SPACE-FILLING DESIGNS FOR NUMERICAL INTEGRATION AND STOCHASTIC PROGRAMMING

by

Jiajie Chen

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Statistics)

at the

UNIVERSITY OF WISCONSIN–MADISON

2014

Date of final oral examination: 06/16/2014

The dissertation is approved by the following members of the Final Oral Committee:

Zhiguang Qian, Professor, Statistics
Jeffrey T. Linderoth, Professor, Industrial and Systems Engineering
Qing Liu, Assistant Professor, Marketing
Kam-Wah Tsui, Professor, Statistics
Stephen J. Wright, Professor, Computer Sciences

To my family

Acknowledgments

I cannot believe my graduate study is about to end so quickly. As expected and unexpected, I have experienced challenges, frustration, disappointment, happiness and success during the past five years. I would like to acknowledge the importance of several people I could not have imagined how my life would be without their kind help and encouragement.

I would like to express the deepest appreciation to my thesis advisor, Professor Peter Qian, for his invaluable guidance and persistent help in my study. I feel very fortunate to be a PhD student of Peter, only under whom I could receive that many opportunities to work with experts from interdisciplinary fields to broaden my knowledge. Peter is also a good advisor beyond research as he convincingly delivers a spirit of adventure for being a successful man.

I want to thank Professor Kam-Wah Tsui, Professor Qing Liu, Professor Jeff Linderoth and Professor Steve Wright for kindly serving as my committee members. I would like to thank Professor Kam-Wah Tsui especially for encouraging me and sharing his stories with me during the stressful job hunting period. I wish to express my gratitude to Professor Qing Liu who continuously forwarded her expertise in

marketing science to me without reservation. Also, a big thank you to Professor Jeff Linderoth and Professor Steve Wright who provided me with insightful comments and suggestions which lead to Chapter 4 of my dissertation.

I place on record, my sincere thanks to Professor Jin Xu of East China Normal University and Cong Han Lim from the Department of Computer Sciences at the University of Wisconsin-Madison. I cannot finish Chapters 2 and 4 without their hard work and precious inputs. I would like to thank Youngdeok Hwang, Xu Xu, Qiong Zhang and Yan Chen, for being my great research colleagues. I also thank my 4-year roommate Xiao Guo who helped me a lot outside university. I appreciate all of other students at UW-Madison who have made my life and study much easier.

Last but definitely not the least, I owe so much to my parents and Mengfan who have been supporting me all the way to pull through difficulties. This thesis is dedicated to them.

Contents

List of Tables

List of Figures

Abstract

This dissertation focuses on developing several space-filling designs for evaluating ensembles of computer experiments in one shot, sequentially or in a parallel manner, and validating sample average approximation solutions.

Chapter 1 reviews several existing space-filling designs for numerical integration. Chapter 2 introduces iteratively enlarged Latin hypercube designs for running computer experiments sequentially and provides a theoretical framework to put this approach on a firm footing. Chapter 3 illustrates a method to construct Latin hypercube designs with both controlled correlations and multi-dimensional stratification which not only filter out lower-dimensional variance components but also bilinear terms effectively. Chapter 4 describes sampling methods to produce negatively dependent batches for reducing the variance of sample-averaged lower bound estimators in stochastic programming. Chapter 5 presents an algorithm to construct sliced Latin hypercube designs with controlled column-wise correlations for both the entire design and each slice.

Chapter 1

Introduction

1.1 Latin Hypercube Designs for Numerical Integration, Computer Experiments and Uncertainty Quantification

Numerical integration appears frequently in statistics and other fields. For example, one may be interested in estimating the expected output of a computer code (McKay et al., 1979; Tang, 1993). Computer models like finite-element analysis codes and computational fluid dynamics codes are widely applied now in diverse fields (Koehler and Owen, 1996; Santner et al., 2003; Fang et al., 2005).

Numerical integration is also important for uncertainty quantification, a rapidly increasing field in applied mathematics and statistics. Estimating the expected output is frequently applied in stochastic computation in the context of understanding

the impact of uncertainty on simulation results. The field of stochastic computation is rather new, but grows rapidly due to the increasing need to conduct verifications and validations and uncertainty quantification for practical systems (Xiu, 2010). One numerical approach in stochastic computation is the collocation method, where one executes established deterministic code on prescribed samples of the random inputs. The solution mean can be accessed from the solution ensembles after completing the simulations. Consider an ordinary differential equation (Xiu, 2010):

$$\frac{du}{dt}(t, x_1, x_2) = -x_1 u, \qquad u(0, x_1, x_2) = x_2,$$

where the rate constant x_1 and the initial condition x_2 are random variables. The solution is $u(t, x_1, x_2) = x_2 \exp\{-x_1 t\}$, which is random. It is then of great interest to know the expected value at time t.

We focus on the estimation of $\mu = E\{f(X)\}$ for a deterministic function $f(X) \in \mathbb{R}$ with inputs $X = (X^1, \ldots, X^m)$. Without loss of generality, assume that X is uniformly distributed on the unit cube $(0, 1]^m$ (Loh, 1996). For $k = 1, \ldots, m$, let F_j denote the uniform distribution for X^k, and let $F = \prod_{k=1}^m F_j$. The issue is to generate n runs $X_1, \ldots, X_n$ from F and to estimate μ by

$$\hat{\mu} = n^{-1} \sum_{i=1}^{n} f(X_i). \tag{1.1}$$

This issue motivated McKay et al. (1979) to propose Latin hypercube designs, referred to as *ordinary Latin hypercube designs* hereinafter, to improve upon the simple Monte Carlo method. Throughout, a uniform permutation of a set of n elements

means a random permutation of the set with all $n!$ possibilities equal probable, and define

$$Z_n = \{1, \ldots, n\}, \text{ for an integer } n \geq 1. \tag{1.2}$$

An $n \times q$ ordinary Latin hypercube design $D = (x_{ik})$ with n runs on $(0, 1]^q$ is generated through

$$X_i^k = \frac{a_{ik} - u_{ik}}{n}, \quad i = 1, \ldots, n, \; k = 1, \ldots, q. \tag{1.3}$$

Here, x_{ik} is the level of factor k on the ith run $\boldsymbol{x}_i$; $A = (a_{ik})$ is a random Latin hypercube in which each column is an independent *uniform permutation* on Z_n; the u_{ik} are independent $U(0, 1]$ random variables; and the a_{ik} and the u_{ik} are mutually independent. The design in (1.3) achieves maximum uniformity in univariate margins: when projected onto any dimension, precisely one point falls within each of the n equally spaced intervals of $(0, 1]$ given by $(0/n, 1/n], \ldots, ((n-1)/n, 1]$. Figure 1.1 provides an illustration for a 5×3 ordinary Latin hypercube design.

Sampling properties of ordinary Latin hypercube designs were studied in McKay et al. (1979), Stein (1987), Owen (1992) and Loh (1996). Here are some useful statistical properties of the design D in (1.3). First, the joint probability mass function for $a_{i_1 k}$ and $a_{i_2 k}$ is

$$P(a_{i_1 k} = b, a_{i_2 k} = c) = \frac{1}{n(n-1)}, \quad i_1 \neq i_2, \quad b \neq c \in Z_n. \tag{1.4}$$

Second, the runs of D are *exchangeable*.

Different classes of Latin hypercube designs can be constructed by replacing the uniform permutations by other permutations on Z_n in (1.2). Possibilities include

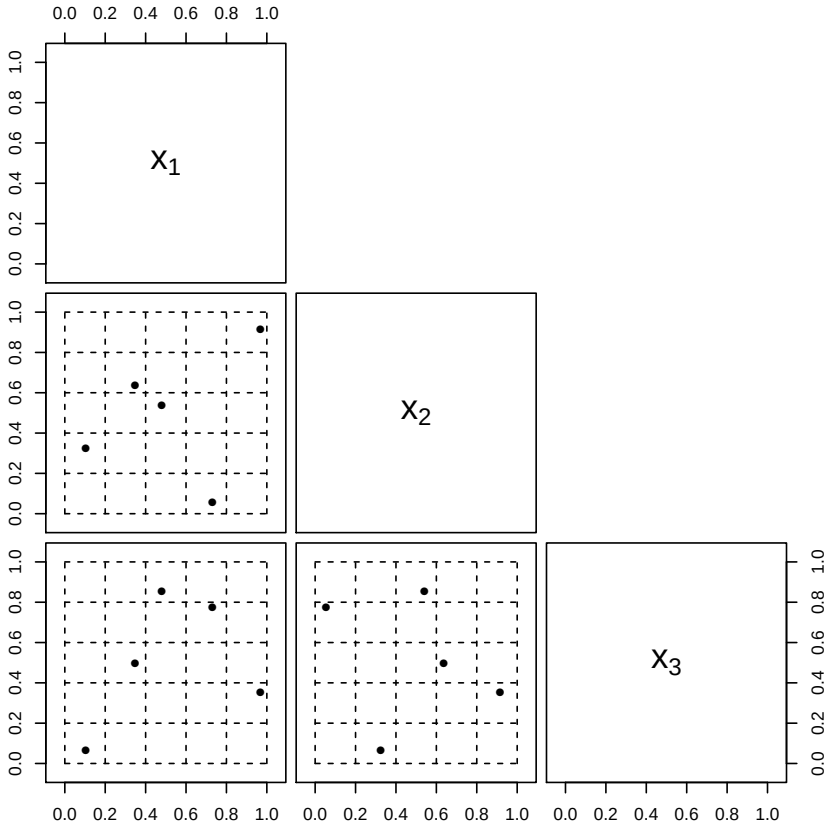

Figure 1.1: Bivariate projections of an ordinary Latin hypercube design.

controlling column correlations (Owen, 1994b; Tang, 1998; Ye, 1998; Butler, 2001; Steinberg and Lin, 2006; Bingham et al., 2009; Pang et al., 2009; Sun et al., 2009; Lin et al., 2009, 2010), maximizing distance (Morris and Mitchell, 1995b), achieving higher dimensional stratification (Tang, 1993; Owen, 1992, 1995), and combining distance and correlation criteria (Joseph and Hung, 2008), among others. To distinguish the ordinary Latin hypercube design and its extensions thus mentioned, we introduce the following definition.

Definition 1.1. *An $n \times m$ array A is a* Latin hypercube *if each column of A is a uniform permutation on Z_n. Moreover, A is an* ordinary Latin hypercube *if all its columns are generated independently. A Latin hypercube design D is ordinary if it is constructed based on an ordinary Latin hypercube.*

1.2 Functional ANOVA Decomposition

In order to understand the asymptotic properties of the integral estimator in (1.1) based on a Latin hypercube design, it is necessary to review the *functional ANOVA* (analysis of variance decomposition (Owen, 1994b). Let $\mathcal{D} = \{1, \ldots, m\}$ represent the axes of $(0,1]^m$ associated with an input vector $X = (X^1, \ldots, X^m)$. For $u \subseteq \mathcal{D}$, let X^u denote a vector consisting of X^k for $k \in u$. Define

$$f_u(X^u) = \int \{f(X) - \sum_{v \subset u} f_v(X^v)\} dF_{\mathcal{D}-u},$$

where $dF_{\mathcal{D}-u} = \prod_{k \notin u} dF_k$ integrates out all components except those included in u, and $v \subset u$ contains only proper subsets $v \neq u$. Hence $f_\emptyset(X^\emptyset) = \int f(X) dF$ is the mean of $f(X)$, $f_{\{k\}}(X^k) = \int \{f(X) - f_\emptyset(X^\emptyset)\} dF_{\mathcal{D}-\{k\}}$ is the main effect function for factor k and $f_{\{k,l\}}(X^k, X^l) = \int \{f(X) - f_{\{k\}}(X^k) - f_{\{l\}}(X^l) - f_\emptyset(X^\emptyset)\} dF_{\mathcal{D}-\{k,l\}}$ is the bivariate interaction function between factors k and l.

Using the main effect functions and bivariate interaction functions, write

$$f(X) = \mu + \sum_{k=1}^{m} f_{\{k\}}(X^k) + \sum_{k<l}^{m} f_{\{k,l\}}(X^k, X^l) + r(X), \tag{1.5}$$

where $r(X)$ is the residual function. Express $\hat{\mu}$ in (1.1) as

$$\hat{\mu} = \mu + n^{-1}\sum_{k=1}^{m}\sum_{i=1}^{n} f_{\{k\}}(X_i^k) + n^{-1}\sum_{k<l}^{m}\sum_{i=1}^{n} f_{\{k,l\}}(X_i^k, X_i^l) + n^{-1}\sum_{i=1}^{n} r(X_i). \quad (1.6)$$

We use the subscripts IID and LH to denote Monte Carlo sampling and sampling with respect to ordinary Latin hypercube designs, respectively. Note that

$$var_{\mathrm{IID}}(\hat{\mu}) = n^{-1}\sum_{k=1}^{m}\int f_{\{k\}}(X^k)^2 dF + n^{-1}\sum_{k<l}^{m}\int f_{\{k,l\}}(X^k, X^l)^2 dF + n^{-1}\int r(X)^2 dF,$$

which consists of the variance components from the main effects, the bivariate interactions and the residual $r(X)$ in (1.5).

Stein (1987) showed that

$$var_{\mathrm{LH}}(\hat{\mu}) = n^{-1}\sum_{k<l}^{m}\int f_{\{k,l\}}(X^k, X^l)^2 dF + n^{-1}\int r(X)^2 dF + o(n^{-1}), \quad (1.7)$$

where the variance components due to the main effect functions are filtered out.

In the following sections, I will briefly review two types of techniques which also generate Latin hypercube designs with better in variance reduction.

1.3 Orthogonal Array-based Latin Hypercube Designs

The first technique is to generate Latin hypercube designs based on orthogonal arrays which originate from the pioneering work of Rao (1946, 1947, 1949). Patterson (1954)

introduced lattice sampling based on randomized orthogonal arrays, which is found to be suitable for computer experiments and other related fields (Owen, 1992). Tang (1993) and Owen (1994b) independently studied the Monte Carlo variance of means over orthogonal-array-based Latin hypercube designs and randomized orthogonal arrays, respectively.

We introduce some properties and notation for orthogonal arrays that pertain to Latin hypercube designs. To be consistent with the notation for a Latin hypercube A in § 1.1, we denote the levels by $Z_s = \{1, 2, \ldots, s\}$. The following definition of orthogonal arrays is due to Hedayat et al. (1999).

Definition 1.2. *An $n \times m$ array $A = (a_{ik})$ with entries from Z_s is said to be an orthogonal array with s levels, strength τ and index λ (for some tau satisfying $0 \leq \tau \leq m$) if every $n \times \tau$ subarray of A contains each $\tau-$tuple based on Z_s exactly λ times as a row.*

Let $\mathrm{OA}(n, m, s, \tau)$ denote an orthogonal array, where n, m, s, τ, and λ are all integers, with $n = \lambda s^{\tau}$. An ordinary Latin hypercube is an orthogonal array with $\tau = 1$.

There exist many methods for constructing orthogonal arrays with $\tau = 2$ using Galois fields and finite geometry and other techniques. We summarize some popular constructions and their restrictions for n, m, s, and λ in Table 1.1.

Table 1.2 presents two strength-two orthogonal arrays. Both have sixteen scenarios and five columns, the left one having four levels while the right one has just two levels. Intuitively, the left one seems preferable, as it includes all 16 possible level combinations in any two columns. Owen (1994b) defines the concept of coincidence

Table 1.1: Methods of constructing orthogonal arrays with strength two

Method	Orthogonal Array	Restrictions
Bush (1952)	$\mathrm{OA}(s^2, m, s, 2)$	$m \leq s+1$ where s is a prime power
Bose and Bush (1952)	$\mathrm{OA}(\lambda s^2, m, s, 2)$	$m \leq \lambda s+1$ where s and λ are powers of the same prime
Addelman and Kempthorne (1961)	$\mathrm{OA}(2s^2, m, s, 2)$	$m \leq 2s+1$ where s is an odd prime power
Addelman and Kempthorne (1961)	$\mathrm{OA}(2s^3, m, s, 2)$	$m \leq 2s^2+2s+1$ where s is an odd prime power, 2 or 4

defect, which can be used to more formally justify the superiority of the left array which has more levels. An orthogonal array A with strength τ has *coincidence defect* if there exist two rows of A that agree in $\tau+1$ columns. The left array in Table 1.2 does not have coincidence defect because no two rows of A agree in more than a single column. The right one contains coincidence defects; for example, the second and the third rows agree in columns 2, 3, and 4.

We give further specifics. As in Owen (1994b), we define $\omega_{ij}(u)$ for each $u \subseteq \mathcal{D} := \{1, 2, \ldots, m\}$ as follows: $\omega_{ij}(u) := \{k \in u | a_{ik} = a_{jk}\}$. Define

$$M(u, r) := \sum_{i=1}^{n} \sum_{j=1}^{n} 1_{|\omega_{ij}(u)|=r}, \tag{1.8}$$

for $u \subseteq \mathcal{D}$ and $r = 0, 1, \ldots, |u|$. $M(u, r)$ is the number of pairs of ith and jth rows in an orthogonal array C (i can be the same as j) such that c_i and c_j agree on exactly r of the axes in u. For an $\mathrm{OA}(n, m, s, 2)$ without coincidence defect, $M(u, 3) = n$

Table 1.2: Two Orthogonal Arrays

An OA$(16, 5, 4, 1)$					
Scenarios#	a^1	a^2	a^3	a^4	a^5
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

An OA$(16, 5, 2, 1)$					
Senarios#	a^1	a^2	a^3	a^4	a^5
1	1	1	1	1	1
2	1	2	2	2	1
3	2	2	2	2	2
4	2	1	1	1	2
5	2	1	1	2	2
6	2	2	2	1	2
7	1	2	2	1	1
8	1	1	1	2	1
9	1	1	2	1	2
10	2	2	1	1	1
11	2	2	1	2	1
12	1	1	2	2	2
13	2	1	2	2	1
14	1	2	1	2	2
15	1	2	1	1	2
16	2	1	2	1	1

for any $u \subseteq \mathcal{D}$. For the array on the right in Table 1.2, $M(u, 3)$ may be much larger than 16 — for example, $M(\{1, 2, 3\}, 3) = 3n$. In general, we would like to select the orthogonal array C with no coincidence defect such that there are no duplicates in any three columns of C. If we are forced to use orthogonal arrays with coincidence defect, we should pick the one with the smallest value of $M(u, r)$. Discussion on the existence of coincidence defects for orthogonal arrays constructed using the methods of Table 1.1 can be found in Owen (1994b).

