + (X_A'X_A)^{-1}X_A'\epsilon.*\beta_A - \frac{\lambda}{2}(X_A'X_A)^{-1}sgn(\hat{\beta}_A).*\beta_A \tag{B.9}$$

Define $\tau = (X_A'X_A)^{-1}X_A'\epsilon$ and $T = (X_A'X_A)^{-1}sgn(\beta_A)$, then it becomes,

$$\hat{\beta}_A.*\beta_A = \beta_A.*\beta_A + \tau.*\beta_A - \frac{\lambda}{2}T.*\beta_A \tag{B.10}$$

Implementing (10) and rewrite the right side of (13) as the element-wise form,

$$\beta_{iA}^2 + \tau_{iA}\beta_{iA} - \frac{\lambda}{2}T_{iA}.*\beta_{iA} > 0 \tag{B.11}$$

where the index iA refers to ith element of the corresponding vector which belongs to A. There are two roots across the X-axis, 0 and $\frac{\lambda}{2}T_{iA} - \tau_{iA}$. According to the properties of concave function, the inequation satisfied when the following stands,

$$\beta_{iA} \in [(-\infty, \min(0, \frac{\lambda}{2}T_{iA} - \tau_{iA})) \cup (\max(0, \frac{\lambda}{2}T_{iA} - \tau_{iA}), \infty)] \tag{B.12}$$

Clearly, (14) could guarantee β_{iA} has the correct sign, the conjunction of all the entries of β which satisfy (14) makes the sufficient and necessary sign correctness condition of the lasso:

$$\bigcap_i [\beta_{iA} \in [(-\infty, \min(0, \frac{\lambda}{2}T_{iA} - \tau_{iA})) \cup (\max(0, \frac{\lambda}{2}T_{iA} - \tau_{iA}), \infty)]] \tag{B.13}$$

Note that ϵ follows $N(0, \sigma^2 I_n)$, the equality holds when ϵ is normal. The constrains turn out to be some probabilistic events determined by a set of normal distribution. Define $R = X_C'X_A(X_A'X_A)^{-1}sgn(\beta_A)$, the two events can be written as

$$\mathcal{E}_1 = \{-\frac{\lambda}{2}(\mathbf{1} + R) \le X_C'(I-P)\epsilon \le \frac{\lambda}{2}(\mathbf{1} - R)\} \tag{B.14}$$

$$\mathcal{E}_2 = \{\bigcap_i [\beta_{iA} \in [(-\infty, \min(0, \frac{\lambda}{2}T_{iA} - \tau_{iA})) \cup (\max(0, \frac{\lambda}{2}T_{iA} - \tau_{iA}), \infty)]]\} \tag{B.15}$$

Thus, the Probabilistic the lasso sign correctness is:

$$P(\text{SC}) = P(\mathcal{E}_1 \cap \mathcal{E}_2)(= P(\mathcal{E}_1)P(\mathcal{E}_2)) \tag{B.16}$$

The second equality holds when ϵ is normal. Recall the definition of P, R, and D, let $U = R + \frac{2}{\lambda} X'_C (I-P)\epsilon$ and $V = D - (X'_A X_A)^{-1} X'_A \epsilon$, then U follows the multivariate normal distribution $\mathrm{N}(R, \frac{\sigma^2}{\lambda^2} X'_C (I-P) X_C)$, and V follows the multivariate normal distribution $\mathrm{N}(D, \sigma^2 (X'_A X_A)^{-1})$. Then

$\mathcal{E}_1$ is the event of,

$$\{-1 \le U \le 1\}, \tag{B.17}$$

$\mathcal{E}_2$ is complement of,

$$\left\{ \bigcup_{i=1}^{q} [(\min(0, V_i) \le \beta_i^0 \le \max(0, V_i))] \right\}, \tag{B.18}$$

Proof of Theorem 2

Similarly with Theorem 1, for any $sgn(\hat{\beta}_A)$, we have

$$-\frac{\lambda}{2}(1 + X'_C X_A (X'_A X_A)^{-1} sgn(\hat{\beta}_A)) \le X'_C (I-P)\epsilon \le \frac{\lambda}{2}(1 - X'_C X_A (X'_A X_A)^{-1} sgn(\hat{\beta}_A)) \tag{B.19}$$

Thus, $\mathcal{E}_1^{(k)}$ is the event of,

$$\left\{-1 \le U^{(k)} \le 1\right\}, \tag{B.20}$$

And also, when $sgn(\hat{\beta}_A^{(k)}) = sgn(\beta_A)$, it is equivalent to the following inequations:

$$\hat{\beta}_A^{(k)} .* \beta_A > 0; \tag{B.21}$$

Similarly, when $sgn(\hat{\beta}_A^{(k)}) = -sgn(\beta_A)$, it is equivalent to the following inequations:

$$\hat{\beta}_A^{(k)} .* \beta_A < 0; \tag{B.22}$$

From the similar procedure in Theorem 1. Let $\mathcal{E}_{V_i}^{(k)}$ is the event $(\min(0, V_{iA}) \le \hat{\beta}_{iA}^{(k)} \le \max(0, V_{iA}))$ and $\mathcal{E}_{V_i}^{(k)C}$ is the complement of the $\mathcal{E}_{V_i}^{(k)}$, thus

$$\mathcal{E}_2^{(k)} = \{(\bigcap_{i:\delta_i^{(k)} = -\delta_i^{(0)}} \mathcal{E}_{V_i}^{(k)}) \bigcap (\bigcap_{i:\delta_i^{(k)} = \delta_i^{(0)}} \mathcal{E}_{V_i}^{(k)C})\} \tag{B.23}$$

Thus, the Probability of the lasso achieves ordinary correctness is,

$$P(\mathrm{OC}) = \sum_{k=1}^{2^q} P(\mathcal{E}_1^{(k)}) P(\mathcal{E}_2^{(k)}) \tag{B.24}$$