We focus on a specific type of orthogonal array-based Latin hypercube designs proposed by (Tang, 1993), referred to as U designs hereinafter. As described in Algorithm 1.1, an $n \times m$ U design E is constructed in three steps (Tang, 1993).

Algorithm 1.1 Generating a U Design

Step 1. Randomize the rows, columns and symbols of an OA(n, m, s, t) to get an array $A = (a_{ik})_{n\times m}$ with columns $a_1, \ldots, a_m$.

Step 2. Construct an array $B = (b_{ik})_{n\times m} = A$ with columns $b_1, \ldots, b_m$. Replace all the e's in b_k by a uniform permutation on Z_q for $e = 1, \ldots, s$.

Step 3. Construct an array $D = (d_{ik})_{n\times m}$ with columns $d_1, \ldots, d_m$, where $d_{ik} = q(a_{ik} - 1) + b_{ik}$. Obtain the (i, k)th entry of E as

$$X_i^k = \frac{d_{ik} - \eta_{ik}}{n}, \tag{1.9}$$

where the η_{ik} are $U[0, 1)$ random variables that are mutually independent of the d_{ik}.

Figure 1.2 depicts bivariate projections of a U design constructed by Algorithm 1.1 based on an OA(16,4,3,2). Each of the 16 equally spaced intervals of $(0, 1]$ contains exactly one point. Additionally, each of the 4×4 squares in the dashed lines has exactly one points. Compared with ordinary Latin hypercube designs in (1.3), U designs achieve additional two-dimensional stratification. For U sampling based on a strength two orthogonal array, Tang (1993) showed that

$$var_{\mathrm{U}}(\hat{\mu}) = n^{-1} \int r(X)^2 dF + o(n^{-1}), \tag{1.10}$$

which outperforms (1.7), as the main effects and bivariate interactions are filtered out.

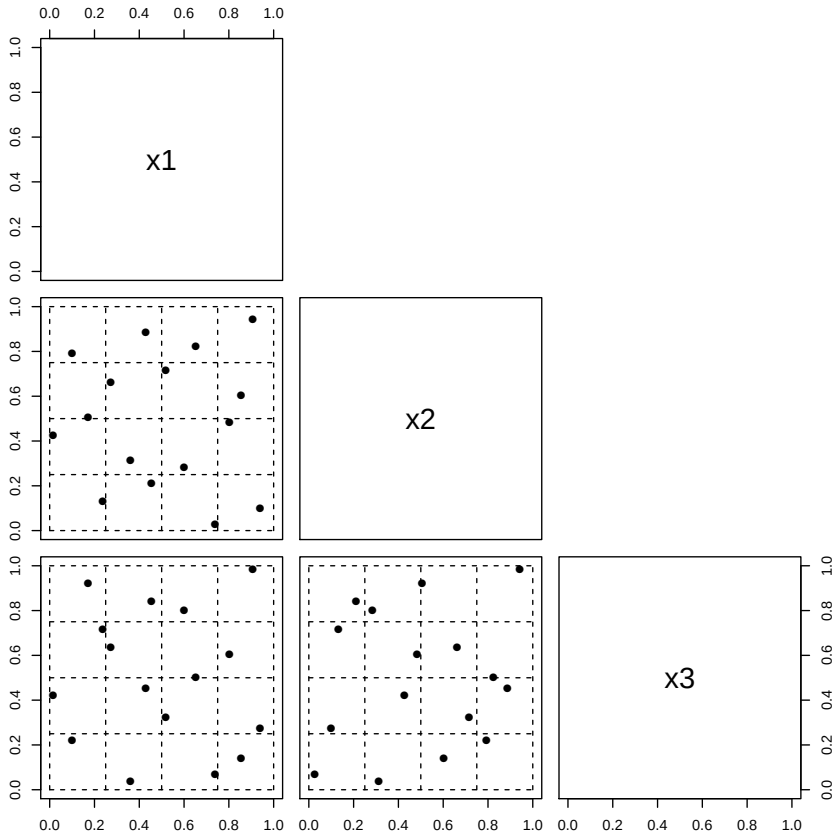

Figure 1.2: Bivariate projections of a U design.

1.4 Correlation-controlled Latin Hypercube Designs

Another way to improve (1.7) is to control column-wise correlations (Iman and Conover, 1982; Owen, 1994a; Tang, 1998). Decompose the bivariate interactions

$\sum_{k<l}^{m} f_{\{k,l\}}(X^k, X^l)$ of f in (1.5) to

$$\sum_{k<l}^{m} \gamma_{kl}(X^k - 1/2)(X^l - 1/2) + r'(X), \tag{1.11}$$

where $\sum_{k<l}^{m} \gamma_{kl}(X^k - 1/2)(X^l - 1/2)$ denotes the bilinear parts with bilinear coefficients

$$\gamma_{kl} = \frac{\int f_{\{k,l\}}(X^k, X^l)(X^k - 1/2)(X^l - 1/2)dF}{\int \{(X^k - 1/2)(X^l - 1/2)\}^2 dF}, \tag{1.12}$$

and $r'(X)$ in (1.11) is the bivariate residual from the bilinear parts. The γ_{kl} are chosen such that $r'(X)$ is orthogonal to all $(X^k - 1/2)(X^l - 1/2)$ in that $E\{r'(X)(X^k - 1/2)(X^l - 1/2)\} = 0$ for any $k \neq l$ (Owen, 1994a). Clearly, the $m(m-1)/2+1$ terms in (1.11), the m main effects functions and the residual $r(X)$ in (1.5) are pairwise orthogonal in that the product of any two terms has expectation zero.

Define

$$\rho_{kl} = n^{-1} \sum_{i=1}^{n} (X_i^k - 1/2)(X_i^l - 1/2), \tag{1.13}$$

which differs from the sample covariance between X^k and X^l as the sample mean of X^k is $1/2 + O_p(n^{-3/2})$ when $X_1^k, \ldots, X_n^k$ are sampled in (1.3).

Substituting (1.11) into (1.6) gives

$$\hat{\mu} = \mu + n^{-1} \sum_{k=1}^{m} \sum_{i=1}^{n} f_{\{k\}}(X_i^k) + \sum_{k<l}^{m} \gamma_{kl}\rho_{kl} + n^{-1} \sum_{i=1}^{n} r'(X_i) + n^{-1} \sum_{i=1}^{n} r(X_i). \tag{1.14}$$

If all the ρ_{kl} are controlled to be $o_p(n^{-1/2})$, the means of $r'(X_i)$ and $r(X_i)$ in (1.14) will dominate and the bilinear parts are filtered out.

Owen (1994a) introduced the RGS algorithm which can successfully generate a

correlation controlled Latin hypercube design. For simplicity, all the η_{ik} in (1.3) are taken to be .5 instead of independent $U[0,1)$ variables unless stated otherwise. Following the notation in Owen (1994a), let $\texttt{takeout}(y,z)$ denote the residual of a linear regression with an intercept, y as the values of the predictor and $\mathbf{z}$ as the values of the response. Let $\texttt{rank}(y)$ denote the vector of ranks of y. Starting with an $n \times p$ ordinary Latin hypercube design D, let x^j denote the jth column in D, the RGS algorithm proceeds by alternating a forward and backward steps as follows.

Algorithm 1.2 RGS Algorithm

Forward:

$$\begin{aligned}
&\text{for} \quad k = 2, ..., p \\
&\qquad \text{for} \quad \ell = 1, ..., k-1 \\
&\qquad\qquad x^\ell \leftarrow \texttt{takeout}(x^k, x^\ell) \\
&\text{for} \quad k = 1, ..., p \\
&\qquad x^k \leftarrow (\texttt{rank}(x^k) - .5)/n,
\end{aligned} \tag{1.15}$$

Backward:

$$\begin{aligned}
&\text{for} \quad k = p-1, ..., 1 \\
&\qquad \text{for} \quad \ell = p, ..., k+1 \\
&\qquad\qquad x^\ell \leftarrow \texttt{takeout}(x^k, x^\ell) \\
&\text{for} \quad k = 1, ..., p \\
&\qquad x^k \leftarrow (\texttt{rank}(x^k) - .5)/n.
\end{aligned} \tag{1.16}$$

Note the residual $\texttt{takeout}(x^k, x^\ell)$ has zero correlation with x^ℓ for any $k \neq \ell$. The `rank` function is necessary to force a Latin hypercube structure of being a permutation of $1, \ldots, n$. And as a result, $\texttt{rank}[\texttt{takeout}(x^k, x^\ell)]$ and x^ℓ will have a small correlation but not exactly zero for most cases.

Owen (1994a) concluded that when p is much smaller than n the number of complete alternations between forward and backward steps required for converges is

no more than five complete alternations and seems not to grow with n. To check the performance of the RGS algorithm, the root mean squared correlation among columns of $\mathbf{X}$ was proposed by Owen (1994a) as

$$\rho_{\text{rms}}(D) = \sqrt{\frac{\sum_{1\le k<\ell\le p}\rho_{k\ell}^2}{p(p-1)/2}}, \tag{1.17}$$

where $\rho_{k\ell}$ is, from now on, the sample correlation between $\mathbf{x}^k$ and $\mathbf{x}^\ell$ which should has the same order as the sample covariance in (1.13). Owen's numerical study revealed that $\rho_{rms}(\mathbf{X}) = O_p(n^{-1})$ when p is fixed, which means $\mathbf{X}$ becomes a correlation-controlled Latin hypercube design.

Now we summarize the reason given by Owen (1994a) to explain the performance of the RGS algorithm. Each update in a single `takeout` step is equivalent to

$$\mathbf{x}^\ell \to \mathbf{x}^\ell - (\mathbf{x}^k - .5)\rho_{k\ell}\sigma_k/\sigma_j, \tag{1.18}$$

where $\rho_{k\ell}$ is calculated before the update, σ_k and σ_ℓ are the standard deviations of the two columns vectors. The amount of change for each component in $\mathbf{x}^\ell$ is $O_p(\rho_{k\ell})$. Changing $\mathbf{x}^\ell$ means a change in $\texttt{rank}(\mathbf{x}^\ell)$, which requires modifying some components in $\mathbf{x}^\ell$ by $O(n^{-1})$. Assume p independent updates in (1.18) are made for $\mathbf{x}^\ell$ *before* re-rank that column, the RGS algorithm will stop updating when $\rho_{k\ell}$ is small compared to $n^{-1}p^{-1/2}$ for every $k \neq \ell$. This explains why $\rho_{rms} = O_p(n^{-1})$ when p is fixed. Clearly, one can get $\rho_{rms} = O_p(n^{-3/2})$ when $p = n-1$. Owen suggested use the RGS algorithm to get an $n \times n-1$ Latin hypercube design first, then randomly pick p columns among $n-1$ columns. In this work, we begin with an ordinary Latin

hypercube design of p columns . We choose fixed p over $p = n - 1$ for two reasons. First, $\rho_{rms} = O_p(n^{-1})$ with fixed p already guarantees a correlation-controlled Latin hypercube design. More importantly, the number operations need with fixed p is $O(nlog(n))$ while it is $O(n^3)$ with $p = n - 1$ (Tang, 1998).

Tang (1998) extended the RGS algorithm to further control quadratic polynomial canonical correlations. It was claimed that Owen's algorithm generally did better than Tang's algorithm in reducing the column-wise correlations due to the latter achieves additional quadratic correlations (Tang, 1998). However, we found the major reason could be that Tang re-ranked $\mathbf{x}^\ell$ *after* every single update. We make sure that the RGS algorithm in this work can replicate numerical results in Owen (1994a).

Chapter 2

Sequentially Refined Latin Hypercube Designs

2.1 Background

A sequential approach for numerical integration, referred to as *sequentially refined Latin hypercube designs* (SLHD) hereinafter, is used in computer experiments (Tong, 2006; Sallaberry et al., 2008; Qian, 2009), and implemented in software like Sandia National Labs stochastic toolkit DAKOTA (Adams et al., 2014). It runs a computer code with an ordinary Latin hypercube design in one shot. When more computational resources become available, the code is evaluated in another shot with runs chosen elaborately so that the augmented design of the two shots forms a larger Latin hypercube design. This augmenting process can be replicated multiple stages so that the points in all previous stages are reused and the augmented design in any stage

is a Latin hypercube design. This approach can be used for sequential uncertainty quantification in a non-intrusive manner (Xiu, 2010). Consider Figure 2.1 for illustration. In this figure, E_1 is an ordinary Latin hypercube design of five runs ('$\bullet$'); by augmenting E_2 of five runs ('$\triangle$') to E_1, the combined design, denoted by $E_1 \cup E_2$, forms a Latin hypercube design of ten runs; by augmenting E_3 of ten points ('$\square$') to $E_1 \cup E_2$, $E_1 \cup E_2 \cup E_3$ forms a Latin hypercube design of 20 runs. Methods for constructing E_2 and E_3 are given in Section 2.2. Both $E_1 \cup E_2$ and $E_1 \cup E_2 \cup E_3$ achieve maximum marginal uniformity: when the runs of the former are projected onto any dimension, precisely one point falls within each of the ten equally spaced intervals of $(0, 1]$ given by $(0, 1/10], \ldots, (9/10, 1]$, and when the points of the latter are projected onto any dimension, precisely one point falls within each of the 20 equally spaced intervals on $(0, 1]$ given by $(0, 1/20], \ldots, (19/20, 1]$.

2.2 Construction and Sampling Properties

This section provides some useful definitions and notation. For sets A and B, A/B denotes the subset of A lying outside B. Consider design construction for q continuous factors $\boldsymbol{x} = (x_1, \ldots, x_q)$ in $(0, 1]^q$. For $0 \leq z_1, z_2 \leq 1$ and an integer $p > 0$, define

$$\delta_p(z_1, z_2) = I(\lceil pz_1 \rceil = \lceil pz_2 \rceil), \tag{2.1}$$

where $I(\cdot)$ is the indicator function.

The key in sequential refinement of a Latin hypercube design is to adaptively add points to retain maximum one-dimensional stratification as the mesh size decreases.

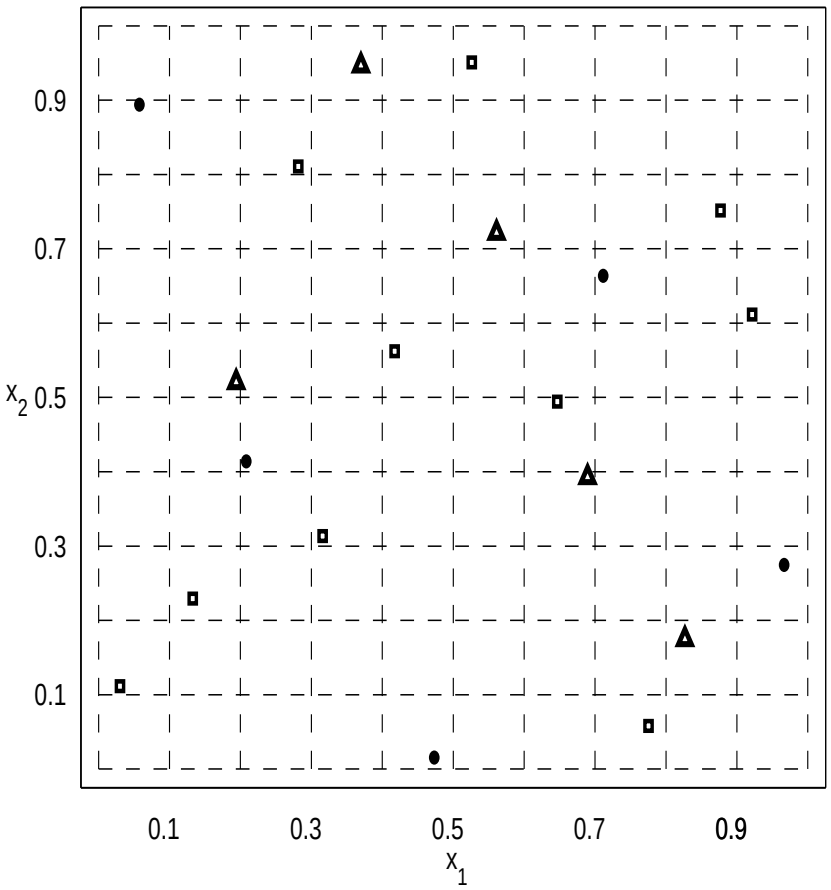

Figure 2.1: The '•'s denote an ordinary Latin hypercube design E_1 of five runs; augmenting five points of E_2 ('△') to E_1 gives a Latin hypercube design of 10 runs; and augmenting E_2 of ten points ('□') to $E_1 \cup E_2$ yields a Latin hypercube design of 20 runs.

Consider Figure 2.2 for illustration. In stage 1, the four 'X's are sampled from the four equally spaced intervals $(0, 1/4]$, $(1/4, 2/4]$, $(2/4, 3/4]$, $(3/4, 1]$ of $(0, 1]$, respectively. When these four intervals are divided into the eight equally spaced intervals $(0, 1/8]$,..., $(7/8, 1]$ of finer size, the four 'X's occupy $(1/8, 1/4]$, $(3/8, 1/2]$, $(1/2, 5/8]$,

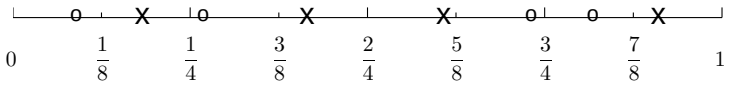

Figure 2.2: Combining four points ('X') in the four equally spaced intervals $(0, 1/4]$, $(1/4, 2/4]$, $(2/4, 3/4]$, $(3/4, 1]$ of $(0, 1]$ with four new points ('o') to fill all eight equally spaced intervals $(0, 1/8], \ldots, (7/8, 1]$.

$(7/8, 1]$. So in stage 2, four new points ('o') are added to fill the four unoccupied intervals $(0, 1/8]$, $(1/4, 3/8]$, $(5/8, 6/8]$, $(3/4, 7/8]$.

Specifically, sequential construction of Latin hypercube designs of q factors in u stages is as follows. In stage 1, obtain $E_1 = (e_{ik})$ as an $n_1 \times q$ ordinary Latin hypercube design from (1.3). Let $n_0 = 0$ and

$$n_j = n_1 \prod_{i=2}^{j} s_i \tag{2.2}$$

for $j = 2, \ldots, u$ and integers $s_2, \ldots, s_u \geq 2$. Note that s_i's in (2.2) represent the 'multiplicative factors', which are fixed numbers determined by external constrains (resources, etc.) in each stage. For $j = 1, \ldots, u$, let

$$P_j = \left\{ \left(\frac{i-1}{n_j}, \frac{i}{n_j} \right] : \ i = 1, \ldots, n_j \right\} \tag{2.3}$$

denote the n_j equally spaced intervals of $(0, 1]$. For $j = 2, \ldots, u$ and $k = 1, \ldots, q$, let

$$C_{j,k} = \{ \lceil n_j e_{ik} \rceil : i = 1, \ldots, n_{j-1} \} \tag{2.4}$$

and

$$\Gamma_{j,k} = Z_{n_j}/C_{j,k}, \tag{2.5}$$

where Z_{n_j} is defined in (1.2). Essentially, $C_{j,k}$ consists of the n_{j-1} integers of Z_{n_j} corresponding to the intervals in P_j in which $e_{1,k}$, ..., $e_{n_{j-1},k}$ are located and $\Gamma_{j,k}$ consists of the $n_j - n_{j-1}$ intervals in P_j containing no point in stages 1 to $j-1$. In stage j, generate an $(n_j - n_{j-1}) \times q$ array $E_j = (e_{ik})$ through

$$e_{ik} = \frac{b_{ik} - u_{ik}}{n_j}, \quad i = 1, \ldots, n_j - n_{j-1},\ k = 1, \ldots, q, \tag{2.6}$$

where $b_{1,k}, \ldots, b_{n_j - n_{j-1},k}$ form a uniform permutation on $\Gamma_{j,k}$, the u_{ik} are independent $U[0,1)$ random variables, and the b_{ik} and the u_{ik} are mutually independent.

Using $E_1, \ldots, E_u$ in (2.6), a computer model can be run in u stages. For $j = 1, \ldots, u$, the accumulated design set of stages 1 to j is

$$D_j = \cup_{i=1}^{j} E_i. \tag{2.7}$$

Proposition 2.1. *Consider $D_1, \ldots, D_u$ constructed above. We have that (i) $D_1 \subset \cdots \subset D_u$ and (ii) for $j = 1, \ldots, u$, D_j is a Latin hypercube design of n_j runs.*

When projected onto any dimension, precisely one point in D_j falls into one of the n_j equally spaced intervals on $(0,1]$ given by $(0, 1/n_j], \ldots, ((n_j - 1)/n_j, 1]$.

Example 2.2. *Figure 2.3 depicts three sequentially refined Latin hypercube designs $D_1 \subset D_2 \subset D_3$ in three factors constructed from (2.7) with $n_1 = 2$ and $s_2 = s_3 = 2$ which implies $n_2 = 4$ and $n_3 = 8$. In any one-dimensional projection, the two runs*

of D_1 in stage 1 ('•') fall into the two equally spaced intervals of length $1/2$, respectively. Stage 2 augments E_2 of two runs ('△') to D_1 to give a Latin hypercube D_2 of four points, where precisely one point of D_2 falls into one of the four equally spaced intervals given by $(0, 1/4], \ldots, (3/4, 1]$ in any one-dimensional projection. Stage 3 augments E_2 of four runs ('□') to D_2 to form D_3, where precisely each point of D_3 falls into one of the eight equally spaced intervals given by $(0, 1/8], \ldots, (7/8, 1]$ in any one-dimensional projection.

We now connect the sequentially refined Latin hypercube designs constructed in (2.7) to those obtained by using nested permutations introduced in Section 5 of (Qian, 2009).

Define

$$t_j = \prod_{i=j+1}^{u} s_i \quad \text{for} \quad j = 1, \ldots, u-1 \quad \text{and} \quad t_u = 1, \tag{2.8}$$

where the s_i are defined in (2.2). A *nested permutation* $\boldsymbol{\pi}_u = \{\pi(1), \ldots, \pi(n_u)\}$ with u layers is a special permutation on Z_{n_u} in (1.2), where, concerning the collection of the first n_j elements, $\{\lceil \pi(1)/t_j \rceil, \ldots, \lceil \pi(n_j)/t_j \rceil\}$ is a permutation on Z_{n_j}. Such a permutation can be generated by the following algorithm:

Step 1: Draw a uniform permutation

$$\boldsymbol{\zeta}_j = \{\zeta_j(1), \ldots, \zeta_j(n_j - n_{j-1})\} \tag{2.9}$$

on Z_{n_j}/C_j, where C_1 is the empty set, $n_0 = 0$ and for $j \geq 2$,

$$C_j = \{\lceil \pi(1)/t_j \rceil, \ldots, \lceil \pi(n_{j-1})/t_j \rceil\}. \tag{2.10}$$

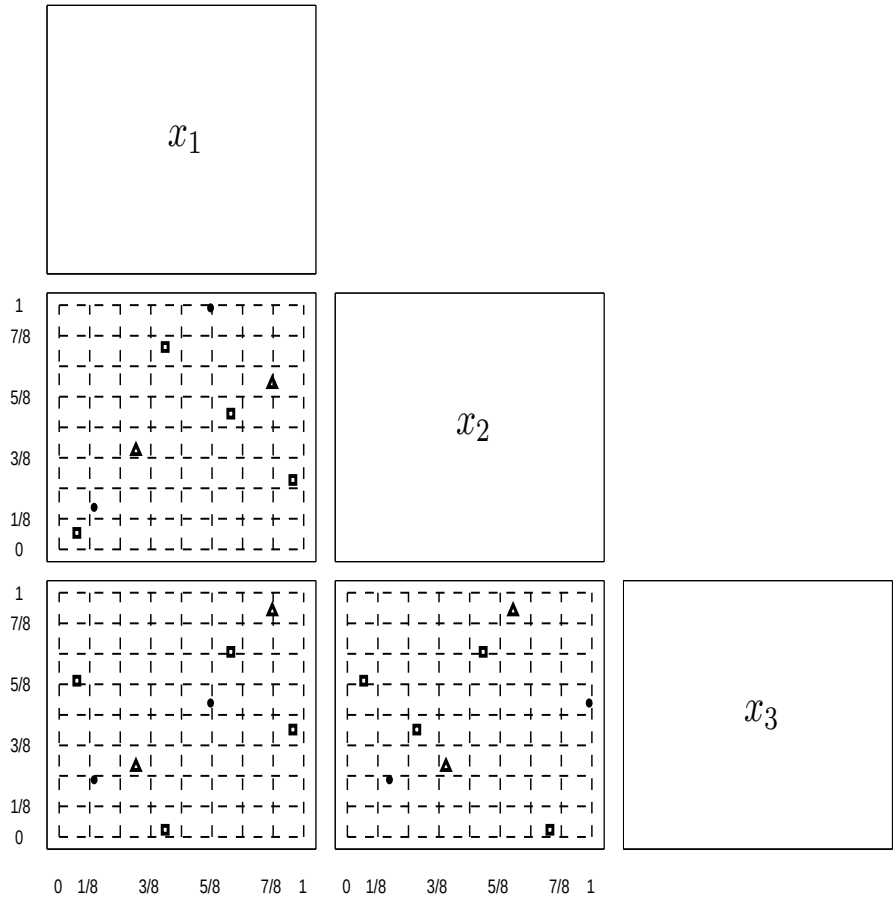

Figure 2.3: Bivariate projections of three sequentially refined Latin hypercube designs $D_1 \subset D_2 \subset D_3$ in three factors for Example 2.2 with $n_1 = 2$, $n_2 = 4$ and $n_3 = 8$. Here, D_1 is an ordinary Latin hypercube design of two runs ('$\bullet$'). Augmenting two runs ('$\triangle$') to D_1 gives a Latin hypercube design D_2 of four runs. Augmenting four runs ('$\square$') to D_2 produces a Latin hypercube design D_3 of eight runs.

Step 2: For $i = n_{j-1}+1, \ldots, n_j$, sample $\pi(i)$ from a *discrete uniform distribution* with support on t_j consecutive integers from $\{\zeta_j(i-n_{j-1})-1\}t_j+1$ to $\zeta_j(i-n_{j-1})t_j$.

Let $A = (a_{ik})$ be an $n_u \times q$ *nested Latin hypercube* in which each column is a nested permutation with u layers produced by the foregoing algorithm, and all columns are generated independently. Then an $n_u \times q$ design $D_u = (X_i^k)$ is generated through

$$X_i^k = \frac{a_{ik} - u_{ik}}{n_u}, \quad i = 1, \ldots, n_u,\ k = 1, \ldots, q, \tag{2.11}$$

where X_i^k is the level of the kth factor on the ith run, the u_{ik} are independent $U[0,1)$ random variables, and the a_{ik} and the u_{ik} are mutually independent. For $j = 1, \ldots, u-1$, let D_j be the subarray of D_u consisting of the first n_j runs.

Proposition 2.3. *Consider $D_1, \ldots, D_u$ constructed above using nested permutations $\boldsymbol{\pi}_u$ with u layers. We have that (i) $D_1 \subset \cdots \subset D_u$ and (ii) for $j = 1, \ldots, u$, D_j is a Latin hypercube design of n_j levels.*

Proposition 2.4 shows that the designs from Propositions 2.1 and those from Propositions 2.3 are statistically equivalent.

Proposition 2.4. *For the same $n_1, \ldots, n_u$ in (2.2), $D_1 \subset \cdots \subset D_u$ obtained by (2.11) has exactly the same distribution as its counterpart from the sequential construction in (2.7).*

Next, we derive some joint probability mass functions of nested permutations with an arbitrary number of layers. Because of the lack of exchangeability among all

elements of $\boldsymbol{\pi}_u$, these functions are much more complicated than those of uniform permutations for ordinary Latin hypercube designs in (1.4).

Proposition 2.5. *Let $\boldsymbol{\pi}_u$ be a nested permutation with u layers, with the n_j and the t_j defined in (2.2) and (2.8), respectively. For $j = 1, \ldots, u$, $i = 1, \ldots, n_j - n_{j-1}$, let $\tau_{j,i}$ be the $(n_{j-1} + i)$th element of $\boldsymbol{\pi}_u$. For $a, b \in Z_{n_u}$, we have that*

(i) for $j = 1, \ldots, u$ and $i = 1, \ldots, n_j - n_{j-1}$, the probability mass function for $\tau_{j,i}$ is

$$P(\tau_{j,i} = a) = n_u^{-1};$$

(ii) for $j = 1, \ldots, u$ and $i_1, i_2 = 1, \ldots, n_j - n_{j-1}$, the joint probability mass function for τ_{j,i_1} and τ_{j,i_2} is

$$P(\tau_{j,i_1} = a, \tau_{j,i_2} = b) = \begin{cases} \frac{t_{j-1} - 2t_j}{n_u\{n_u - t_j(n_{j-1}+1)\}(t_{j-1} - t_j)}, & \lceil a/t_{j-1} \rceil = \lceil b/t_{j-1} \rceil, \lceil a/t_j \rceil \neq \lceil b/t_j \rceil, \\ \frac{t_{j-1} - t_j}{n_u\{n_u - t_j(n_{j-1}+1)\}t_{j-1}}, & \lceil a/t_{j-1} \rceil \neq \lceil b/t_{j-1} \rceil, \lceil a/t_j \rceil \neq \lceil b/t_j \rceil, \\ 0, & \lceil a/t_j \rceil = \lceil b/t_j \rceil; \end{cases}$$

(iii) for $j, \ell = 1, \ldots, u$ with $j < \ell$ and $i_1 = 1, \ldots, n_j - n_{j-1}$, $i_2 = 1, \ldots, n_\ell - n_{\ell-1}$,

the joint probability mass function for τ_{j,i_1} *and* τ_{ℓ,i_2} *is*

$$P(\tau_{j,i_1}=a,\tau_{\ell,i_2}=b)=\begin{cases}\frac{1}{n_u(n_u-t_\ell n_{\ell-1})}, & \lceil a/t_{\ell-1}\rceil=\lceil b/t_{\ell-1}\rceil, \lceil a/t_\ell\rceil\neq\lceil b/t_\ell\rceil,\\ n_u^{-2}, & \lceil a/t_{\ell-1}\rceil\neq\lceil b/t_{\ell-1}\rceil, \lceil a/t_\ell\rceil\neq\lceil b/t_\ell\rceil,\\ 0, & \lceil a/t_\ell\rceil=\lceil b/t_\ell\rceil.\end{cases}$$

Let $n_0=0$ and $t_0=\infty$ in (ii) and (iii) of Proposition 2.5. Proposition 2.5 reduces to (1.3) for ordinary Latin hypercube designs when $u=1$ and to Proposition 1 of (Qian, 2009) for two nested Latin hypercube designs when $u=2$. The proof is deferred in Appendix.

The nested permutations are closely related to Owen's random permutation for scrambled nets (Owen, 1997).

Take $D_1\subset\cdots\subset D_u$ from (2.11). By Proposition 2.5, the joint density function of two different runs $\boldsymbol{x}_{i_1}$ and $\boldsymbol{x}_{i_2}$ of D_u is

$$\begin{cases}\left(\frac{n_j}{n_j-n_{j-1}-1}\right)^q\prod_{k=1}^q\{c_{j,0}-c_{j,1}\delta_{n_{j-1}}(x_{i_1,k},x_{i_2,k})-c_{j,2}\delta_{n_j}(x_{i_1,k},x_{i_2,k})\},\\ \qquad\qquad i_1,i_2=n_{j-1}+1,\ldots,n_j,\\ \left(\frac{n_\ell}{n_\ell-n_{\ell-1}}\right)^q\prod_{k=1}^q\{e_{\ell,0}-e_{\ell,1}\delta_{n_{\ell-1}}(x_{i_1,k},x_{i_2,k})-e_{\ell,2}\delta_{n_\ell}(x_{i_1,k},x_{i_2,k})\},\\ \qquad i_1=n_{j-1}+1,\ldots,n_j,\ i_2=n_{\ell-1}+1,\ldots,n_\ell,\ j<\ell.\end{cases}\tag{2.12}$$

Here, δ is defined in (2.1); $c_{1,0}=1$, $c_{1,1}=0$, and $c_{1,2}=1$; for $j,\ell\geq 2$, $c_{j,0}=1-s_j^{-1}$, $c_{j,1}=s_j^{-1}(s_j-1)^{-1}$, and $c_{j,2}=(s_j-2)(s_j-1)^{-1}$; $e_{\ell,0}=1-s_\ell^{-1}$, $e_{\ell,1}=-s_\ell^{-1}$, and $e_{\ell,2}=1$. By the second part of (2.12), $\boldsymbol{x}_{i_1}$ and $\boldsymbol{x}_{i_2}$ from two different sets E_j and E_ℓ

with $j < \ell$ are conditionally independent if $\delta_{n_{\ell-1}}(x_{i_1,k}, x_{i_2,k}) = 0$ for $k = 1, \ldots, q$.

The theorem of (McKay et al., 1979) states that two different entries of the same column of an ordinary Latin hypercube design are *negatively quadrant dependent*. A pair of random variables (X, Y) is negatively quadrant dependent if $P(X \leq x, Y \leq y) \leq P(X \leq x)P(Y \leq y)$ for all x and y (Lehmann, 1966). For ordinary Latin hypercube designs, negative quadrant dependence is critical to achieving variance reduction. Proposition 2.6 gives an analogous result for sequentially refined Latin hypercube designs.

Proposition 2.6. *Let $D_1 \subset \cdots \subset D_u$ be the sequentially refined Latin hypercube designs constructed by (2.11). Let $\boldsymbol{x}_{i_1} = (x_{i_1,1}, \ldots, x_{i_1,q})$ and $\boldsymbol{x}_{i_2} = (x_{i_2,1}, \ldots, x_{i_2,q})$ be two different runs of D_u. Then, for $k = 1, \ldots, q$, $x_{i_1,k}$ and $x_{i_2,k}$ are negatively quadrant dependent. That is, for $0 \leq v_1, v_2 < 1$,*

$$P(x_{i_1,k} \leq v_1, x_{i_2,k} \leq v_2) \leq P(x_{i_1,k} \leq v_1)P(x_{i_2,k} \leq v_2). \tag{2.13}$$

Proposition 2.6 shows that when the uniform permutations are replaced by the nested permutations in constructing Latin hypercube designs, negatively quadrant dependence still holds.

Now consider the problem of estimating the expected output μ in (1.1) of a computer code f in u stages. For $j = 1, \ldots, u$, let D_j be the set of input values for stage j in (2.7) and let $E_j = D_j/D_{j-1}$ be the jth augmenting set where D_0 is the empty set. Definition 2.7 describes four different schemes to generate $D_1, \ldots, D_u$.

Definition 2.7. *For $n_1, \ldots, n_u$ defined in (2.2) and $j = 1, \ldots, u$,*

(i) let IID denote a scheme that takes E_j to be an independent and identically distributed sample of $n_j - n_{j-1}$ runs;

(ii) let LH denote a scheme that takes D_j to be independent ordinary Latin hypercube design of n_j runs;

(iii) let IL denote a scheme that takes E_j to be an independent ordinary Latin hypercube design of $n_j - n_{j-1}$ runs;

(iv) let SL denote a scheme that takes $D_1 \subset \cdots \subset D_j$ to be the sequentially refined Latin hypercube designs constructed by (2.11).

The LH scheme needs $n_1 + \cdots + n_{u-1}$ more runs than the SL scheme. The scheme in (iv) is referred to as *sequential Latin hypercube sampling*. For each scheme, μ is estimated by

$$\hat{\mu}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} f(\boldsymbol{x}_i), \quad \text{for} \quad j = 1, \ldots, u, \tag{2.14}$$

where $\boldsymbol{x}_i$ is the ith run of D_j. We use the subscripts IID, LH, IL and SL to denote these four schemes, respectively.