Proof of Proposition 1

$$P(\mathcal{E}_1) = P(-1 \le \mathrm{N1} \le 1) \tag{B.25}$$

N1 $= N(R, \frac{4}{\lambda^2}X'_C(I-P)X_C\sigma^2)$*; Let* $R_j = X'_{Cj}X_A(X'_AX_A)^{-1}sgn(\beta_A)$ *implies the jth entry in the mean vector of the normal distribution, and* $\sigma^2_{1jj} = \frac{4}{\lambda^2}X'_{Cj}(I-P)X_C\sigma^2$ *implies the* (j,j) *element in the covariance matrix.*

$$\begin{aligned}
P(\mathcal{E}_1) &= P(A_1 \cap A_2 \cdots \cap A_q) && \text{(B.26)}\\
&= 1 - P(A_1^c \cup A_2^c \cdots \cup A_q^c) && \text{(B.27)}\\
&\geq 1 - P(A_1^c) - P(A_2^c) \cdots - P(A_q^c) && \text{(B.28)}\\
&\geq 1 - \sum_{j=1}^{l} P((N(R_j, \sigma^2_{1jj}) \leq -1) \cup (N(R_j, \sigma^2_{1jj}) \geq 1)) && \text{(B.29)}
\end{aligned}$$

Where, $l = p - q$. According to the upper and lower bounds for normal distribution function,

(1) when $(1 - R_j > 0)$, the upper bound for $P(N(R_j, \sigma^2_{1jj}) \geq 1)$:

$$P(N(R_j, \sigma^2_{1jj}) \geq 1) = \frac{\sigma_{1jj}}{\sqrt{2\pi}(1-R_j)} e^{\frac{-(1-R_j)^2}{2\sigma^2_{1jj}}} \tag{B.30}$$

It's easy to know the value of the above equation will be within $[0, 0.5]$;

(2) when $(1 - R_j < 0)$, the upper bound for $P(N(R_j, \sigma^2_{1jj}) \geq 1)$:

$$P(N(R_j, \sigma^2_{1jj}) \geq 1) = 1 - \frac{\sigma_{1jj}(R_j - 1)}{\sqrt{2\pi}((1-R_j)^2 + \sigma^2_{1jj})} e^{\frac{-(1-R_j)^2}{2\sigma^2_{1jj}}} \tag{B.31}$$

Similarly, the value of the above equation will be within $[0.5, 1]$;

(3) when $(-1 - R_j < 0)$, the upper bound for $P(N(R_j, \sigma^2_{1jj}) \leq -1)$ is:

$$P(N(R_j, \sigma^2_{1jj}) \leq -1) = \frac{\sigma_{1jj}}{\sqrt{2\pi}(1+R_j)} e^{\frac{-(1+R_j)^2}{2\sigma^2_{1jj}}} \tag{B.32}$$

(4) when $(-1 - R_j > 0)$, the upper bound for $P(N(R_j, \sigma^2_{1jj}) \leq -1)$ is:

$$P(N(R_j, \sigma^2_{1jj}) \geq 1) = 1 + \frac{\sigma_{1jj}(R_j + 1)}{\sqrt{2\pi}((1+R_j)^2 + \sigma^2_{1jj})} e^{\frac{-(1+R_j)^2}{2\sigma^2_{1jj}}} \tag{B.33}$$

(5) when $(|R_j| = 1)$,

$$P((N(R_j, \sigma^2_{1jj}) \leq -1) \cup (N(R_j, \sigma^2_{1jj}) \geq 1)) = 0.5 + \Phi(\frac{-2 - R_j}{\sigma_{1jj}}) : \tag{B.34}$$

Thus, when $|R_j| < 1$, lower bound for $P(\mathcal{E}_1)$ is

$$P(\mathcal{E}_1)_{low} = 1 - \frac{\sigma_{1jj}}{\sqrt{2\pi}} \sum_{j=1}^{l} [\frac{1}{(1+R_j)} e^{\frac{-(1+R_j)^2}{2\sigma^2_{1jj}}} + \frac{1}{(1-R_j)} e^{\frac{-(1-R_j)^2}{2\sigma^2_{1jj}}}] \tag{B.35}$$

Similarly, when $R_j > 1$, lower bound for $P(\mathcal{E}_1)$ is

$$P(\mathcal{E}_1) \geq 1 - \sum_{j=1}^{l} P((N(R_j, \sigma_{1jj}^2) \leq -1) \cup (N(R_j, \sigma_{1jj}^2) \geq 1)) \tag{B.36}$$

$$\geq 1 - \sum_{j=1}^{l} [\frac{\sigma_{1jj}}{\sqrt{2\pi}(1+R_j)} e^{\frac{-(1+R_j)^2}{2\sigma_{1jj}^2}} + 1 - \frac{\sigma_{1jj}(R_j - 1)}{\sqrt{2\pi}((1-R_j)^2 + \sigma_{1jj}^2)} e^{\frac{-(1-R_j)^2}{2\sigma_{1jj}^2}}] \tag{B.37}$$

(B.38)

Cause the $sgn(\beta_A)$ is unknown, thus we use the $\tilde{R}_j$ to eliminate the sign, clearly, $|\tilde{R}_j| > |R_j|$, for (50), the sign of R_j does not affect the value, thus replace R_j with $|R_j|$. Let $f(|R_j|) = \frac{1}{(1+|R_j|)} e^{\frac{-(1+|R_j|)^2}{2\sigma_{1jj}^2}}$, $f(-|R_j|) = \frac{1}{(1-|R_j|)} e^{\frac{-(1-|R_j|)^2}{2\sigma_{1jj}^2}}$. $g(|R_j|) = f(|R_j|) + f(-|R_j|)$, it can be easily proved that $g^{'}(|R_j|) > 0$, when $|R_j| < 1$. Thus, for (50), directly replace R_j with $|\tilde{R}_j|$ can achieve the lower bound.

We can also prove (51) is also achieve the lower bound when replace R_j with $|\tilde{R}_j|$. Thus, we can eliminate the sign of β in $P(\mathcal{E}_1)_{low}$. Besides, the lower bound of (50) is the case that satisfied the irrepresentable condition, and lower bound of (51) is not. We can also get the upper bound for both of them.

For $P(\mathcal{E}_2)$, we have

$$P(\mathcal{E}_2) = P(\bigcap_i [\beta_{iA} \in [(\min(-\infty, \mathrm{N2})) \cup (\max(\mathrm{N2}, \infty)]]) \tag{B.39}$$

$$\geq 1 - \sum_{i=1}^{q} P(\min(0, \mathrm{N2i}) \leq \beta_{iA} \leq \max(0, \mathrm{N2i})) \tag{B.40}$$

Where $N2i = N(\mu_{2i}, \sigma_{2ii})$, let μ_{2i} represent the ith entry of the $\frac{\lambda}{2}(X_A' X_A)^{-1} sgn(\beta_A)$; let σ_{2ii}^2 represent the (i,i) element of the $(X_A' X_A)^{-1}\delta^2)$.