We now derive variance formulas for sequential Latin hypercube sampling. Theorem 2.8 presents a result under some monotonicity assumption on f, providing an analogy to the theorem in (McKay et al., 1979) for ordinary Latin hypercube designs.

Theorem 2.8. *If $f(\boldsymbol{x})$ is monotonic in each of its arguments, then $var_{SL}(\hat{\mu}_j) \leq var_{IID}(\hat{\mu}_j)$ for $j = 1, \ldots, u$.*

Theorem 2.9 further shows that sequentially refined Latin hypercube designs always outperform the scheme that takes independent ordinary Latin hypercube

designs in a finite sample sense when the computer code f is either monotonic or additive.

Theorem 2.9. *Assume $s_i = 2$ for $i = 2, \ldots, u$. We have $var_{SL}(\hat{\mu}_j) \leq var_{IL}(\hat{\mu}_j) \leq var_{IID}(\hat{\mu}_j)$ for $j = 1, \ldots, u$ if* (i) *$f(\boldsymbol{x})$ is monotonic in each of its arguments; or* (ii) *$E\{f(\boldsymbol{x})\}$ is finite and $f(\boldsymbol{x}) = \mu + \sum_{k=1}^{q} f_k(x_k)$.*

2.3 Controlling correlations in sequentially refined Latin hypercube designs

Similar as (1.17), for a sequentially refined Latin hypercube design D_j in (2.7), we use the root mean square of correlation among columns of D_j defined in (Owen, 1994b) as the overall correlation criterion,

$$\rho^2_{\text{rms}}(D_j) = \frac{2\sum_{1\leq k<\ell\leq q}\rho^2_{j,k\ell}}{q(q-1)}, \tag{2.15}$$

where $\rho_{j,k\ell}$ is defined as the sample correlation between the kth and ℓth columns of D_j. Our algorithm adjusts the augmenting design E_j in (2.6) according to the column-wise correlations of D_{j-1} such that $\rho^2_{\text{rms}}(D_j)$ is controlled after combining D_{j-1} and E_j. Let $\boldsymbol{x}_j^k$ be the vector of the kth column in E_j. Let $\texttt{takeout}(\boldsymbol{y}, \boldsymbol{z})$ denote the residual of a linear regression model, which has an intercept and the values of the independent variable are given in vector $\boldsymbol{y}$ and the values of responses in vector $\boldsymbol{z}$. Let $\texttt{rank}(\boldsymbol{x})$ denote the vector of ranks of the corresponding elements in $\boldsymbol{x}$. The algorithm has the following steps.

Algorithm 2.1 Generating a Correlation-controlled Sequentially Refined Latin Hypercube Design

Step 1: Obtain $\Gamma_{j,k}$ defined in (2.5) and generate E_j through (2.6). Let $D_j = D_{j-1} \cup E_j$. Compute $\rho^2_{\text{rms}}(D_j)$ in (2.15), and $\rho_{j-1,k\ell}$ of D_{j-1} for $1 \le k < \ell \le q$.

Step 2:

Forward:

$$\begin{aligned} &\text{for} \quad k = 2, ..., q \\ &\qquad \text{for} \quad \ell = 1, ..., k-1 \\ &\qquad\qquad \boldsymbol{x}_j^\ell \leftarrow \texttt{takeout}(\boldsymbol{x}_j^k, \boldsymbol{x}_j^\ell) - (s_j - 1)^{-1} \rho_{j-1,k\ell} \boldsymbol{x}_j^k \qquad (2.16) \\ &\text{for} \quad k = 1, ..., q \\ &\qquad \boldsymbol{x}_j^k \leftarrow (\Gamma_{j,k}\{\texttt{rank}(\boldsymbol{x}_j^k)\} - \boldsymbol{u}_k)/n_j, \end{aligned}$$

Backward:

$$\begin{aligned} &\text{for} \quad k = q-1, ..., 1 \\ &\qquad \text{for} \quad \ell = q, ..., k+1 \\ &\qquad\qquad \boldsymbol{x}_j^\ell \leftarrow \texttt{takeout}(\boldsymbol{x}_j^k, \boldsymbol{x}_j^\ell) - (s_j - 1)^{-1} \rho_{j-1,k\ell} \boldsymbol{x}_j^k \qquad (2.17) \\ &\text{for} \quad k = 1, ..., q \\ &\qquad \boldsymbol{x}_j^k \leftarrow (\Gamma_{j,k}\{\texttt{rank}(\boldsymbol{x}_j^k)\} - \boldsymbol{u}_k)/n_j, \end{aligned}$$

where '$\leftarrow$' denotes assignment, the vector $\Gamma_{j,k}\{\texttt{rank}(\boldsymbol{x}_j^k)\}$ is a permutation on $\Gamma_{j,k}$ such that $\texttt{rank}[\Gamma_{j,k}\{\texttt{rank}(\boldsymbol{x}_j^k)\}] = \texttt{rank}(\boldsymbol{x}_j^k)$, $\boldsymbol{u}_k$ is a vector of independent $U(0,1]$ variables of the same length as $\boldsymbol{x}_j^k$.

Step 3: Update E_j, D_j and $\rho^2_{\text{rms}}(D_j)$.

Step 4: Stop the algorithm if $\rho^2_{\text{rms}}(D_j)$ converges or the number of iterations reaches a pre-specified value; go to step 2, otherwise.

Unlike Owen's algorithm, the proposed algorithm includes an extra term in (2.16) and (2.17) to guarantee the correlation between $\boldsymbol{x}_j^k$ and $\boldsymbol{x}_j^\ell$ of E_j is approximately $-(s_j-1)^{-1}\rho_{j-1,k\ell}$. As a result, the sample correlation between the kth and ℓth columns of $D_j = D_{j-1} \cup E_j$ will be close to zero since $\rho_{j,k\ell} \approx n_{j-1}\rho_{j-1,k\ell} + (n_j - n_{j-1})\{-(s_j-1)^{-1}\rho_{j-1,k\ell}\} = 0$.

For $n_1, \ldots, n_u$ in (2.2) and $j = 1, \ldots, u$, let SL-CC denote a scheme that takes $D_1 \subset \cdots \subset D_j$ to be the sequentially refined Latin hypercube designs with correlations controlled by the above algorithm.

2.4 Numerical illustration

In this section, we provide numerical examples to corroborate the theoretical results in Sections 2.2. We compare the performances of the estimator $\hat{\mu}_j$ in (2.14) for each stage under the four schemes in Definition 2.7 of Section 2.2 and the SL-CC scheme defined in Section 2.3. We include the schemes that apply correlation control algorithm to each E_j in IL and each D_j in LH and denote them by IL-CC and LH-CC, respectively. The maximum number of iterations of Step 2 in the algorithm is 10. We also consider a method based on scrambled Sobol sequences (Niederreiter, 1992; Owen, 1995) denoted by Sobol in the comparison generated by using the functions `sobol` and `scramble` with the option `MatousekAffineOwen` in MATLAB. A Sobol sequence has infinitely many runs and we only the first n_j runs as D_j. In addition, the n_j are chosen as powers of two such that each D_j is a (t, m, s)-net in base 2 (Owen, 1995).

Define

M1: $f(\boldsymbol{x}) = (x_1 + x_2 + x_3 + x_4)^2$,

M2: $f(\boldsymbol{x}) = \log(x_1^{-1/2} + x_2^{-1/2})$,

M3: $f(\boldsymbol{x}) = \log(x_1 x_2 x_3 x_4 x_5)$,

M4: $f(\boldsymbol{x}) = (30 + x_1 \sin x_1)(4 + e^{-x_2^2})$,

M5: $f(\boldsymbol{x}) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10, \quad -5 \le x_1 \le 10,$ $0 \le x_2 \le 15$,

M6: $f(\boldsymbol{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2, \quad -10 \le x_1, x_2 \le 10$,

M7: $f(\boldsymbol{x}) = \sin^2(\pi y_1) + (y_1 - 1)^2(1 + 10\sin^2(\pi y_2)) + (y_2 - 1)^2(1 + 10\sin^2(2\pi y_2))$, $y_i = 1 + \frac{x_i - 1}{4}, \quad -10 \le x_1, x_2 \le 10$,

M8: the borehole function in Section 2,

where M1 and M4 are from Mease and Bingham (2006) and M2 and M3 are from Drew and Homem-de Mello (2005). M5, M6, M7 are the Branin, Matyas, Levy functions, respectively. The random factors in these functions are independent, and each is rescaled to have the uniform measure on $(0, 1]$. For each f in M1–M8, consider an experiment to estimate its expected output μ in four stages with $s_2 = s_3 = s_4 = 2$. For all eight methods, $\hat{\mu}_1, \ldots, \hat{\mu}_4$ are computed 10,000 times and the same D_1 is used for the IL, LH and SL schemes. Tables 2.2 and 2.1 report root mean squared errors (RMSE) of $\hat{\mu}_1, \ldots, \hat{\mu}_4$ for functions M1$\sim$M8 with $n_1 = 10$ and $n_1 = 16$, respectively. We have the following findings. (i) The seven methods with stratification yield significantly smaller RMSEs than the IID method. (ii) The RMSEs of the LH and SL methods are similar to each other as both achieve maximum one-dimensional stratification. The (one-stage) LH method requires 10, 30, 70 more runs than the

SL method in stages 2, 3, 4, respectively. (iii) The SL method has further reduction from the IL method for functions M2, M3, M4 and M7 which are either monotonic or dominated by main effects as supported by Theorem 2.9. The advantages of SL over IL are highlighted in Tables 2.1 and 2.2. (iv) The correlation control algorithm improves the variance reduction significantly for both ordinary Latin hypercube design (in IL or LH) or sequentially refined Latin hypercube design (in SL) as seen in columns with '-CC'. The improvement from SL to SL-CC is more significant than that from IL to IL-CC, which is intuitively true as the SL-CC method uses the correlation information from the design in the previous stage. (v) The Sobol sequences perform well when the number of runs gets large, but are sometimes inferior to the SL-CC method. Compared with sequentially refined Latin hypercubes, the Sobol method cannot be constructed by sequential argumentation and need to be generated by using sophisticated algebraic methods.Besides, the Sobol method requires the n_j to be powers of two in each stage to assure a good performance, otherwise the performance may heavily degrade as shown in Table 2.2.

2.5 Discussion

We have derived a general theory for sequentially refined Latin hypercube designs. When used for sequential experiments with computer models, such designs not only achieve attractive maximum stratification in one dimension but also can be generated in multiple stages. This property is particularly useful for large-scale computer experiments. Sampling properties in Sections 2.2 are derived for the purpose of

Table 2.1: Comparison of the RMSEs of $\hat{\mu}_1, \ldots, \hat{\mu}_4$ for functions M1∼M8 by eight methods over 10,000 replications, where $n_1 = 16$, $n_2 = 32$, $n_3 = 64$, $n_4 = 128$.

$f(\boldsymbol{x})$	$\hat{\mu}_i$	IID	IL	IL-CC	LH	LH-CC	SL	SL-CC	Sobol
M1	1	0.594	0.111	0.039	0.111	0.039	0.111	0.039	0.053
	2	0.413	0.079	0.027	0.075	0.014	0.076	0.015	0.021
	3	0.291	0.054	0.016	0.052	0.005	0.053	0.006	0.010
	4	0.206	0.037	0.008	0.036	0.002	0.037	0.002	0.003
M2	1	0.109	0.040	0.039	0.040	0.039	0.040	0.039	0.039
	2	0.077	**0**.028	0.027	0.022	0.020	**0**.022	0.021	0.020
	3	0.055	**0**.018	0.017	0.012	0.011	**0**.012	0.011	0.010
	4	0.039	**0**.011	0.010	0.007	0.006	**0**.007	0.006	0.005
M3	1	0.554	0.146	0.143	0.146	0.143	0.146	0.143	0.146
	2	0.393	**0**.104	0.101	0.073	0.072	**0**.073	0.073	0.074
	3	0.279	**0**.064	0.062	0.036	0.036	**0**.036	0.036	0.037
	4	0.197	**0**.037	0.036	0.018	0.018	**0**.018	0.018	0.018
M4	1	1.563	0.095	0.096	0.095	0.096	0.095	0.096	0.096
	2	1.089	**0**.067	0.068	0.035	0.034	**0**.035	0.034	0.033
	3	0.774	**0**.038	0.038	0.014	0.012	**0**.014	0.012	0.012
	4	0.547	**0**.020	0.020	0.006	0.004	**0**.006	0.004	0.004
M5	1	12.785	10.386	5.111	10.386	5.111	10.386	5.111	3.914
	2	9.022	7.291	3.608	7.211	3.362	7.287	3.473	1.540
	3	6.363	5.154	2.482	5.021	2.350	5.128	2.372	0.601
	4	4.563	3.603	1.706	3.554	1.650	3.583	1.668	0.232
M6	1	4.849	4.099	0.779	4.099	0.779	4.099	0.779	1.222
	2	3.403	2.931	0.543	2.868	0.304	2.909	0.400	0.493
	3	2.431	2.037	0.312	1.981	0.130	2.043	0.196	0.186
	4	1.725	1.424	0.169	1.441	0.056	1.430	0.097	0.071
M7	1	5.664	3.164	3.176	3.164	3.176	3.164	3.176	3.225
	2	3.998	**2**.253	2.265	1.664	1.671	**1**.688	1.667	1.720
	3	2.843	**1**.403	1.415	0.981	0.979	**0**.990	0.983	0.826
	4	1.997	**0**.861	0.862	0.651	0.650	**0**.645	0.649	0.361
M8	1	11.416	2.437	0.877	2.437	0.877	2.437	0.877	4.171
	2	7.977	1.718	0.613	1.650	0.389	1.689	0.405	0.704
	3	5.638	1.192	0.365	1.136	0.214	1.176	0.222	0.277
	4	4.010	0.829	0.212	0.798	0.143	0.810	0.144	0.225

Table 2.2: Comparison of the RMSEs of $\hat{\mu}_1, \ldots, \hat{\mu}_4$ for functions M1∼M8 by eight methods over 10,000 replications, where $n_1 = 10$, $n_2 = 20$, $n_3 = 40$, $n_4 = 80$.

$f(\boldsymbol{x})$	$\hat{\mu}_i$	IID	IL	IL-CC	LH	LH-CC	SL	SL-CC	Sobol
M1	1	0.741	0.155	0.080	0.155	0.080	0.155	0.080	0.210
	2	0.520	0.110	0.056	0.097	0.028	0.099	0.030	0.083
	3	0.369	0.073	0.031	0.066	0.010	0.067	0.011	0.039
	4	0.262	0.050	0.017	0.046	0.004	0.047	0.004	0.012
M2	1	0.137	0.063	0.058	0.063	0.058	0.063	0.058	0.071
	2	0.098	**0**.044	0.042	0.034	0.031	**0**.034	0.032	0.039
	3	0.069	**0**.027	0.027	0.018	0.017	**0**.019	0.017	0.021
	4	0.049	**0**.016	0.016	0.010	0.009	**0**.010	0.009	0.011
M3	1	0.711	0.232	0.233	0.232	0.233	0.232	0.233	0.305
	2	0.505	**0**.165	0.165	0.116	0.116	**0**.115	0.117	0.154
	3	0.355	**0**.102	0.100	0.058	0.058	**0**.058	0.057	0.078
	4	0.249	**0**.058	0.058	0.029	0.029	**0**.029	0.029	0.038
M4	1	1.981	0.194	0.191	0.194	0.191	0.194	0.191	0.466
	2	1.403	**0**.136	0.136	0.070	0.067	**0**.069	0.068	0.163
	3	0.979	**0**.076	0.077	0.026	0.024	**0**.025	0.024	0.058
	4	0.689	**0**.040	0.040	0.010	0.009	**0**.010	0.009	0.020
M5	1	16.101	13.735	7.188	13.735	7.188	13.735	7.188	9.946
	2	11.493	9.651	5.090	9.277	4.362	9.549	4.620	4.962
	3	8.097	6.725	3.371	6.374	2.993	6.669	3.070	2.133
	4	5.744	4.614	2.234	4.475	2.067	4.580	2.123	0.870
M6	1	6.199	5.321	1.493	5.321	1.493	5.321	1.493	3.832
	2	4.356	3.783	1.062	3.693	0.569	3.738	0.683	1.680
	3	3.104	2.652	0.603	2.551	0.232	2.585	0.314	0.690
	4	2.145	1.830	0.324	1.797	0.098	1.822	0.157	0.274
M7	1	7.207	4.337	4.374	4.337	4.374	4.337	4.374	4.918
	2	5.094	**3**.070	3.095	2.981	3.006	**2**.957	2.995	2.821
	3	3.578	**2**.117	2.129	1.369	1.385	**1**.391	1.386	1.540
	4	2.534	**1**.253	1.266	0.850	0.845	**0**.862	0.853	0.836
M8	1	14.668	3.354	1.656	3.354	1.656	3.354	1.656	5.612
	2	10.252	2.375	1.181	2.140	0.662	2.176	0.685	2.862
	3	7.207	1.586	0.674	1.446	0.316	1.494	0.332	1.079
	4	5.091	1.085	0.373	1.019	0.186	1.043	0.194	0.867

numerical integration. The sample size gets doubled at each stage in sequentially refining Latin hypercube designs. For more restrictive sample size, one can enlarge an ordinary Latin hypercube design into an orthogonal array based Latin hypercube design (Tang, 1993; He and Qian, 2011).

Sequentially refined Latin hypercube designs have also been used for the purpose of prediction (Wang, 2003; Donnelly, 2010), while alternative sequential designs proposed by Jones et al. (1998); Wang (2003); Rennen et al. (2010); Donnelly (2010); Leoppky et al. (2010); Loeppky et al. (2011) are available. For a better performance in prediction, one may optimize sequentially refined Latin hypercube designs according to the maximin criterion (Morris and Mitchell, 1995a). This problem, however, poses significant challenges because it involves multi-objective optimization and needs to possess the Latin hypercube structure after each augmentation. Some progress in this direction is made in Rennen et al. (2010) for problems in low dimensions.

Chapter 3

Latin Hypercube Designs with Controlled Correlations and Multi-dimensional Stratification

3.1 Construction of U Designs with Controlled Correlations

Various methods have been proposed to construct Latin hypercube designs with small correlations, while orthogonal arrays have been used to construct Latin hypercube designs with multi-dimensional stratification (see § 1.3 and § 1.4). To integrate these two ideas, we propose a method to construct a new type of orthogonal array-based Latin hypercube design with controlled column-wise correlations, called correlation-controlled U designs. This work has now been published in *Biometrika* (Chen and

Qian, 2014). An $n \times m$ correlation-controlled U design E is generated as follows.

Algorithm 3.1 Generating a Correlation-controlled U Design

Step 1. Randomize the rows, columns and symbols of an $\mathrm{OA}(n, m+1, s, t)$ with $n = \lambda s^t$ to get an array $A = (a_{ik})_{n\times(m+1)}$ with columns $a_1, \ldots, a_{m+1}$.

Step 2. Construct an auxiliary array $B = (b_{ik})_{n\times m}$ with columns $b_1, \ldots, b_m$ by setting $b_{ik} = \pi_k\{a_{i(m+1)}\}$, where the $\pi_k = \{\pi_k(1), \ldots, \pi_k(s)\}$ are independent uniform permutations on Z_s.

Step 3. For $e = 1, \ldots, s$, select the rows in A with $a_{ik} = e$, replace all h of b_k in the selected rows by independent random permutations on $Z_{\lambda s^{t-2}} \oplus (h-1)\lambda s^{t-2}$ for $h = 1, \ldots, s$.

Step 4. Construct an array $D = (d_{ik})_{n\times m}$ with columns $d_1, \ldots, d_m$, where $d_{ik} = q(a_{ik} - 1) + b_{ik}$. Obtain the (i,k)th entry of E as

$$X_i^k = (d_{ik} - \eta_{ik})/n, \tag{3.1}$$

where the η_{ik} are $U[0,1)$ random variables mutually independent of d_{ik}.

Proposition 3.1. *The design E thus constructed achieves uniformity up to t dimensions.*

For an $n \times m$ design E constructed as a correlation-controlled U design in (3.1), generate an array $C = (c_{ik})_{n\times m}$ with columns $c_1, \ldots, c_m$, where $c_{ik} = \lceil b_{ik}s/q \rceil$. Note that C is identical to the array B constructed in step 2 and the c_k are selected based on a symbol permutation of a_{m+1}. The orthogonality of two columns in an orthogonal array will remain unchanged after symbol permutations on either column (Hedayat et al., 1999). Thus, for $k, l = 1, \ldots, m$, c_l is orthogonal to a_k in the construction of a correlation-controlled U design. For an $n \times m$ U design E in (1.9), we can generate C in the same way, where c_l is not guaranteed to be orthogonal to a_k for $k, l = 1, \ldots, m$.

The orthogonality between a_k and c_l can reduce column-wise correlations. Decompose X_i^k in (1.9) and (3.1) as

$$\begin{aligned} X_i^k - 1/2 = &\{(a_{ik} - 1/2)/s - 1/2\} + s^{-1}\{(c_{ik} - 1/2)/s - 1/2\} \\ &- s^{-2}\{(qc_{ik}/s - b_{ik} + 1/2)/(q/s) - 1/2\} - n^{-1}(\eta_{ik} - 1/2), \end{aligned} \tag{3.2}$$

where $(a_{ik} - 1/2)/s - 1/2$, $(c_{ik} - 1/2)/s - 1/2$, $(qc_{ik}/s - b_{ik} + 1/2)/(q/s) - 1/2$ and $\eta_{ik} - 1/2$ are all $O_p(1)$.

Define $\bar{a}_{ik} = (a_{ik} - 1/2)/s - 1/2$ and $\bar{c}_{ik} = (c_{ik} - 1/2)/s - 1/2$ for $i = 1, \ldots, n$, $k = 1, \ldots, m$. The order of $var(\rho_{kl})$ mainly depends on $var\,(\sum_{i=1}^n \bar{a}_{ik}\bar{a}_{il})$, $var\,(\sum_{i=1}^n \bar{a}_{ik}\bar{c}_{il})$, $var\,(\sum_{i=1}^n \bar{c}_{ik}\bar{c}_{il})$, where if E is a correlation-controlled U design, the first two terms are zero while if E is a U design, only the first term is zero. Proposition 3.2 makes this more rigorous.

Proposition 3.2. *Consider constructions of U and correlation-controlled U designs. Then*

(i) *for E constructed in* (1.9) *based on an $OA(\lambda s^t, m, s, t)$, a_k and a_l are orthogonal for $k, l = 1, \ldots, m$, $k \neq l$, and $var\,(\sum_{i=1}^n \bar{a}_{ik}\bar{a}_{il}) = 0$;*

(ii) *for E constructed in* (3.1) *based on an $OA(\lambda s^t, m+1, s, t)$, a_k and a_l, a_k and c_l are orthogonal for $k, l = 1, \ldots, m$, $k \neq l$, and $var\,(\sum_{i=1}^n \bar{a}_{ik}\bar{a}_{il}) = var\,(\sum_{i=1}^n \bar{a}_{ik}\bar{c}_{il}) = 0$.*

Proposition 3.2 indicates that correlation-controlled U designs can result in smaller column-wise correlations. The exact orders of the $var(\rho_{kl})$ for U and correlation-

controlled U designs are studied in §3.2.

Example 3.3. *Let $n = 8$, $m = 5$, $s = 2$, $t = 2$ and $\lambda = 2$. In Table 3.1, A is a randomized OA$(8, 5, 2, 2)$, C is equivalent to B in step 2, B and D are constructed in step 3 and 4, respectively. Here, C is constructed based on four independent uniform permutations on Z_2 gives as $\pi_1 = \{2, 1\}$, $\pi_2 = \{2, 1\}$, $\pi_3 = \{1, 2\}$, $\pi_4 = \{1, 2\}$. Figure 3.1 displays the bivariate projections of the final design E. In any one dimension, each of the eight equally-spaced intervals of (0, 1] contains exactly one point and in any two dimensions, each of the four 4×4 square bins contains precisely two points by construction, as $\lambda = 2$. In the concatenation of a_k and c_l for $k, l = 1, \ldots, 4$, each level combination occurs exactly twice.*

3.2 Sampling Properties

We derive sampling properties of correlation-controlled U designs constructed in §3.1 based on an OA$(s^2, m + 1, s, 2)$ and provide some new results for U designs based on an OA$(s^2, m, s, 2)$ where $m \leq s$. The orthogonal arrays involved have the most economical run size with s levels (Tang, 1993) and guarantee that the array C is identical to the array B from step 3 in the construction of correlation-controlled U designs. The message of our results should be carried over to more general cases, but higher strengths or index $\lambda \geq 2$ will make the proof significantly more complicated.

Proposition 3.4. *Let E be a correlation-controlled U design in* (3.1)*, where $d_{ik} = \lceil nX_i^k \rceil$. Assume $y, z \in Z_n$. Then for $k = 1, \ldots, m$, we have that*

(i) *the probability mass function for d_{ik}, $i = 1, \ldots, n$, is $pr(d_{ik} = y) = n^{-1}$;*

Table 3.1: Arrays A, C, B and D in Example 3.3

a_1	a_2	a_3	a_4	a_5
1	2	2	1	2
1	1	1	1	1
2	1	2	2	1
1	1	2	2	2
2	2	2	1	1
2	2	1	2	2
2	1	1	1	2
1	2	1	2	1

c_1	c_2	c_3	c_4
1	1	2	2
2	2	1	1
2	2	1	1
1	1	2	2
2	2	1	1
1	1	2	2
1	1	2	2
2	2	1	1

b_1	b_2	b_3	b_4
1	2	3	4
3	4	1	2
4	3	2	2
2	1	4	3
3	4	1	1
1	1	3	4
2	2	4	3
4	3	2	1

d_1	d_2	d_3	d_4
1	6	7	4
3	4	1	2
8	3	6	6
2	1	8	7
7	8	5	1
5	5	3	8
6	2	4	3
4	7	2	5

(ii) *the joint probability mass function for* d_{ik} *and* d_{jk}, $i, j = 1, \ldots, n, i \neq j$, *is*

$$pr(d_{ik} = y, d_{jk} = z) = \begin{cases} \{n(n-1)\}^{-1}, & y \neq z, \\ 0, & y = z. \end{cases}$$

Partition $(0, 1]^m$ into n^m equally-spaced cubes of size n^{-m}. Because $d_{ik} = s(a_{ik} - 1) + b_{ik}$, Table 3.2 divides these cubes into four groups according to different conditional probabilities for the jth run X_j given the ith run X_i for $i, j = 1, \ldots, n$ and $i \neq j$, where the number of cubes for each group is given in the last column. The

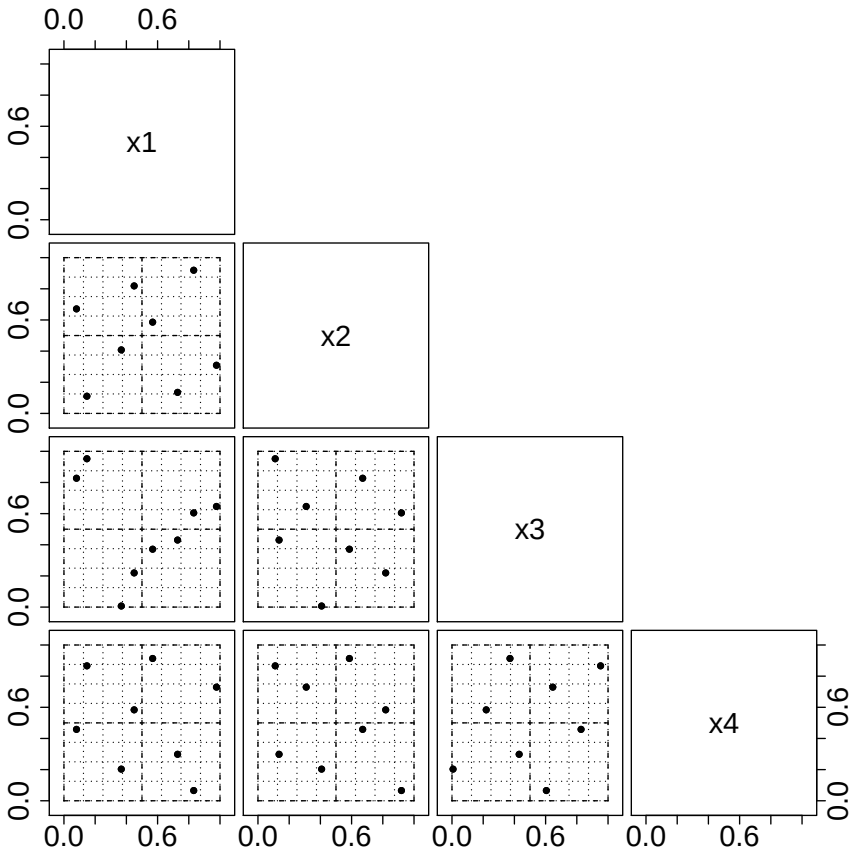

Figure 3.1: Bivariate projections of E in Example 3.3.

table will be used to establish Proposition 3.5.

Proposition 3.5. *Let H_4^C denote the complementary set of H_4 in Table 3.2. Let w be the uniform measure with total mass 1 on H_4^C, and let $g(x_j \mid d_{ik}, k = 1, \ldots, m)$ be the probability density function of X_j conditional on $\lceil nx_{ik} \rceil = d_{ik}$, $k = 1, \ldots, m$,*

Table 3.2: A scheme for grouping entries in row j given row i of the design E in (3.1)

Group	Definition	Size
H_1	Exists some $k \in \mathcal{D}$, $a_{jk} = a_{ik}$, $b_{jk} \neq b_{ik}$, and for any $l \neq k$, $a_{jl} \neq a_{il}$, $b_{jl} \neq b_{il}$	$m(s-1)^{2m-1}$
H_2	For any $k \in \mathcal{D}$, $a_{jk} \neq a_{ik}$ and $b_{jk} = b_{ik}$	$(s-1)^m$
H_3	For any $k \in \mathcal{D}$, $a_{jk} \neq a_{ik}$ and $b_{jk} \neq b_{ik}$	$(s-1)^{2m}$
H_4	Otherwise	$s^{2m} - (s-1)^m\{(s-1)^m + m(s-1)^{m-1} + 1\}$

with respect to w. Then we have

$$g(x_j \mid d_{ik}, k = 1, \ldots, m) = \begin{cases} \frac{(s-1)^m + m(s-1)^{m-1} + 1}{(s+1)(s-1)^{m-1}}, & (i,j) \in H_1, \\ \frac{(s-1)^m + m(s-1)^{m-1} + 1}{s+1}, & (i,j) \in H_2, \\ \frac{(s-m)\{(s-1)^m + m(s-1)^{m-1} + 1\}}{(s+1)(s-1)^m}, & (i,j) \in H_3, \\ 0, & (i,j) \in H_4. \end{cases}$$

Theorem 3.6 follows from Propositions 3.4 and 3.5.

Theorem 3.6. *Suppose $\hat{\mu}$ in* (1.1) *is based on a correlation-controlled U design in* (3.1). *Then*

(i) *$\hat{\mu}$ is an unbiased estimator for μ;*

(ii) *if f is Lipschitz continuous on $(0,1]^m$, then as $n \to \infty$,*

$$var(\hat{\mu}) = n^{-1} var\{f(X)\} - n^{-1} \sum_{k=1}^{m} var\left\{f_{\{k\}}(X^k)\right\} - n^{-1} \sum_{k<l}^{m} var\left\{f_{\{k,l\}}(X^k, X^l)\right\} + o(n^{-1}).$$

Compared with the variance formula (1.10) for a U design, Theorem 3.6 indicates

that a correlation-controlled U design achieves a similar degree of variance reduction from one- and two-dimensional stratification and additional permutations on b_k's do no harm in this regard. Numerical results are provided in the next section to support this observation.

Proposition 3.7. *Let E be a U design in* (1.9)*, where $a_{ik} = \lceil sX_i^k \rceil$, $b_{ik} = \lceil nX_i^k \rceil - s(a_{ik} - 1)$. Assume $y, z \in Z_s$. Then for $i, j = 1, \ldots, n$, $i \neq j$ and $k = 1, \ldots, m$, we have that*

(i) *for any specific $l = 1, \ldots, m$ and $l \neq k$, the joint conditional probability mass function for a_{ik} and a_{jk} given $a_{il} \neq a_{jl}$ is $pr(a_{ik} = y, a_{jk} = z \mid a_{il} \neq a_{jl}) = n^{-1}$;*

(ii) *for any specific $l = 1, \ldots, m$ and $l \neq k$, the joint conditional probability mass function for a_{ik} and a_{jk} given $a_{il} = a_{jl}$ is*

$$pr(a_{ik} = y, a_{jk} = z \mid a_{il} = a_{jl}) = \begin{cases} \{s(s-1)\}^{-1}, & y \neq z, \\ 0, & y = z; \end{cases}$$

(iii) *the joint conditional probability mass function for b_{ik} and b_{jk} given $a_{ik} \neq a_{jk}$ is $pr(b_{ik} = y, b_{jk} = z \mid a_{ik} \neq a_{jk}) = n^{-1}$;*

(iv) *the joint conditional probability mass function for b_{ik} and b_{jk} given $a_{ik} = a_{jk}$ is*

$$pr(b_{ik} = y, b_{jk} = z \mid a_{ik} = a_{jk}) = \begin{cases} \{s(s-1)\}^{-1}, & y \neq z, \\ 0, & y = z. \end{cases}$$

Lemma 3.8 follows from Proposition 3.7.

Lemma 3.8. *Let E be a U design in* (1.9)*, where $a_{ik} = \lceil sX_i^k \rceil$, and $b_{ik} = \lceil nX_i^k \rceil - s(a_{ik} - 1)$. Then for columns $k \neq l$, we have that*

(i) $var\big(\sum_{i=1}^n \bar{a}_{ik}\bar{b}_{il}\big) = s^{-1}(s+1)(n-1)/144$;

(ii) $var\big(\sum_{i=1}^n \bar{b}_{ik}\bar{b}_{il}\big) = n^{-1}(n-1)^2/144$,

where $\bar{a}_{ik} = (a_{ik} - 1/2)/s - 1/2$ and $\bar{b}_{ik} = (b_{ik} - 1/2)/s - 1/2$.

Lemma 3.8(i) shows $var(\sum_{i=1}^n \bar{a}_{ik}\bar{b}_{il})$ in $var(\rho_{kl})$ is not zero for a U design while it is zero for a correlation-controlled U design by Proposition 3.2(ii). This difference leads to Theorem 3.9, which shows that correlation-controlled U designs can reduce $var(\hat{\mu})$ for the bilinear parts to a lower order compared with U designs.

Theorem 3.9. *Suppose that $f(X)$ is a function of bilinear parts in* (1.11) *so that $f(X) = \sum_{k<l}^m \gamma_{kl}(X^k - 1/2)(X^l - 1/2)$ with the γ_{kl} in* (1.12)*. Then for $\hat{\mu}$ in* (1.1)*, as $n \to \infty$,*

(i) $var(\hat{\mu}) = n^{-2}\sum_{k<l}^m \gamma_{kl}^2/72 + O(n^{-5/2})$ *if $\hat{\mu}$ is based on a U design in* (1.9);

(ii) $var(\hat{\mu}) = n^{-5/2}\sum_{k<l}^m \gamma_{kl}^2/144 + O(n^{-3})$ *if $\hat{\mu}$ is based on a correlation-controlled U design in* (3.1)*.*

Corollary 3.10 indicates the correlation-controlled U design results in smaller column-wise correlations.

Corollary 3.10. *For ρ_{kl} defined in* (1.13)*, as $n \to \infty$,*

(i) $var(\rho_{kl}) = n^{-2}/72 + O(n^{-5/2})$ *if ρ_{kl} is based on a U design in* (1.9);

(ii) $var(\rho_{kl}) = n^{-5/2}/144 + O(n^{-3})$ *if* ρ_{kl} *is based on a correlation-controlled* U *design in* (3.1).