Let β_{iA} implies the *ith* entry in β_A

(1) when $\beta_{iA} > 0$ and $\mu_{2i} < \beta_{iA}$ or $\beta_{iA} > 0$ and $\mu_{2i} > \beta_{iA}$

$$P(\min(0, \mathrm{N2i}) \leq \beta_{iA} \leq \max(0, \mathrm{N2i})) = P(\mathrm{N2i} \geq \beta_{iA}) \tag{B.41}$$

$$\leq \frac{\sigma_{2ii}}{\sqrt{2\pi}(\beta_{iA} - \mu_{2i})} e^{\frac{-(\beta_{iA} - \mu_{2i})^2}{2\sigma_{2ii}^2}} \tag{B.42}$$

(2) when $\beta_{iA} < 0$ and $\mu_{2i} > \beta_{iA}$ or $\beta_{iA} < 0$ and $\mu_{2i} < \beta_{iA}$:

$$P(\min(0, \text{N2i}) \leq \beta_{iA} \leq \max(0, \text{N2i})) = P(\text{N2i} \leq \beta_{iA}) \tag{B.43}$$

$$\leq \frac{-\sigma_{2ii}}{\sqrt{2\pi}(\beta_{iA} - \mu_{2i})} e^{\frac{-(\beta_{iA}-\mu_{2i})^2}{2\sigma_{2ii}^2}} \tag{B.44}$$

Thus, when $|\mu_{2i}| < |\beta_{iA}|$, the lower bound for $P(\mathcal{E}_2)$ is:

$$P(\mathcal{E}_2)_{low} = 1 - \frac{\sigma_{2ii}}{\sqrt{2\pi}} \sum_{i=1}^{q} [\frac{1}{|\beta_{iA} - \mu_{2i}|} e^{\frac{-(\beta_{iA}-\mu_{2i})^2}{2\sigma_{2ii}^2}}] \tag{B.45}$$

Replace $|\mu_{2i}|$ with $|\tilde{\mu}_{2i}|$, clearly, $|\tilde{\mu}_{2i}| > |\mu_{2i}|$, same with the above discussion, it can be proved $P(\mathcal{E}_2)_{low}$ will decrease when $|\mu_{2i}|$ increase. Achieve the lower bound too.

Proof of Theorem 3

$$\begin{aligned}\overline{\text{RSS}}_1 &= C_{pq}^{-1} \sum_A \sum_j trace(x_j^{'} x_A (x_A^{'} x_A)^{-2} x_A^{'} x_j) = \sum_A \sum_j x_j^{'} x_A (n)^{-2} x_A^{'} x_j \\ &= C_{pq}^{-1} \frac{1}{n^2} \sum_A \sum_j x_j^{'} x_A x_A^{'} x_j = C_{pq}^{-1} \frac{1}{n^2} \sum_A \sum_j (x_A^{'} x_j)^{'} x_A^{'} x_j \\ &= \frac{1}{n^2} E(s^2)\end{aligned} \tag{B.46}$$

Proof of Proposition 2

We can get the gradient of RSS_q for any given column, noted as Xb. Recall the definition of RSS_q:

$$\begin{aligned}\text{RSS}_q(D) &= \sum_{A:|A|=q} \sum_{j \in C} X_j' X_A (X_A{}' X_A)^{-2} X_A' X_j \\ &= \sum_A \text{trace}(X_C' X_A (X_A{}' X_A)^{-2} X_A' X_C)\end{aligned}$$

Now we sequentially look into the scenarios with different q.

(1) $q = 1$

Firstly, separate the summation into two parts, $X_b \in A$ or $Xb \notin A$,

$$\begin{aligned}
\mathrm{RSS}_q(D) &= \sum_{A:|A|=q} \sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j \\
&= \sum_{X_b\in A} `\sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j + \sum_{X_b\notin A} \sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j \\
&= \sum_{X_b\in A} \sum_{j\in C} X_j' X_b ({X_b}' X_b)^{-2} X_b' X_j + \sum_{i\neq b} \sum_{j\in C} X_j' X_i ({X_i}' X_i)^{-2} X_i' X_j \\
&= \sum_{X_b\in A} \sum_{j\in C} X_j' X_b ({X_b}' X_b)^{-2} X_b' X_j + \sum_{i\neq b} \sum_{j\in C} X_j' X_i ({X_i}' X_i)^{-2} X_i' X_j \\
&= n^{-2}[\sum_{X_b\in A} \sum_{j\in C} X_j' X_b X_b' X_j + \sum_{i\neq b} \sum_{j\in C} X_j' X_i X_i' X_j]
\end{aligned}$$

The partial gradient of RSS_1 for X_b is:

$$\begin{aligned}
\frac{\partial}{\partial X_b}\mathrm{RSS}_1 &= n^{-2}\frac{\partial}{\partial X_b}[\sum_{X_b\in A} \sum_{j\in C} X_j' X_b X_b' X_j] + n^{-2}\frac{\partial}{\partial X_b}[\sum_{i\neq b} \sum_{j\in C} X_j' X_i X_i' X_j] \\
&= n^{-2}\frac{\partial}{\partial X_b}[\sum_{j\in C} X_b' X_j X_j' X_b] + n^{-2}\frac{\partial}{\partial X_b}[\sum_{i\neq b} X_b' X_i X_i' X_b] \\
&= 2n^{-2}\frac{\partial}{\partial X_b}[\sum_{j\in C} X_b' X_j {X_j}' X_b] \\
&= 4n^{-2} X_c X_c' X_b
\end{aligned}$$

(2) $q = 2$

Similarly, separate the summation into two parts, $X_b \in A$ or $Xb \notin A$,

$$\begin{aligned}
\mathrm{RSS}_q(D) &= \sum_{A:|A|=q} \sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j \\
&= \sum_{X_b\in A} \sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j + \sum_{X_b\notin A} \sum_{j\in C} X_j' X_A ({X_A}' X_A)^{-2} X_A' X_j \\
&= \mathrm{RSSA}^{(2)} + \mathrm{RSSB}^{(2)}
\end{aligned}$$