3.3 Numerical Illustration

We use examples to compare three types of designs, correlation-controlled Latin hypercube designs by the ranked Gram–Schmidt algorithm (Owen, 1994a), U designs from (1.9) and correlation-controlled U designs from (3.1), for numerical integration. Unless otherwise stated, both U and correlation-controlled U designs are based on the same $\mathrm{OA}(s^2, m+1, s, 2)$, but the former only uses the first m columns of the orthogonal array. For each example, each of the three methods is replicated 100000 times to obtain a table of sample standard deviations of $\hat{\mu}$.

Example 3.11. *Consider* $f(x) = 1000(x_1 - 1/2)(x_2 - 1/2)$*, which only contains purely bilinear bivariate interactions. The difference between* U *and correlation-controlled* U *designs is significant as shown in Table 3.3. In Figure 3.11, the slopes for* U *and correlation-controlled* U *designs are* -2 *and* $-5/2$*, respectively, which agrees with Theorem 3.9. Moreover, the variance reduction of the correlation-controlled* U *design is comparable to that of the correlation-controlled Latin hypercube design.*

Example 3.12. *Consider* $f(x) = 1000(x_1 - 1/2)^2(x_2 - 1/2)^2$*. This function contains bivariate interactions but no bilinear parts. As shown in Table 3.3 and Figure 3.11, the* U *and correlation-controlled* U *designs perform similarly; and both outperform the correlation-controlled Latin hypercube design, which supports Theorem 3.6.*

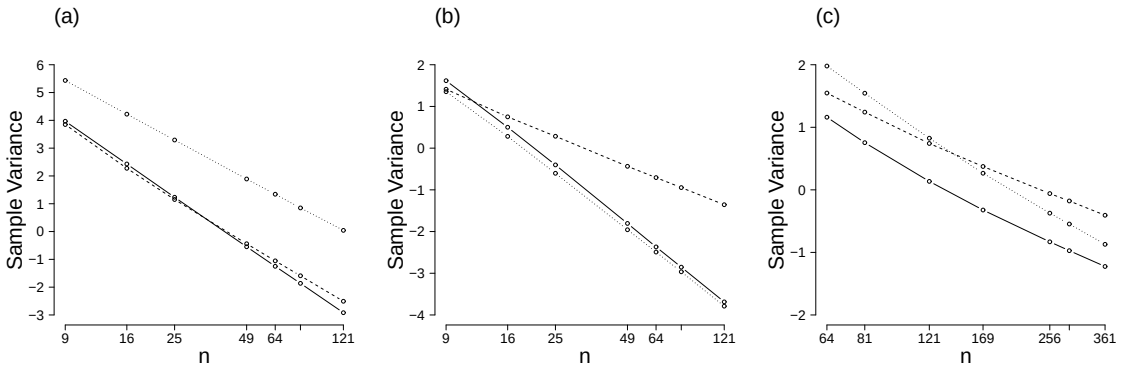

Figure 3.2: Sample variance of $\hat{\mu}$ versus sample size n on a log-log scale for correlation-controlled Latin hypercube designs (dashed), U designs (dotted) and correlation-controlled U designs (solid) in (a) Example 3.11; (b) Example 3.12; (c) Example 3.13.

Example 3.13. *Consider a function proportional to the borehole function (Morris et al., 1993),*

$$f(X) = \frac{20\pi T_u(H_u - H_l)}{\log(r/r_u)\left[1 + 2LT_u/\{\log(r/r_\omega)r_\omega^2 K_\omega\} + T_u/T_l\right]}, \tag{3.3}$$

where the eight input variables, after appropriate scaling, lie in $(0,1]^8$*. This function has significant main effects and bivariate interactions. The sample standard deviations of* $\hat{\mu}$ *for correlation-controlled* U *designs in Table 3.3 are consistently smaller than those of* U *designs, and the difference shrinks when* n *becomes large as shown in Figure 3.11.*

The construction method for correlation-controlled U designs in §3.1 works for any strength $t \geq 3$, but it is complicated to extend Theorems 3.6 and 3.9 to more general orthogonal arrays. Correlation-controlled U designs based on orthogonal arrays with strength $t \geq 3$ are expected to filter out multivariate interactions up to order t and

Table 3.3: Results of the simulation studies. Sample standard deviations of $\hat{\mu}$ are displayed from 100000 replicates per setting

Example	Type of design	Sample standard deviation of $\hat{\mu}$						
		n =9	n =16	n =25	n =49	n =64	n =81	n =121
3.11	CLHD	6.85	3.12	1.78	0.80	0.59	0.45	0.29
	UD	15.14	8.26	5.19	2.57	1.96	1.53	1.02
	CUD	7.27	3.37	1.85	0.76	0.54	0.39	0.23
		n =9	n =16	n =25	n =49	n =64	n =81	n =121
3.12	CLHD	2.03	1.46	1.15	0.80	0.70	0.62	0.51
	UD	1.97	1.15	0.74	0.38	0.29	0.23	0.15
	CUD	2.24	1.28	0.82	0.41	0.31	0.24	0.16
		n =64	n=81	n =121	n =169	n =256	n =289	n =381
3.13	CLHD	2.16	1.86	1.45	1.21	0.97	0.91	0.82
	UD	2.69	2.17	1.51	1.15	0.83	0.76	0.65
	CUD	1.79	1.46	1.07	0.85	0.66	0.62	0.54
		$n = 27$	$n = 125$	$n = 343$	$n = 1331$	$n = 2197$		
3.15	CLHD	46.85	21.57	13.02	6.58	5.10		
	UD	26.57	7.46	3.21	1.04	0.68		
	CUD	9.18	1.65	0.52	0.11	0.06		
		$n = 1331$	$n = 2197$	$n = 4913$	$n = 6859$			
3.16	CLHD	0.42	0.33	0.22	0.19			
	UD	0.33	0.22	0.11	0.08			
	CUD	0.24	0.15	0.08	0.06			

can reduce the variance of the multi-linear terms $\rho_{k_1 k_2 \ldots k_{t'}} = n^{-1} \sum_{i=1}^{n} \prod_{j=1}^{t'} (X_i^{k_j} - 1/2)$ with $t' \leq t$. Below we provide numerical results to illustrate this possibility.

Example 3.14. *Take an* $OA(27, 4, 3, 3)$ *with* $n = 27$, $m = 3$, $s = 3$, $t = 3$ *and* $\lambda = 1$. *Randomize it to obtain* A *in step 1 from §3.1. In step 2, the auxiliary array* B *is constructed using three independent uniform permutations on* Z_2, $\pi_1 = \{1, 2, 3\}$, $\pi_2 = \{3, 1, 2\}$, $\pi_3 = \{2, 3, 1\}$. *Then* B *is modified in step 3 and* D *is constructed in step 4. All relevant arrays are given in Table 3.4.*

Example 3.15. *Consider a trilinear function* $f(x) = 10000(x_1 - 1/2)(x_2 - 1/2)(x_3 - 1/2)$. *Both* U *and correlation-controlled* U *designs are based on the same* $OA(s^3, 4, s, 3)$

Table 3.4: Arrays A, C, B and D in Example 3.14

a_1	a_2	a_3	a_4
2	3	1	1
1	3	1	2
1	3	2	1
2	1	3	1
3	1	3	3
3	3	3	1
3	3	2	2
3	3	1	3
3	1	2	1
2	3	3	2
3	2	3	2
1	1	1	1
2	1	2	2
2	1	1	3
1	3	3	3
3	2	2	3
2	3	2	3
2	2	2	1
1	2	1	3
1	2	2	2
1	2	3	1
1	1	2	3
2	2	3	3
1	1	3	2
2	2	1	2
3	2	1	1
3	1	1	2

c_1	c_2	c_3
1	3	2
2	1	3
1	3	2
1	3	2
3	2	1
1	3	2
2	1	3
3	2	1
1	3	2
2	1	3
2	1	3
1	3	2
2	1	3
3	2	1
3	2	1
3	2	1
3	2	1
1	3	2
3	2	1
2	1	3
1	3	2
3	2	1
3	2	1
2	1	3
2	1	3
1	3	2
2	1	3

b_1	b_2	b_3
2	9	4
5	1	9
3	7	5
3	7	5
8	4	3
1	8	6
5	3	7
7	6	2
2	8	6
5	2	8
4	2	7
2	9	6
4	2	9
7	6	1
7	5	1
9	4	3
8	4	1
1	8	4
9	6	3
4	3	8
1	9	4
8	5	2
9	5	2
6	3	9
6	1	8
3	7	5
6	1	7

d_1	d_2	d_3
11	27	4
5	19	9
3	25	14
12	7	23
26	4	21
19	26	24
23	21	16
25	24	2
20	8	15
14	20	26
22	11	25
2	9	6
13	2	18
16	6	1
7	23	19
27	13	12
17	22	10
10	17	13
9	15	3
4	12	17
1	18	22
8	5	11
18	14	20
6	3	27
15	10	8
21	16	5
24	1	7

but the former only uses the first three columns of the orthogonal array. Notice that $E\{f(X)\}$ *is estimated by* $10000\rho_{123}$*. Table 3.3 indicates that the variance of the trilinear term is substantially reduced by correlation-controlled U designs.*

Example 3.16. *Consider the function in (3.3). Both U and correlation-controlled U designs are based on the same* $OA(s^3, 9, s, 3)$ *but the former only uses the first eight columns of the orthogonal array. Table 3.3 shows that correlation-controlled U designs reduce the variances of* $\hat{\mu}$ *due to bivariate and trivariate interactions as U designs, but also improve the accuracy by controlling the variance components of the bilinear and trilinear terms.*

We conclude that correlation-controlled U designs do no harm relative to U designs for estimating the mean output with respect to a uniform distribution, but can do significantly better for outputs with substantial variance explained by bilinear, trilinear, etc. terms. When orthogonal arrays are not available for desired run sizes, one can use nearly-orthogonal arrays (Wang and Wu, 1992; Nguyen, 1996) to construct correlation-controlled U designs by slightly modifying the procedure in §3.1. Instead of using the last column of array A to generate array B in step 2, we randomly select a column that is orthogonal to all other columns as a_{m+1}. The constructed design will achieve one-dimensional stratification and Proposition 3.2(ii) holds for columns that are orthogonal to each other in the underlying nearly orthogonal array.

We also compared the proposed designs with scrambled Sobol sequences (Niederreiter, 1992; Owen, 1995) and sparse grids (Gerstner and Griebel, 1998) to estimate the expected output μ of the function in (3.3). These scrambled Sobol sequences are generated by MATLAB functions `sobolset` and `scramble` with the op-

tion `MatousekAffineOwen` and sparse grids are generated by the MATLAB function `nwspgr` from `http://www.sparse-grids.de` with the option `GQU`. For the scrambled Sobol sequences with $n = 256$ runs, the estimated standard error of $\hat{\mu}$ is 1.27, which is larger than 0.66 under the proposed method with the same run size. The error under sparse grids with $n = 145$ runs is only 0.03, which is lower than 0.85, the estimated standard error under correlation-controlled U designs with $n = 169$ runs. Sparse grids are based on sophisticated polynomial function approximations and the function in this example may be well approximated by a combination of polynomials. But they can perform poorly for functions of higher dimensions. Consider the function

$$f(x) = 10000 \exp\{-\sum_{k=1}^{m} h_k^2 (x_i - 1/2)^2\},$$

where $h_k = 10/2^{2+k}$. When dimension $m = 15$, the error under sparse grids with $n = 481$ runs is 5.43 while the estimated standard error under controlled correlation U designs with $n = 381$ runs is only 0.39.

A sliced Latin hypercube designs is recently developed for collective evaluations of computer models and ensembles of multiple computer models. Such a special Latin hypercube design can be partitioned into slices of smaller Latin hypercube designs. In this article, we propose an algorithm to select a sliced Latin hypercube design such that the column-wise correlations are controlled for the entire design and each slice of smaller design simultaneously. The performance of the algorithm is investigated.

Chapter 4

Validating Sample Average Approximation Solutions with Negatively Dependent Batches

4.1 Background

Stochastic programming provides a means for formulating and solving optimization problems that contain uncertainty in the model. When the number of possible scenarios for the uncertainty is extremely large or infinite, sample-average approximation (SAA) provides a means for finding approximate solutions for a reasonable expenditure of computational effort. In SAA, a finite number of scenarios is sampled randomly from the full set of possible scenarios, and an approximation to the full problem of reasonable size is formulated from the sampled scenarios and solved us-

ing standard algorithms for deterministic optimization (such as linear programming solvers). Solutions obtained from SAA procedures are typically suboptimal. Substantial research has been done in assessing the quality of an obtained solution (or a set of candidate solutions), and in understanding how different sampling methods affect the quality of the approximate solution.

Important information about a stochastic optimization problem is provided by the *optimality gap* (Mak et al., 1999; Bayraksan and Morton, 2006), which provides a bound on the difference between the objective value for the computed SAA solution and the true optimal objective value. An estimate of the optimality gap can be computed using upper and lower bounds on the true optimal objective value. Mak et al. (1999) proves that the expected objective value of an SAA problem is a lower bound of the objective of the true solution, and that this expected value approaches the true optimal objective value as the number of scenarios increases. We can estimate this lower bound (together with confidence intervals) by solving multiple SAA problems, a task that can be implemented in parallel in an obvious way. An upper bound can be computed by taking a candidate solution x and evaluating the objective by sampling from the scenario set, typically taking a larger number of samples than were used to set up the SAA optimization problem for computing x.

Much work has been done to understand the quality of SAA solutions obtained from Monte Carlo (MC) sampling, Latin hypercube (LH) sampling (McKay et al., 1979), and other methods. MC generates independent identically distributed scenarios where the value of each variable is picked independently from its corresponding distribution. The simplicity of this method has made it an important practical tool;

it has been the subject of much theoretical and empirical research. Many results about the asymptotic behavior of optimal solutions and values of MC have been obtained; see (Shapiro et al., 2009, Chapter 5) for a review. By contrast with MC, LH stratifies each dimension of the sample space in such a way that each strata has the same probability, then samples the scenarios so that each strata is represented in the scenario sample set. This procedure introduces a dependence between the different scenarios of an individual SAA problem. The sample space is in some sense "uniformly" covered on a per-variable basis, thus ensuring that there are no large unsampled regions and leading to improved performance. Linderoth et al. (2006) provides empirical results showing that the bias and variance of a lower bound obtained by solving multiple SAA problems constructed with LH sampling is considerably smaller than the statistics obtained from an MC-based procedure. Theoretical results about the asymptotic behavior of these estimates were provided later by Homem-de Mello (2008). Other results on the performance of LH have been obtained, including results on large deviations (Drew and Homem-de Mello, 2005), rate of convergence to optimal values (Homem-de Mello, 2008), and bias reduction of approximate optimal values (Freimer et al., 2012), all of which demonstrate the superiority of LH over MC. This favorable empirical and theoretical evidence makes LH the current state-of-the-art method for obtaining high-quality lower bounds and optimality gaps via SAA. In this paper, we build on the ideas behind the LH method to obtain LH variants with even better variance properties.

In the past, when solving a set of SAA problems to obtain a lower-bound estimate of the true optimal objective value, each batch of scenarios determining each

SAA was chosen independently of the other batches. In this paper, we introduce two approaches to sampling — *sliced Latin hypercube (SLH)* sampling and *sliced orthogonal-array Latin hypercube (SOLH)* sampling — that yield better estimates of the lower bound by imposing negative dependencies between the batches in the different SAA approximations. These approaches not only stratify *within* each batch (as in LH) but also *between all batches*. The SLH approach is easy to implement, while the SOLH approach provides better variance reduction. With these methods, we can significantly reduce the variance of the lower bound estimator even if the size of each SAA problem or the number of SAA problems were kept the same, which can be especially useful when solving each SAA problem is time consuming or when computing resources are limited. We will provide theoretical results analyzing the variance reduction properties of both approaches, and present empirical results demonstrating their efficacy across a variety of stochastic programs studied in the literature.

4.2 Preliminaries

We consider a stochastic program of the form

$$\min_{x \in X} \; g(x) := \mathbb{E}[G(x, \xi)], \tag{4.1}$$

where $X \subset \mathbb{R}^p$ is a compact feasible set, $x \in X$ is a vector of decision variables, $\xi = (\xi^1, \xi^2, \ldots, \xi^m)$ is a vector of random variables, and $G : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}$ is a real-valued measurable function. Unless stated otherwise, we assume that ξ is a

random vector with uniform distribution on $(0,1]^m$ and that $\mathbb{E}$ is the expectation with respect to the distribution of ξ. If ξ has a different distribution on $\mathbb{R}^m$, we can transform it into a uniform random vector on $(0,1]^m$ as long as $\xi^1, \xi^2, \ldots, \xi^m$ are either (i) independent discrete or continuous random variables or (ii) dependent random variables which are absolutely continuous (Rosenblatt, 1953).

Problem (4.1) may be extremely challenging to solve directly, since the evaluation of $g(x)$ involves a high-dimensional integration. We can replace (4.1) with the following approximation:

$$\min_{x \in X} \hat{g}_n(x) := \frac{1}{n} \sum_{i=1}^{n} G(x, \xi_i), \tag{4.2}$$

where $\xi_1, \xi_2, \ldots, \xi_n$ are scenarios sampled from the uniform distribution on $(0,1]^m$. The function $\hat{g}_n$ is called a *sample-average approximation (SAA)* to the objective g in (4.1). In this paper we will frequently use the term *SAA problem* to refer to equation (4.2). We use x^* and v^* to denote a solution and the optimal value of the true problem (4.1), respectively, while x_n^* and v_n^* denote the solution and optimal value of the SAA problem (4.2), respectively.

We introduce here some items of terminology that are used throughout the remainder of the paper. Let D denote an $n \times m$ matrix with ξ_i^T in (4.2) as its ith row. Hence, D represents a batch of scenarios that define an SAA problem. We will refer ξ_i to as the ith scenario in D. We use $D(:,k)$ to denote the kth column of D, and ξ_{ik} to denote the (i,k) entry of D, that is, the kth entry in the ith scenario in D.