Here, For RSSA$^{(2)}$, note X_i as the other column in A except the X_b, the partial gradient of RSSA$^{(2)}$is

$$\begin{aligned}
\frac{\partial}{\partial X_b}\mathrm{RSSA}^{(2)} &= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C} X_j' X_A({X_A}'X_A)^{-2}X_A' X_j \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C} X_j'(X_b, X_i)\begin{pmatrix} {X_b}'X_b & {X_b}'X_i \\ {X_i}'X_b & {X_i}'X_i \end{pmatrix}^{-2}\begin{pmatrix} X_b' \\ X_i' \end{pmatrix} X_j \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C}({X_j}'X_b, {X_j}'X_i)\begin{pmatrix} {X_b}'X_b & {X_b}'X_i \\ {X_i}'X_b & {X_i}'X_i \end{pmatrix}^{-2}\begin{pmatrix} {X_b}'X_j \\ {X_i}'X_j \end{pmatrix} \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C}(S, C)\begin{pmatrix} n & t \\ t & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C \end{pmatrix} \\
&= \frac{\partial}{\partial S}[\sum_{X_b\in A}\sum_{j\in C}(S, C)\begin{pmatrix} n & t \\ t & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C \end{pmatrix}]\frac{\partial S}{\partial X_b} \\
&+ \frac{\partial}{\partial t}[\sum_{X_b\in A}\sum_{j\in C}(S, C)\begin{pmatrix} n & t \\ t & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C \end{pmatrix}]\frac{\partial t}{\partial X_b} \\
&= PA_1^{(2)} + PA_2^{(2)}
\end{aligned}$$

where, $S = {X_j}'X_b, C = {X_j}'X_i, t = {X_b}'X_i$, n is the row number of X. Then the first term of above equation is

$$\begin{aligned}
PA_1^{(2)} &= \frac{\partial}{\partial S}[\sum_{X_b\in A}\sum_{j\in C}(S, C)\begin{pmatrix} n & t \\ t & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C \end{pmatrix}]\frac{\partial S}{\partial X_b} \\
&= \sum_{X_b\in A}\sum_{j\in C}[2(1, 0)\begin{pmatrix} n & t \\ t & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C \end{pmatrix}]X_j
\end{aligned}$$

The second term of the above equation is

$$\begin{aligned}
PA_2^{(2)} &= \frac{\partial}{\partial t}[\sum_{X_b\in A}\sum_{j\in C}(S,C)\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-2}\begin{pmatrix} S \\ C\end{pmatrix}]\frac{\partial t}{\partial X_b} \\
&= \sum_{X_b\in A}\sum_{j\in C}(S,C)\frac{\partial}{\partial t}[\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-2}]\begin{pmatrix} S \\ C\end{pmatrix}X_i \\
&= \sum_{X_b\in A}\sum_{j\in C}(S,C)[\frac{\partial}{\partial t}[\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1}]\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1} + \begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1}\frac{\partial}{\partial t}[\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1}]]\begin{pmatrix} S \\ C\end{pmatrix}X_i \\
&= -2\sum_{X_b\in A}\sum_{j\in C}(S,C)[\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1}\begin{pmatrix} 0 & 1 \\ 1 & 0\end{pmatrix}\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-2} \\
&+ \begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-2}\begin{pmatrix} 0 & 1 \\ 1 & 0\end{pmatrix}\begin{pmatrix} n & t \\ t & n\end{pmatrix}^{-1}]\begin{pmatrix} S \\ C\end{pmatrix}X_i
\end{aligned}$$

For RSSB, the partial gradient is

$$\begin{aligned}
\frac{\partial}{\partial X_b}\text{RSSB}^{(2)} &= \frac{\partial}{\partial X_b}[\sum_{X_b\notin A}\sum_{j\in C}X_j'X_A(X_A'X_A)^{-2}X_A'X_j] \\
&= \frac{\partial}{\partial X_b}[\sum_{X_b\notin A}X_b'X_A(X_A'X_A)^{-2}X_A'X_b] \\
&= \sum_{X_b\notin A}2X_A(X_A'X_A)^{-2}X_A'X_b
\end{aligned}$$

Thus, the partial gradient of RSS_2 is

$$\begin{aligned}
\frac{\partial}{\partial X_b}\text{RSS}_2 &= \frac{\partial}{\partial X_b}\text{RSSA}^{(2)} + \frac{\partial}{\partial X_b}\text{RSSB}^{(2)} \\
&= PA_1^{(2)} + PA_2^{(2)} + \frac{\partial}{\partial X_b}PB^{(2)}
\end{aligned}$$

(3) $q = 3$

Similarly, separate the summation into two parts, $X_b \in A$ or $Xb \notin A$,

$$\begin{aligned}
\text{RSS}_q(D) &= \sum_{A:|A|=q}\sum_{j\in C}X_j'X_A({X_A}'X_A)^{-2}X_A'X_j \\
&= \sum_{X_b\in A}\sum_{j\in C}X_j'X_A({X_A}'X_A)^{-2}X_A'X_j + \sum_{X_b\notin A}\sum_{j\in C}X_j'X_A({X_A}'X_A)^{-2}X_A'X_j \\
&= \text{RSSA}^{(3)} + \text{RSSB}^{(3)}
\end{aligned}$$

Here, For $\text{RSSA}^{(3)}$, note X_{i_1}, X_{i_2} as the other two columns in A except the X_b, the partial gradient of $\text{RSSA}^{(3)}$ is

Let $T_3 = \begin{pmatrix} X_b{}'X_b & X_b{}'X_{i_1} & X_b{}'X_{i_2} \\ X_{i_1}{}'X_b & X_{i_1}{}'X_{i_1} & X_{i_1}{}'X_{i_2} \\ X_{i_2}{}'X_b & X_{i_2}{}'X_{i_1} & X_{i_2}{}'X_{i_2} \end{pmatrix}^{-2} \begin{pmatrix} X_b' \\ X_{i_1}' \\ X_{i_2}' \end{pmatrix}$

$$\begin{aligned}
\frac{\partial}{\partial X_b}\mathrm{RSSA}^{(3)} &= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C} X_j'X_A(X_A{}'X_A)^{-2}X_A'X_j \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C} X_j'(X_b, X_{i_1}, X_{i_2})T_3^{-2}\begin{pmatrix} X_b' \\ X_{i_1}' \\ X_{i_2}' \end{pmatrix} X_j \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C}(X_j{}'X_b, X_j{}'X_{i_1}, X_j{}'X_{i_2})T_3^{-2}\begin{pmatrix} X_b{}'X_j \\ X_{i_1}{}'X_j \\ X_{i_2}{}'X_j \end{pmatrix} \\
&= \frac{\partial}{\partial X_b}\sum_{X_b\in A}\sum_{j\in C}(S, C_1, C_2)\begin{pmatrix} n & t_1 & t_2 \\ t_1 & n & H_1 \\ t_2 & H_1 & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} \\
&= \frac{\partial}{\partial S}Z\frac{\partial}{\partial X_b}S + \frac{\partial}{\partial t_1}Z\frac{\partial}{\partial X_b}t_1 + \frac{\partial}{\partial t_2}Z\frac{\partial}{\partial X_b}t_2 \\
&= (\frac{\partial}{\partial S}Z)X_j' + (\frac{\partial}{\partial t_1}Z)X_{i_1}' + (\frac{\partial}{\partial t_2}Z)X_{i_2}' \\
&= PA_1^{(3)} + PA_2^{(3)} + PA_3^{(3)}
\end{aligned}$$

where,

$$Z = \sum_{X_b\in A}\sum_{j\in C}(S, C_1, C_2)\begin{pmatrix} n & t_1 & t_2 \\ t_1 & n & H_1 \\ t_2 & H_1 & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix},$$

Let $S = X_j{}'X_b$, $C_1 = X_j{}'X_{i_1}$, $C_2 = X_j{}'X_{i_2}$, $H_1 = X_{i_1}{}'X_{i_2}$, $t_1 = X_b{}'X_{i_1}$, $t_2 = X_b{}'X_{i_2}$, $n = X_b{}'X_b = X_{i_1}{}'X_{i_1} = X_{i_2}{}'X_{i_2}$.

Let $B = \begin{pmatrix} n & t_1 & t_2 \\ t_1 & n & H_1 \\ t_2 & H_1 & n \end{pmatrix}$

Then,

$$\begin{aligned}
PA_1^{(3)} &= (\frac{\partial}{\partial S}Z)X_j' \\
&= \sum_{X_b\in A}\sum_{j\in C}(2(1,0,0)\begin{pmatrix} n & t_1 & t_2 \\ t_1 & n & H_1 \\ t_2 & H_1 & n \end{pmatrix}^{-2}\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} X_j')
\end{aligned}$$

$$\begin{aligned}
PA_2^{(3)} &= (\frac{\partial}{\partial t_1}Z)X_{i_1}' \\
&= \sum_{X_b\in A}\sum_{j\in C}(-2(S, C_1, C_2)[B^{-1}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}B^{-2} + B^{-2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}B^{-1}]\begin{pmatrix} S \\ C_1 \\ C_2 \end{pmatrix} X_{i_1}')
\end{aligned}$$

$$\begin{aligned}
PA_3^{(3)} &= (\frac{\partial}{\partial t_2}Z)X'_{i_2} \\
&= \sum_{X_b\in A}\sum_{j\in C}(-2(S,C_1,C_2)[B^{-1}\begin{pmatrix}0&0&1\\0&0&0\\1&0&0\end{pmatrix}B^{-2}+B^{-2}\begin{pmatrix}0&0&1\\0&0&0\\1&0&0\end{pmatrix}B^{-1}]\begin{pmatrix}S\\C_1\\C_2\end{pmatrix}X'_{i_1})
\end{aligned}$$

For RSSB, similar with RSS$_2$, the partial gradient is

$$\begin{aligned}
\frac{\partial}{\partial X_b}\text{RSSB}^{(3)} &= \frac{\partial}{\partial X_b}[\sum_{X_b\notin A}\sum_{j\in C}X'_jX_A(X'_AX_A)^{-2}X'_AX_j] \\
&= \frac{\partial}{\partial X_b}[\sum_{X_b\notin A}X'_bX_A(X'_AX_A)^{-2}X'_AX_b] \\
&= \sum_{X_b\notin A}2X_A(X'_AX_A)^{-2}X'_AX_b \\
&= PB^{(3)}.
\end{aligned}$$

Thus, the partial gradient of RSS$_3$ is

$$\begin{aligned}
\frac{\partial}{\partial X_b}\text{RSSB}_3 &= \frac{\partial}{\partial X_b}\text{RSSA}^{(3)}+\frac{\partial}{\partial X_b}\text{RSSB}^{(3)} \\
&= PA_1^{(3)}+PA_2^{(3)}+PA_3^{(3)}+PB^{(3)}
\end{aligned}$$

(4) for genearl q

From the Mathematical Induction, it's easy to get any q scenarios partial gradient formulation:

Similarly, separate the summation into two parts, $X_b\in A$ or $Xb\notin A$,

$$\begin{aligned}
\text{RSS}_q(D) &= \sum_{A:|A|=q}\sum_{j\in C}X'_jX_A(X_A{}'X_A)^{-2}X'_AX_j \\
&= \sum_{X_b\in A}\sum_{j\in C}X'_jX_A(X_A{}'X_A)^{-2}X'_AX_j+\sum_{X_b\notin A}\sum_{j\in C}X'_jX_A(X_A{}'X_A)^{-2}X'_AX_j \\
&= \text{RSSA}^{(q)}+\text{RSSB}^{(q)}
\end{aligned}$$

Here, For RSSA$^{(q)}$, note $X_{i_1},X_{i_2},\ldots,X_{i_{q-1}}$ as the other columns in A except the X_b, the partial gradient of RSSA$^{(q)}$ is