We can express the dependence of v_n^* in (4.2) on D explicitly by writing this

quantity as $v_n(D)$, where $v_n : (0,1]^{n\times m} \to \mathbb{R}$. Given a distribution over the D matrices where $\xi_1, \xi_2, \ldots, \xi_n$ are each drawn from the uniform $(0,1]^m$ distribution but not necessarily independently, it is well known and easy to show that the expectation with respect to the D matrices $\mathbb{E}[v_n(D)] \leq v^*$ giving us a lower bound of the true optimal value (Norkin et al., 1998; Mak et al., 1999). $\mathbb{E}[v_n(D)]$ can be estimated as follows. Generate t independent batches $D_1, D_2, \ldots, D_t$ and compute the optimal value min $v_n(D_r)$ by solving *subproblem* (4.2) for each D_r, $r = 1, 2, \ldots, t$. From Mak et al. (1999), a lower bound estimate of v^* is

$$L_{n,t} := \frac{v_n(D_1) + v_2(D_2) + \cdots + v_n(D_t)}{t}. \tag{4.3}$$

Latin hypercube sampling, which stratifies sample points along each dimension (McKay et al., 1979), has been used in numerical integration for many years. It has been shown that the variance of the mean output of a computer experiment under Latin hypercube sampling can be much lower than for experiments based on Monte Carlo methods (McKay et al., 1979; Stein, 1987; Loh, 1996). Let $v_n^{MC}(D)$ and $v_n^{LH}(D)$ denote the approximate optimal value when the ξ_i in D are generated using Monte Carlo and Latin hypercube sampling, respectively. Homem-de Mello (2008) showed that the asymptotic variance of $v_n^{LH}(D)$ is smaller than the variance of $v_n^{MC}(D)$ under some fairly general conditions. Extensive numerical simulations have provided empirical demonstrations of the superiority of Latin hypercube sampling (Homem-de Mello, 2008; Linderoth et al., 2006).

Following (1.3) with different notation, an $n\times m$ matrix D with scenarios $\xi_1, \xi_2, \ldots, \xi_n$

that defines an SAA problem is obtained as follows (McKay et al., 1979):

$$\xi_{ik} = \frac{a_{ik} - \gamma_{ik}}{n}, \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, m, \tag{4.4}$$

where all γ_{ik} are $U[0, 1)$ random variables, $A = (a_{ik})_{n \times m}$ is an ordinary Latin hypercube, and the quantities a_{ik} and the γ_{ik} are mutually independent. We refer the matrix D thus constructed as an *ordinary Latin hypercube design* defined in Definition 1.1.

We now introduce a different way of looking at design matrices $D \in (0, 1]^{n \times m}$ that will be useful when we discuss extensions to sliced Latin hypercube designs in later sections. We can write a design matrix D as

$$D = (B - \Theta)/n, \tag{4.5}$$

where

$$B = (b_{ik})_{n \times m}, \quad \text{with } b_{ik} = \lceil n\xi_{ik} \rceil,$$
$$\Theta = (\theta_{ik})_{n \times m}, \quad \text{with } \theta_{ik} = b_{ik} - n\xi_{ik}.$$

When D is an ordinary Latin hypercube design, B is an ordinary Latin hypercube and θ_{ik} corresponds to γ_{ik} in (4.4) for all $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$. By the properties of an ordinary Latin hypercube design, the entries in Θ are mutually independent, and Θ and B are independent. We refer B to as the *underlying array* of D.

The lower bound on v^* can be estimated from (4.3) by taking t *independently generated* ordinary Latin hypercube designs $D_1, D_2, \ldots, D_t$ (Linderoth et al., 2006). We denote this sampling scheme by ILH and denote the estimator obtained from (4.3) by $L_{n,t}^{ILH}$.

To illustrate the limitations of the ILH scheme, Figure 4.1 displays three independent 3×3 ordinary Latin hypercube designs generated under ILH with $n = t = m = 3$. Scenarios from each three-dimensional design are denoted by the same symbol, and are projected onto each of the three bivariate planes. The dashed lines stratify each dimension into three partitions. For any design, each of the these three intervals will contain exactly one scenario in each dimension. This scheme covers the space more "uniformly" than three scenarios that are picked identically and independently from the uniform distribution, as happens in Monte Carlo schemes. However, the combination of points from all three designs does not cover the space particularly well, which gives some cause for concern, since all designs are being used in the lower bound estimation. Specifically, when we combine the three designs together (to give nine scenarios in total), it is usually *not* the case that each of the nine equally spaced intervals of (0,1] contains exactly one scenario in any dimension. This shortcoming provides the intuition behind the sliced Latin hypercube (SLH) design, which we will describe in the subsequent sections.

When the stochastic program (4.1) has a unique solution x^* and some proper conditions are satisfied, one has

$$\frac{v_n^{MC}(D) - v^*}{\sigma_{n,MC}(x^*)} \xrightarrow{d} \text{Normal}(0, 1),$$

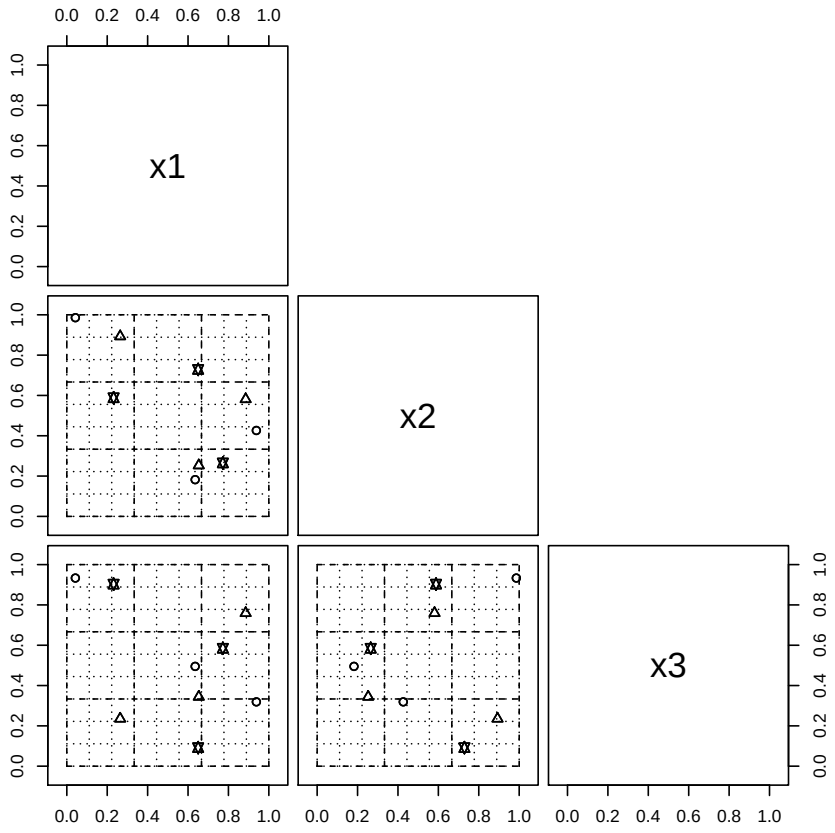

Figure 4.1: Bivariate projections of three independent 3×3 ordinary Latin hypercube designs.

where $\sigma^2_{n,MC} := n^{-1}var[G(x^*, \xi)]$ (Shapiro, 1991). With additional assumptions, Homem-de Mello (2008) shows that

$$\frac{v_n^{LH}(D) - v^*}{\sigma_{n,LH}(x^*)} \xrightarrow{d} \text{Normal}(0, 1), \tag{4.6}$$

where $\sigma^2_{n,LH} := n^{-1}var[G(x^*, \xi)] - n^{-1}\sum_{k=1}^{m} var[G_{\{k\}}(x^*, \xi^k)] + o(n^{-1})$ and $G_{\{k\}}(x^*, \xi^k)$ is the main effect function of $G(x^*, \xi)$ as defined in (1.5).

4.3 Sliced Latin Hypercube Sampling

Instead of generating $D_1, D_2, \ldots, D_t$ independently for each SAA subproblem, we propose a new scheme called *sliced Latin hypercube (SLH) sampling* that generates a family of correlated designs. An SLH design (Qian, 2012) is a $nt \times m$ matrix that can be partitioned into t separate LH designs, represented by the matrices D_r, $r = 1, 2, \ldots, t$, each having the same properties as ILH, with respect to SAA. SLH was originally introduced to aid in the collective evaluation of computer models, but here we demonstrate its utility in creating multiple SAA problems to solve.

Let $L_{n,t}^{SLH}$ denote the lower bound estimator of v^* under SLH. Because the individual designs D_r, $r = 1, 2, \ldots, t$ are LH designs, we have that $\mathbb{E}(L_{n,t}^{SLH}) = \mathbb{E}(L_{n,t}^{ILH})$. Consider two distinct batches of scenarios D_r and D_s for any $r, s = 1, 2, \ldots, t$ and $r \neq s$. We will show that when $v_n(D)$ fulfills a certain monotonicity condition, $v_n(D_r)$ and $v_n(D_s)$ are negatively dependent under SLH. Compared with ILH, SLH introduces *between-batch* negative dependence while keeping the *within-batch* structure intact. As a result, we expect a lower-variance estimator: $var(L_{n,t}^{SLH}) \leq var(L_{n,t}^{ILH})$.

Algorithm 4.1 describes the construction of the matrices D_r, $r = 1, 2, \ldots, t$ for the SLH design. We use notation $D_r(:, k)$ for the kth column of D_r, $\xi_{r,i}$ for the ith scenario of D_r, and $\xi_{r,ik}$ for the kth entry in $\xi_{r,i}$, for $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$, and $r = 1, 2, \ldots, t$.

Algorithm 4.1 Generating a Sliced Latin Hypercube Design

Step 1. Randomly generate t independent ordinary Latin hypercubes $A_r = (a_{r,ik})_{n\times m}$, $r = 1, 2, \ldots, t$. Denote the kth column of A_r by $A_r(:, k)$, for $k = 1, \ldots, m$.

Step 2. For $k = 1, 2, \ldots, m$, do the following independently: Replace all the ℓs in $A_1(:, k), A_2(:, k), \ldots, A_t(:, k)$ by a random permutation on $Z_t \oplus t(\ell - 1)$, for $\ell = 1, 2, \ldots, n$.

Step 3. For $r = 1, 2, \ldots, t$, obtain the (i, k)th entry of D_r as follows:

$$\xi_{r,ik} = \frac{a_{r,ik} - \gamma_{r,ik}}{nt}, \tag{4.7}$$

where the $\gamma_{r,ik}$ are $U[0, 1)$ random variables that are mutually independent of the $a_{r,ik}$.

By vertically stacking the matrices $D_1, D_2, \ldots, D_t$, we obtain the $nt \times m$ matrix representing the SLH design, as defined in Qian (2012).

As in (4.5), we can express each D_r as

$$D_r = (B_r - \Theta_r)/n, \tag{4.8}$$

where $B_r = (b_{r,ik})_{n\times m}$ with $b_{r,ik} = \lceil n\xi_{r,ik} \rceil$ and $\Theta_r = (\theta_{r,ik})_{n\times m}$. We have the following proposition regarding properties of the SLH design, including dependence of the batches. (This result is closely related to (Qian, 2012, Lemma 2).)

Proposition 4.1. *Consider the SLH design with D_r, $r = 1, 2, \ldots, t$ constructed according to Algorithm 4.1, with B_r and Θ_r, $r = 1, 2, \ldots, t$ defined as in* (4.8)*. The following are true.*

(i) B_r, $r = 1, 2, \ldots, t$ *are independent ordinary Latin hypercubes.*

(ii) B_r *and* Θ_r *are independent, for each* $r = 1, 2, \ldots, t$.

(iii) Within each Θ_r, $r = 1, 2, \ldots, t$, *the* $\theta_{r,ik}$ *are mutually independent* $U[0, 1)$ *random variables.*

(iv) For $r, s = 1, 2, \ldots, t$ *with* $r \neq s$, $\theta_{r,ik}$ *is dependent on* $\theta_{s,ik}$ *if and only if* $B_{r,ik} = B_{s,ik}$;

(v) The D_r, $r = 1, 2, \ldots, t$ *are ordinary Latin hypercube designs, but they are not independent.*

Figure 4.2 displays the bivariate projection of the three 3×3 ordinary Latin hypercube designs, each denoted by a different symbol, which are generated under an SLH scheme. For each design, each of the three equally spaced intervals of $(0, 1]$ contains exactly one scenario in each dimension. In contrast to Figure 4.1, when we combine the three designs together, each of the *nine* equally spaced intervals of $(0, 1]$ contains exactly one scenario in any one dimension.

4.4 Theoretical Results under SLH

We derive theoretical results to show $var(L_{n,t}^{SLH}) \leq var(L_{n,t}^{ILH})$ under a monotonicity condition that will be defined shortly.

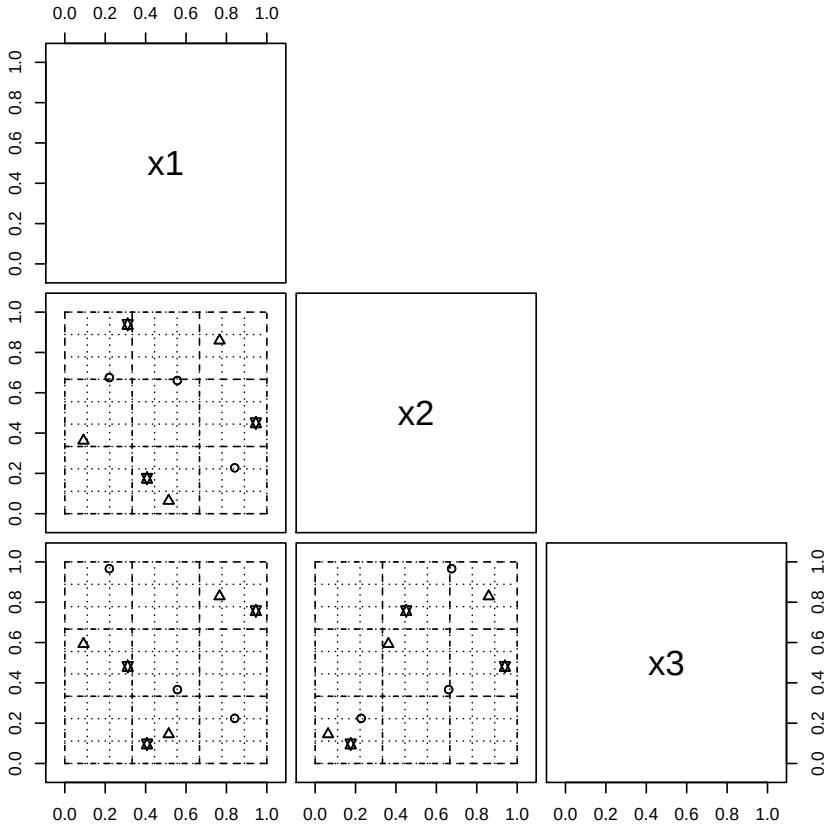

Figure 4.2: Bivariate projections of a sliced Latin hypercube design that consists of three 3×3 ordinary Latin hypercube designs, each denoted by a different symbol.

Definition 4.2. *We say that two random variables Y and Z are* negatively quadrant dependent *if*

$$P(Y \leq y, Z \leq z) \leq P(Y \leq y)P(Z \leq z), \quad \textit{for all } y, z.$$

Definition 4.3. *Let $B = (b_{ik})$ denote the underlying ordinary Latin hypercube of D*

in (4.5) *such that* $D = (B - \Theta)/n$. *Let* $v_n(D) = H(\Delta; B)$ *given* B, *where* $H(\Delta; B) : (0, 1]^{n \times m} \to \mathbb{R}$ *and* $\Delta = (\delta_{\ell k})$, *with* $\delta_{\ell k} = \theta_{ik}$ *such that* $b_{ik} = \ell$ *for* $\ell = 1, 2, \ldots, n$ *and* $k = 1, 2, \ldots, m$. *The function* $v_n(D)$ *is said to be* monotonic *if the following two conditions hold: (a) for all* B, $H(\Delta; B)$ *is monotonic in each argument of* Δ, *and (b) the direction of the monotonicity in each argument of* Δ *is consistent across all* B.

Example 4.4. *Consider* $D = (\xi_{ik})_{3 \times 2}$. *Let* $v_n(D) = \sum_{i=1}^{3} \sum_{k=1}^{2} (\xi_{ik} - 1/3)^2$. *When*

$$B = \begin{bmatrix} 3 & 1 \\ 2 & 3 \\ 1 & 2 \end{bmatrix},$$

we have $\delta_{11} = \theta_{31}$, $\delta_{21} = \theta_{21}$, $\delta_{31} = \theta_{11}$, $\delta_{12} = \theta_{12}$, $\delta_{22} = \theta_{32}$, *and* $\delta_{32} = \theta_{22}$ *and*

$$H(\Delta; B) = \frac{1}{9} \left[\delta_{11}^2 + (1 - \delta_{21})^2 + (2 - \delta_{31})^2 + \delta_{12}^2 + (1 - \delta_{22})^2 + (2 - \delta_{32})^2 \right],$$

Thus, $H(\Delta; B)$ *is increasing in* δ_{11} *and* δ_{12} *while it is decreasing in the other* δ_{ik}*s, for* $\delta_{ik} \in (0, 1]$. *This is true for any underlying ordinary Latin hypercube* B, *since* δ_{11} *and* δ_{12} *are always associated with values in* D *which are between (0,1/3] in this example. By Definition 4.3,* $v_n(D)$ *is monotonic.*

The monotonicity condition can be checked by directly studying the function $v_n(D)$, but it can also be shown to be satisfied if the stochastic program has certain nice properties. Later we will prove some sufficient conditions for two-stage stochastic

linear programs and give a simple argument to show how some problems from the literature have the monotonicity property.

Qian (2012) proves that the function values of any two scenarios in a sliced Latin hypercube design are negatively quadrant dependent. The next lemma extends this result, showing the function values of any two *batches* in a sliced Latin hypercube design are also negatively quadrant dependent, under the monotonicity assumption on $v_n(D)$ described in Definition 4.3.