Let $T_q = \begin{pmatrix} {X_b}'X_b & {X_b}'X_{i_1} & {X_b}'X_{i_2} & \dots {X_b}'X_{i_{q-1}} \\ {X_{i_1}}'X_b & {X_{i_1}}'X_{i_1} & {X_{i_1}}'X_{i_2} & \dots {X_{i_1}}'X_{i_{q-1}} \\ {X_{i_2}}'X_b & {X_{i_2}}'X_{i_1} & {X_{i_2}}'X_{i_2} & \dots {X_{i_2}}'X_{i_{q-1}} \\ \dots & \dots & \dots & \dots \\ {X_{i_{q-1}}}'X_b & {X_{i_{q-1}}}'X_{i_1} & {X_{i_{q-1}}}'X_{i_2} & \dots {X_{i_{q-1}}}'X_{i_{q-1}} \end{pmatrix}$

$$= \frac{\partial}{\partial X_b} \sum_{X_b \in A} \sum_{j \in C} X_j' X_A ({X_A}'X_A)^{-2} X_A' X_j$$

$$= \frac{\partial}{\partial X_b} \sum_{X_b \in A} \sum_{j \in C} X_j' (X_b, X_{i_1}, X_{i_2}, \dots, X_{i_{q-1}}) T_q^{-2} \begin{pmatrix} X_b' \\ X_{i_1}' \\ X_{i_2}' \\ \dots \\ X_{i_{q-1}}' \end{pmatrix} X_j$$

$$= \frac{\partial}{\partial X_b} \sum_{X_b \in A} \sum_{j \in C} ({X_j}'X_b, {X_j}'X_{i_1}, {X_j}'X_{i_2}, \dots, {X_j}'X_{i_{q-1}}) T_q^{-2} \begin{pmatrix} {X_b}'X_j \\ {X_{i_1}}'X_j \\ {X_{i_2}}'X_j \\ \dots \\ {X_{i_{q-1}}}'X_j \end{pmatrix}$$

$$= \frac{\partial}{\partial X_b} \sum_{X_b \in A} \sum_{j \in C} (S, C_1, C_2, \dots, C_{q-1}) \begin{pmatrix} n & t_1 & t_2 & \dots & t_{q-1} \\ t_1 & n & H_{1,2} & \dots & H_{1,q-1} \\ t_2 & H_{1,2} & n & \dots & H_{2,q-1} \\ \dots & \dots & \dots & \dots & \dots \\ t_{q-1} & H_{1,q-1} & H_{2,q-1} & \dots & n \end{pmatrix}^{-2} \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix}$$

$$= \frac{\partial}{\partial S} Z \frac{\partial}{\partial X_b} S + \frac{\partial}{\partial t_1} Z \frac{\partial}{\partial X_b} t_1 + \frac{\partial}{\partial t_2} Z \frac{\partial}{\partial X_b} t_2 + \dots + \frac{\partial}{\partial t_2} Z \frac{\partial}{\partial X_b} t_{q-1}$$
$$= (\frac{\partial}{\partial S} Z) X_j' + (\frac{\partial}{\partial t_1} Z) X_{i_1}' + (\frac{\partial}{\partial t_2} Z) X_{i_2}' + \dots + (\frac{\partial}{\partial t_2} Z) X_{i_{q-1}}'$$
$$= PA_1^{(q)} + PA_2^{(q)} + PA_3^{(q)} + \dots + PA_{k-1}^{(q)}$$

where,

$$Z = \sum_{X_b \in A} \sum_{j \in C} (S, C_1, C_2, \dots, C_{q-1}) \begin{pmatrix} n & t_1 & t_2 & \dots & t_{q-1} \\ t_1 & n & H_{1,2} & \dots & H_{1,q-1} \\ t_2 & H_{1,2} & n & \dots & H_{2,q-1} \\ \dots & \dots & \dots & \dots & \dots \\ t_{q-1} & H_{1,q-1} & H_{2,q-1} & \dots & n \end{pmatrix}^{-2} \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix}.$$

Let $S = {X_j}'X_b$, $C_1 = {X_j}'X_{i_1}$, $C_2 = {X_j}'X_{i_2}$, $C_{q-1} = {X_{i_1}}'X_{i_{q-1}}$, $t_1 = {X_b}'X_{i_1}$, $t_2 = {X_b}'X_{i_2}$, $n = {X_b}'X_b = {X_{i_1}}'X_{i_1} = {X_{i_2}}'X_{i_2} = \dots = {X_{i_{q-1}}}'X_{i_{q-1}}$, $H_{1,2} = {X_{i_1}}'X_{i_2}$, $H_{1,q-1} = {X_{i_1}}'X_{i_{q-1}}$, $H_{2,q-1} = {X_{i_2}}'X_{i_{q-1}}$.

Let $B = \begin{pmatrix} n & t_1 & t_2 & \dots & t_{q-1} \\ t_1 & n & H_{1,2} & \dots & H_{1,q-1} \\ t_2 & H_{1,2} & n & \dots & H_{2,q-1} \\ \dots & \dots & \dots & \dots & \dots \\ t_{q-1} & H_{1,q-1} & H_{2,q-1} & \dots & n \end{pmatrix}$

Then,

$$PA_1^{(q)} = (\frac{\partial}{\partial S} Z) X'_j$$

$$= \sum_{X_b \in A} \sum_{j \in C} (2(1, 0_1, 0_2, \dots, 0_{q-1}) \begin{pmatrix} n & t_1 & t_2 & \dots & t_{q-1} \\ t_1 & n & H_{1,2} & \dots & H_{1,q-1} \\ t_2 & H_{1,2} & n & \dots & H_{2,q-1} \\ \dots & \dots & \dots & \dots & \dots \\ t_{q-1} & H_{1,q-1} & H_{2,q-1} & \dots & n \end{pmatrix}^{-2} \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix} X'_j)$$

$$PA_2^{(q)} = (\frac{\partial}{\partial t_1} Z) X'_{i_1}$$

$$= \sum_{X_b \in A} \sum_{j \in C} (-2(S, C_1, C_2, \dots, C_{q-1})[B^{-1} \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-2}$$

$$+ B^{-2} \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-1}] \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix} X'_{i_1})$$

$$PA_3^{(q)} = (\frac{\partial}{\partial t_2} Z) X'_{i_2}$$

$$= \sum_{X_b \in A} \sum_{j \in C} (-2(S, C_1, C_2, \dots, C_{q-1})[B^{-1} \begin{pmatrix} 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-2}$$

$$+ B^{-2} \begin{pmatrix} 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-1}] \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix} X'_{i_2})$$

$\dots$

$$PA_q^{(q)} = (\frac{\partial}{\partial t_1} Z) X'_{i_{q-1}}$$

$$= \sum_{X_b \in A} \sum_{j \in C} (-2(S, C_1, C_2, \dots, C_{q-1})[B^{-1} \begin{pmatrix} 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-2}$$

$$+ B^{-2} \begin{pmatrix} 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}_{q \times q} B^{-1}] \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix} X'_{i_{q-1}})$$

Let M_q^h to be the q by q matrix in which the element $(h, 1)$ and $(1, h)$ is 1 and all the others are 0, then in general case

$$PA_h^{(q)} = (\frac{\partial}{\partial t_1} Z) X'_{i_{h-1}}$$

$$= \sum_{X_b \in A} \sum_{j \in C} (-2(S, C_1, C_2, \dots, C_{q-1}) B^{-1} [M_q^h B^{-1} + B^{-1} M_q^h] B^{-1} \begin{pmatrix} S \\ C_1 \\ C_2 \\ \dots \\ C_{q-1} \end{pmatrix} X'_{i_{h-1}})$$

For RSSB$^{(q)}$, the partial gradient is

$$\begin{aligned} \frac{\partial}{\partial X_b} \mathrm{RSSB}^{(q)} &= \frac{\partial}{\partial X_b} [\sum_{X_b \notin A} \sum_{j \in C} X'_j X_A (X'_A X_A)^{-2} X'_A X_j] \\ &= \frac{\partial}{\partial X_b} [\sum_{X_b \notin A} X'_b X_A (X'_A X_A)^{-2} X'_A X_b] \\ &= \sum_{X_b \notin A} 2 X_A (X'_A X_A)^{-2} X'_A X_b \end{aligned}$$

Thus, the partial gradient of RSSB$_q$ is

$$\begin{aligned} \frac{\partial}{\partial X_b} \mathrm{RSSB}_q &= \frac{\partial}{\partial X_b} \mathrm{RSSA}^{(q)} + \frac{\partial}{\partial X_b} \mathrm{RSSA}^{(q)} \\ &= PA_1^{(q)} + \sum_{h=2}^{q} PA_h^{(q)} + PB^{(q)} \end{aligned}$$

LIST OF REFERENCES

LIST OF REFERENCES

[1] J. P. C. Kleijnen. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter 6, pages 173–224. John Wiley & Sons, New York, 1998.

[2] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization using Designed Experiments. 2nd ed.* John Wiley & Sons, 2002.

[3] K. Saeger and J. Hinch. Understanding instability in a complex deterministic combat simulation. *Military Operations Research*, 6:43–55, 2001.

[4] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995.

[5] Guido Buzzi-Ferraris and Pio Forzatti. A new sequential experimental design procedure for discriminating among rival models. *Chemical Engineering Science*, 38:225–232, 1983.

[6] Henry P. Wynn. The sequential generation of d-optimum experimental designs. *The Annals of Mathematical Statistic*, 41:1655–1664, 1970.

[7] B. Bettonvil and J. P. C. Kleijnen. Searching for important factors in simulation models with many factors: Sequential bifurcation. *European Journal of Operational Research*, 96:180–194, 1997.

[8] H. Wan, B. E. Ankenman, and B. L. Nelson. Controlled sequential bifurcation: A new factor screening method for discrete-event simulation. *Operations Research*, 54:743–755, 2006.

[9] A. M. Dean and S. M. Lewis. Comparison of group screening strategies for factorial experiments. *Computational Statistics & Data Analysis*, 39:287–297, 2002.

[10] Hong Wan and Bruce E. Ankenman. Two-stage controlled fractional factorial screening for simulation experiments. *Journal of Quality Technology*, 39:126–139, 2007.

[11] Kathleen H. V. Booth and D. R. Cox. Some systematic supersaturated designs. *Technometrics*, 4:489–495, 1962.

[12] Dennis K. J. Lin. A new class of supersaturated designs. *Technometrics*, 35:28–31, 1993.

[13] C.F.J. Wu. Construction of supersaturated designs through partially aliased interactions. *Biometrika*, 80:661–669, 1993.

[14] C.L. Mallows. Some comments on cp. *Technometrics*, 15:661–675, 1973.

[15] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[16] Gideon E. Schwarz. Estimating the dimension of a model. *IEEE Transactions on Automatic Control*, 6:461–464, 1978.

[17] Pierre A. Devijver and Josef Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall, London, GB, 1982.

[18] Alan J. Miller. Selection of subsets of regression variables. *J.R.Statist.Soc.A*, 147:389–425, 1984.

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58:267–288, 1996.

[20] M.R. Osborne, B. Presnell, and B.A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–403, 2000.

[21] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[22] R. Fan, J.and Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of American Statistical Association*, 96:1348–1360, 2001.

[23] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

[24] Emmanuel Candes and Terence Tao. The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35:2313–2351, 2007.

[25] Runze Li and Dennis K.J. Lin. Data analysis in supersaturated designs. *Statistics & Probability Letters*, 59:135–144, 2002.

[26] Ming Yuan and Yi Lin. Efficient empirical bayes variable selection and estimation in linear models. *American Statistical Association*, 100:1215–1225, 2005.

[27] F. K. H. Phoa, Yu-Hui Pan, and Hongquan Xu. Analysis of supersaturated designs via the dantzig selector. *Journal of Statistical Planning and Inference*, 139:2362–2372, 2009.

[28] Steven G. Gilmour. *Handbook of Screening*, chapter 8, pages 169–332. Springer, New York, 2006.

[29] W. W. Li and C. F. J. Wu. Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. *Technometrics*, 39:171–179, 1997.

[30] B.Tang and C.F.J.Wu. A method for constructing supersaturated designs and its Es^2 optimality. *Canadian Journal of Statistics*, 25:191–201, 1997.

[31] Ching-Shui Cheng. E(s2)-optimal supersaturated designs. *Statistica Sinica*, 7:929–939, 1997.

[32] N.K.Nguyen. An algorithmic approach to constructing supersaturated designs. *Statistical Sinica*, 38:69–73, 1996.

[33] D. A. Bulutoglu and C.S.Cheng. Construction of E(s^2)-optimal supersaturated designs. *Annals of Statistics*, 32:1162–1678, 2004.

[34] Minqian Liu and Runchu Zhang. Construction of e(s2) optimal supersaturated designs using cyclic bibds. *Journal of Statistical Planning and Inference*, 91:139–150, 2000.

[35] K. M. Eskridgea, S. G. Gilmour, R. Mead, N. A. Butler, and D. A. Travnicek. Large supersaturated designs. *Journal of Statistical Computation and Simulation*, 74:525–542, 2004.

[36] Yufeng Liu and Angela Dean. k-circulant supersaturated designs. *Technometrics*, 46:32–43, 2004.

[37] Stelios D. Georgiou. Modelling by supersaturated designs. *Computational Statistics Data Analysis*, 53:428–435, 2008.

[38] N.A.Butler, R. Mead, K.M. Eskridge, and S.G.Gilmour. A general method of constructing E(s^2)-optimal supersaturated designs. *Journal of the Royal Statistical Society B*, 63:621–632, 2001.

[39] D.A. Bulutoglu and K. J. Ryan. E(s^2)-optimal supersaturated designs with good minimax properties when n is odd. *Journal of Statistical Planning and Inference*, 138:1754–1762, 2008.

[40] M. E. Johnson and C. J. Nachtsheim. Some guidelines for constructing exact doptimum designs on convex design spaces. *Technometrics*, 25:271–277, 1983.

[41] D.K.J. Lin. Generating systematic supersaturated designs. *Technometrics*, 37:213–225, 1995.

[42] Q.M.Shao. Bounds and estimates of a basic constant in extreme value theory of gaussian processes. *Statistica Sinica*, 6:245–257, 1996.

[43] Jiangqian Fan and Jinchi Lv. A selective overview of variable selection in high dimension feature space. *Statistica Sinica*, 20:101–148, 2010.

[44] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *J. Royal. Statist. Soc B.*, 67:91–108, 2005.

[45] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variable. *Issue Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

[46] Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35:1012–1030, 2007.

[47] Jerome H. Friedman, T. Hastie, Holger Hofling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[48] Hui Zou, Trevor Hastie, and Robert Tibshirani. On the degrees of freedom of the lasso. *The Annals of Statistics*, 35:2173–2192, 2007.

[49] P.Zhao and B.Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2567, 2006.

[50] Chong Gu. Cross-validating non-gaussian data. *J. Comput. Graph. Statist.*, 1:169–179, 1992.

[51] H.Wan Y. Zhu T. Kaymal Xing, D. and S. M. Sanchez. Simulation screening experiments using lasso-optimal supersaturated design and analysis: A maritime operations application. In A. Tolk R. Pasupathy, S-H. Kim and R. Hill, editors, *Proceedings of the 2013 Winter Simulation Conference(WSC)*, 2013.

[52] Lei Liu and Yu Zhu. Partially projected gradient algorithms for computing nonparametric maximum likelihood estimates of mixing distributions. *Journal of Statistical Planning and Inferences*, 137:2509–2522, 2007.

[53] Turgut Kaymal. Assessing the operational effectiveness of a small surface combat ship in an anti-surface warfare environment. Master's thesis, Naval Postgraduate School, Monterey, California, June 2013.

[54] A. Kimber and J. Booth. Novel technologies for enhancing the performance of constabulary and security orientated ships. In *Proceedings of the 2010 RINA Warship Conference.* London: The Royal Institute of Naval Architects, 2010.

[55] M. Annati. OPV for surface patrol: Not every country needs or can afford sophisticated ships and systems. *Naval Forces*, 30, 2009.

[56] J. C. Ryan and O. P. Jons. Improving the ship design, acquisition and construction process. *Naval Engineers Journal*, 104(2):39–57, 2004.

[57] Igor Mizine, Bruce Wintersteen, and Steven Wynn. A multi-level hierarchical system approach to ship concept formulation tools. *Naval Engineers Journal*, 124(3):93–120, 2012.

[58] G. Luft and A. Korin. Terrorism goes to sea. *Foreign Affairs*, 83(6):61–67, 2004.

[59] E. Maggio. *Private Security in the 21st Century: Concepts and Applications.* Sudbury, Massachusetts: Jones and Bartlett Publishers, 2008.

[60] M. Lauren and R. Stephen. Map-Aware Non-uniform Automata (MANA)—a New Zealand approach to scenario modelling. *Journal of Battlefield Technology*, 5(1):27–31, 2002.

[61] G. C. McIntosh, D. P. Galligan, M. A. Anderson, and M. K. Lauren. *MANA Version 4.0 User Manual (Technical Note 2007/3 NR 1465.).* New Zealand: Defence Technology Agency, 2007.

[62] G. C. McIntosh. *MANA–V (Map Aware Non-uniform Automata–Vector) Supplementary Manual.* New Zealand: Defence Technology Agency, 2009.

[63] H. Vieira Jr, S. M. Sanchez, K. H. Kienitz, and M. C. N. Belderrain. Improved efficient, nearly orthogonal, nearly balanced mixed designs. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, editors, *Proceedings of the 2011 Winter Simulation Conference*, pages 3600–3611. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 2011.

[64] S. M. Sanchez and P. J. Sanchez. Very large fractional factorial and central composite designs. *ACM Transactions on Modeling & Computer Simulation*, 15(4):362–377, 2005.

[65] S. M. Sanchez, T. W. Lucas, P. J. Sanchez, H. Wan, and C. Nannini. Designing large scale simulation experiments, with applications to defense and homeland security. In K. Hinkelman, editor, *Design and Analysis of Experiments Volume III: Special Designs and Applications*, pages 413–441. New York: John Wiley & Sons, 2012.

[66] Jinchi Lv and Jiangqian Fan. A unified approach to model selection and sparse recovery using regularized least squares. *Annals of Statistics*, 37:3498–3528, 2009.

VITA

VITA

Mr. Dadi Xing received his Bachelor degree in Electronic Engineering from the University and Technology of China in June 2001 and Master degree in Biomedical Engineering from South China University of Technology in June 2009. He joined Purdue University as a graduate student in the fall of 2010. He now completes his Ph.D. under the supervision of Professors Hong Wan and Michael Yu Zhu.

His research interests mainly focus on simulation based experiment design and model selection methods, and the applications in various areas, including operations management, health care services and financial engineering. His papers on the experiment design and model selection have been submitted to Technometrics and other top journals which are currently under review.

He will join the Discover Financial Services in Chicago as a project manager after graduation.