Lemma 4.5. *Consider $D_1, D_2, \ldots, D_t$ generated by Algorithm 4.1. If $v_n(D)$ is monotonic, then we have*

$$\mathbb{E}[v_n(D_r)v_n(D_s)] \leq \mathbb{E}[v_n(D_r)]\mathbb{E}[v_n(D_s)],$$

for any $r, s = 1, 2, \ldots, t$ with $r \neq s$.

The next result is an immediate consequence of Lemma 4.5. It indicates that the variance of the lower-bound estimator could be reduced by using SLH, when $v_n(D)$ is monotonic.

Theorem 4.6. *Consider two lower bound estimators $L_{n,t}^{ILH}$ and $L_{n,t}^{SLH}$ in (4.3) obtaind under ILH and SLH, respectively. Suppose $v_n(D)$ in monotonic, then for any n and t, we have that*

$$var(L_{n,t}^{SLH}) \leq var(L_{n,t}^{ILH}).$$

Even if the monotonicity condition does not hold, we can prove similar statements about the asymptotic behavior of the variance of ILH and SLH.

Theorem 4.7 gives an asymptotic result that shows the same conclusion can be drawn as in Theorem 4.6 even if the monotonicity condition does not hold.

Theorem 4.7. *Let D denote an $n \times m$ Latin hypercube design based on a Latin hypercube B such that $D = (B - \Theta)/n$. Let $H(\Delta; B) = v_n(D)$. Suppose $\mathbb{E}\{[v_n(D)]^2\}$ is well defined. As $t \to \infty$ with n fixed, the following are true.*

(i) $var(L_{n,t}^{ILH}) = t^{-1} var[v_n(D)]$.

(ii) $var(L_{n,t}^{SLH}) = t^{-1} var[v_n(D)] - t^{-1} \sum_{\ell=1}^{n} \sum_{k=1}^{m} \int \{\mathbb{E}[H_{\{\ell k\}}(\delta_{\ell k}; B)]\}^2 d\delta_{\ell k} + o(t^{-1})$, *where $H_{\{\ell k\}}(\delta_{\ell k}; B)$ is the main effect function of $H(\Delta; B)$ with respect to $\delta_{\ell k}$.*

(iii) If $v_n(D) = \sum_{k=1}^{m} v_{n,\{k\}}(D(:,k))$ is additive, where $v_{n,\{k\}} : (0,1]^n \to \mathbb{R}$, then $var(L_{n,t}^{SLH}) = o(t^{-1})$.

We now discuss the theoretical properties of SLH for two-stage stochastic linear programs. Consider problems of the form

$$\min_{x \in X} c^T x + \mathbb{E}[Q(x, \xi)], \tag{4.9}$$

where X is a polyhedron and

$$Q(x, \xi) = \inf \left\{ q^T y \,:\, Wy \leq h - Tx,\ y \geq 0 \right\}, \tag{4.10}$$

and $\xi := (h, q, T)$. The problem has *fixed recourse* since the recourse matrix W does not depend on the random variable ξ. Defining $G(x, \xi)$ to be the function $c^T x + Q(x, \xi)$, we see that (4.9) is a special case of (4.1). Let $x = (x^1, x^2, \ldots, x^p)$

and $T = (T_{kj})$. By (4.10), $Q(x, \xi)$ is a decreasing function of any entry in h, for any $x \in X$. Furthermore, $Q(x, \xi)$ is a decreasing function of any entry T_{kj} in T if x^j is nonpositive, and an increasing function of any entry T_{kj} in T if x^j is nonnegative.

By LP duality, we have

$$Q(x, \xi) = \sup \left\{ u^T(h - Tx) \,:\, W^T u \leq q,\ u \leq 0 \right\},$$

and hence $Q(x, \xi)$ is an increasing function of any entry in q for any $x \in X$. We conclude that $G(\cdot, \xi)$ is monotonic in each component of ξ if the recourse matrix W is fixed.

Lemma 4.8. *Let $v_n(D)$ in (4.2) denote the approximated optimal value of the two-stage stochastic program in (4.9) with fixed recourse. Then $v_n(D)$ is monotonic by Definition 4.3 if* (i) *T is fixed or* (ii) *for every $j = 1, 2, \ldots, p$, x^j is always nonnegative or nonpositive, given any D.*

Example 4.9. *Consider the newsvendor problem from Freimer et al. (2012), which can be expressed as a two-stage stochastic program. In the first stage, choose an order quantity x. After demand ξ has been realized, we decide how much of the available stock y to sell. Assume that demand is uniformly distributed on the interval $(0, 1]$, and there is a shortage cost $\alpha \in (0, 1)$ and an overage cost $1 - \alpha$. The second stage problem is*

$$P: \qquad Q(x, \xi) = \min_y \left[(1 - \alpha)(x - y) + \alpha(\xi - y) \,|\, y \leq x,\ y \leq \xi \right].$$

Since the first-stage cost is zero, the two-stage stochastic program is

$$MP: \qquad \min_x \mathbb{E}\left\{\min_y \left[(1-\alpha)(x-y)+\alpha(\xi-y)\,|\,y\le x,\ y\le \xi\right]\right\}.$$

The optimal value is $v^*=\alpha(1-\alpha)/2$ *with solution* $x^*=\alpha$.

Based on a sample of n *demands* $\xi_1,\xi_2,\ldots,\xi_n$, *the approximated optimal value is given by*

$$v_n(D)=\min_x \frac{1}{n}\sum_{i=1}^{n}\left[(1-\alpha)(x-\xi_i)^+ +\alpha(\xi_i-x)^+\right],$$

where $D=(\xi_1,\xi_2,\ldots,\xi_n)^T$. *The optimal solution* x_n^* *is the* $\lceil \alpha n\rceil$*th order statistic of* $\{\xi_1,\xi_2,\ldots,\xi_n\}$.

Table 4.1: Means and standard errors (in parentheses) with 1000 replicates

n	Scheme	$t=5$	$t=10$	$t=20$
2	ILH	0.1003 (1.83E-2)	0.1002 (1.31E-2)	0.1000 (9.23E-3)
	SLH	0.0999 (3.71E-3)	0.1000 (1.31E-3)	0.1000 (4.49E-4)
20	ILH	0.1201 (6.99E-4)	0.1200 (5.01E-4)	0.1200 (3.70E-4)
	SLH	0.1200 (1.43E-4)	0.1200 (4.87E-5)	0.1200 (1.72E-5)
200	ILH	0.1200 (2.18E-5)	0.1200 (1.60E-5)	0.1200 (1.14E-5)
	SLH	0.1200 (4.40E-6)	0.1200 (1.59E-6)	0.1200 (5.60E-7)

Let $\alpha=.4$, *for which* $v^*=0.12$. *Table 4.1 gives means and standard errors for the estimators of* $L_{n,t}=\mathbb{E}[v_n(D)]$ *for several values of* n *and* t. *This table shows that SLH reduces the variance of* $L_{n,t}$ *significantly when compared with ILH. Analytically, we have*

$$v_n(D)=H(\Delta;B)=n^{-2}\sum_{i=1}^{r^*-1}(1-\alpha)(r^*-i+\delta_{r^*}-\delta_i)+n^{-2}\sum_{i=r^*+1}^{n}\alpha(i-r^*+\delta_i-\delta_{r^*}),$$

where $r^* = \lceil \alpha n \rceil$ and B is an arbitrary underlying Latin hypercube for D. We notice that for any B, $H(\Delta; B)$ is decreasing in $\delta_1, \ldots, \delta_{r^*-1}$, increasing in $\delta_{r^*+1}, \ldots, \delta_n$ and monotonic in δ_{r^*} (the direction depends on α). Thus, $v_n(D)$ is monotonic, which can alternatively be checked by applying Lemma 4.8. By Theorem 4.6, we should have $var(L_{n,t}^{SLH}) \leq var(L_{n,t}^{ILH})$.

We notice that $var(L_{n,t}^{ILH}) = O(t^{-1})$ while $var(L_{n,t}^{SLH}) = o(t^{-1})$, a fact that can be explained by Theorem 4.7 (iii), since the newsvendor problem only has one random variable and $v_n(D)$ is additive.

Theorems 4.6 and 4.7 confirm the effectiveness of sliced Latin hypercube designs in reducing the variance of lower-bound estimates in SAA. However, the assumptions in those theorems limit their applicability to fairly specialized problems. Theorem 4.6 does not apply to two-stage problem in (4.9) with random recourse. Theorem 4.7 does not apply when $n \to \infty$, which is a more practical assumption than $t \to \infty$. In this section, we consider the case in which $\xi^1, \xi^2, \ldots, \xi^m$ are discrete random variables, as discussed by Homem-de Mello (2008). In fact, we assume $\xi^1, \xi^2, \ldots, \xi^m$ to be *independent* discrete random variables. We plan to show that estimating the lower bound of v^* is almost equivalent to a numerical integration problem. Several existing results regarding numerical integration provide us with tools for studying effects of different sampling schemes for lower bound estimation more generally.

Assumption 1. *For each $x \in X$, $\hat{g}_n(x) \to g(x)$ with probability one (denoted "w.p. 1").*

Assumption 2. *The feasible set X is compact and polyhedral; the function $G(\cdot, \xi)$ is convex piecewise linear for every value of ξ; and the distribution of ξ has finite support.*

Assumption 1 holds under Latin hypercube sampling if $\mathbb{E}\{[G(x,\xi)]^2\} < \infty$ for every $x \in X$, by (Loh, 1996, Theorem 3). Assumption 2 holds in practice for stochastic linear programs in which the random vector is discretized. Let S^* denote the set of optimal solutions of (4.1). Based on both assumptions, Homem-de Mello (2008) shows the consistency of the approximated optimal solution in (4.2).

Proposition 4.10. (Homem-de Mello, 2008, Proposition 2.5). *Suppose that Assumptions 1 and 2 hold. Then $x_n^* \in S^*$ w.p. 1, for n sufficiently large.*

Now let F_k denote the cumulative distribution function of ξ^k for $k = 1, 2, \ldots, m$. Define the inverse of F_k as $F_k^{-1}(z) := \inf\{y \in \Xi_k : F_k(y) \geq z\}$, where Ξ_k is the support of F_k and $z \in 0, 1]$. We can express $\xi = (\xi^1, \xi^2, \ldots, \xi^m)$ as $\Psi(\tilde{\xi}) = (F_1^{-1}(\tilde{\xi}^1), F_2^{-1}(\tilde{\xi}^2), \ldots, F_m^{-1}(\tilde{\xi}^m)))$, where $\tilde{\xi}$ is a random vector uniformly distributed on $(0,1]^m$. Define $\tilde{G}(x, \tilde{\xi})$ on $(0,1]^m$ so that $\tilde{G}(x, \cdot) = G(x, \Psi(\cdot))$. Results obtained in previous sections would still apply with respect to $\tilde{G}$ and $\tilde{\xi}$.

The following proposition connects two-stage stochastic linear program in (4.9) and numerical integration.

Proposition 4.11. *Consider problem* (4.9)*, and suppose that $\mathbb{E}\left\{[\tilde{G}(x,\tilde{\xi})]^2\right\} < \infty$ for every $x \in X$, and that $S^* = \{x^*\}$ is a singleton. Then for n sufficiently large, we have*

$$L_{n,t} = (nt)^{-1} \sum_{r=1}^{t} \sum_{i=1}^{n} \tilde{G}(x^*, \tilde{\xi}_{r,i}) \quad w.p.1,$$

where $\tilde{\xi}_{r,i}$ is the ith scenario in D_r.

Proposition 4.11 indicates that $L_{n,t}$ becomes an integral estimator of $\tilde{G}(x^*, \xi_{r,i})$ and that $L_{n,t}$ is directly estimating the true optimal value v^* of (4.1), for n large enough. Results about the variance formula for $L_{n,t}^{ILH}$ and $L_{n,t}^{SLH}$ are given in the next result.

Theorem 4.12. *Consider problem* (4.9). *With the same conditions in Proposition 4.11, we have that the following results hold as $n, t \to \infty$:*

(i) $var(L_{n,t}^{ILH}) = (nt)^{-1} var\left[\tilde{G}(x^*, \tilde{\xi})\right] - (nt)^{-1} \sum_{k=1}^{m} var\left[\tilde{G}_{\{k\}}(x^*, \tilde{\xi}^k)\right] + o(n^{-1}t^{-1})$;

(ii) $var(L_{n,t}^{SLH}) = (nt)^{-1} var\left[\tilde{G}(x^*, \tilde{\xi})\right] - (nt)^{-1} \sum_{k=1}^{m} var\left[\tilde{G}_{\{k\}}(x^*, \tilde{\xi}^k)\right] + o(n^{-1}t^{-1})$;

(iii) *If $\tilde{G}(x^*, \tilde{\xi}) = \sum_{k=1}^{m} \tilde{G}_{\{k\}}(x^*, \tilde{\xi}^k)$ is additive, then $var(L_{n,t}^{SLH}) \leq var(L_{n,t}^{ILH})$. Furthermore, we have*

$$var(L_{n,t}^{ILH}) = O(n^{-2}t^{-1}) \text{ and } var(L_{n,t}^{SLH}) = O(n^{-2}t^{-2}).$$

Let ν_k denote the number of all possible distinct values for ξ^k in Ξ_k, and denote by $p_{(i)k}$ the probability of ξ taking its ith smallest possible value. To sample ξ^k, we will first take $\tilde{\xi}^1, \tilde{\xi}^2, \ldots, \tilde{\xi}^m$ in n equally spaced subintervals in $(0, 1]$, respectively. If any $\tilde{\xi}^k$ taken from the same subinterval always leads to the same value for $\xi^k = F_k^{-1}(\tilde{\xi}^k)$, then $v_n(D)$ will entirely depend on the underlying ordinary Latin hypercube B. The following proposition gives sufficient conditions for equal performance between ILH and SLH.

Proposition 4.13. *If $np_{(i)k}$ is an integer for all $i = 1, 2, \ldots, \nu_k$ and $k = 1, 2, \ldots, m$, then $var(L_{n,t}^{SLH}) = var(L_{n,t}^{ILH})$.*

Theorem 4.12 (i) and (ii) indicate that ILH and SLH are equally effective, in general, for estimating v^* when $n, t \to \infty$. Proposition 4.13 gives a specific case in which SLH is exactly the same as ILH. Another type of sliced Latin hypercube design is introduced in § 4.5, which possesses similar (if not the same) within-batch structure as ILH and SLH, but much stronger between-batch negative dependence than SLH to further reduce the variance $var(L_{n,t})$ of the lower-bound estimator.

4.5 Sliced Orthogonal Array Based Latin Hypercube Sampling

Section 4.4 indicates that SLH may yield significant improvement in estimating $L_{n,t}$ when the monotonicity condition in Definition 4.3 is satisfied or when $\tilde{G}(x^*, \tilde{\xi}^k)$ is an additive function in $\tilde{\xi}^k$, under Assumptions 1 and 2. Both the monotonicity and additivity conditions emphasize *individual* random variables. As a consequence, the results in Theorems 4.6 and 4.12 are intuitive, because combining all batches under SLH (rather than ILH) gives a design with better stratification in each dimension; see Figures 4.1 and 4.2. On the other hand, the combined design under SLH does not possess better stratification when we consider *groups* of variables. Theorem 4.12 (i) and (ii) suggest that we would need a better sampling scheme than SLH if neither the monotonicity nor additivity conditions are satisfied.

We now introduce another sampling scheme with the following properties. First, the design for each SAA subproblem in this scheme is an ordinary Latin hypercube, as for ILH and SLH. Fixing this property between our different sampling schemes allows us to better study and understand the benefits of improving the between-batch stratification. Second, when we choose the number of batches t sufficiently large, the increased size of the combined design matrix achieves better stratification not just for each variable but also for every pair of variables. With the additional stratification, the $var(L_{n,t})$ can be further reduced, under the assumptions in Proposition 4.11.

As illustrated in § 1.3, such multi-dimensional stratification can be achieved by utilizing orthogonal arrays in Definition 1.2. Let SOLH denote the scheme that generates batches $D_1, D_2, \ldots, D_t$ as slices of a Latin hypercube design based on an orthogonal array. The original purpose of SOLH was to share strength across all batches for numerical integration with higher accuracy. Given an $\mathrm{OA}(N, m+1, t, 2)$ with $N = nt$, $n = \lambda t$, and $s = t$ symbols, batches $D_1, D_2, \ldots, D_t$ each with n scenarios can be constructed under SOLH using Algorithm 4.2 (Hwang et al., 2013).

We obtain a sliced orthogonal array based Latin hypercube design by vertically stacking $D_1, D_2, \ldots, D_t$. The construction above exploits the fact that taking the scenarios in an $\mathrm{OA}(N, m+1, s, \tau)$ that have the same level in the first column, and deleting that first column, gives an $\mathrm{OA}(N/s, m, s, \tau)$ (Hedayat et al., 1999).

Figure 4.3 presents bivariate projections of a Latin hypercube design based on sliced orthogonal arrays, with batches D_1, D_2, D_3 based on an $\mathrm{OA}(18, 4, 3, 2)$. For any D_r, each of the six equally spaced intervals of $(0, 1]$ contains exactly one scenario in each dimension. When combined, each of the 18 equally spaced intervals of $(0, 1]$

Algorithm 4.2 Generating a Sliced Orthogonal Array-Based Latin Hypercube Design

Step 1. Randomize the rows, columns and symbols of an OA$(N, m+1, t, 2)$ to obtain an array $C = (c_{ik})_{N\times(m+1)}$. Let $C(:,1), C(:,2), \ldots, C(:,m+1)$ denote $m+1$ columns of C.

Step 2. Rearrange the rows of C so that $c_{i(m+1)} = \ell$ if $\lceil i/n \rceil = \ell$ for $\ell = 1, 2, \ldots, t$. For $r = 1, 2, \ldots, t$, let $A_r = (a_{r,ik})_{n\times m}$ denote a Latin hypercube design such that $a_{r,ik} = c_{[(r-1)n+i]k}$.

Step 3. For $r = 1, 2, \ldots, t$ and $k = 1, 2, \ldots, m$, do the following independently: replace all the ℓs in $A_r(:,k)$ by a uniform permutation on $Z_\lambda \oplus (l-1)t$ for $\ell = 1, 2, \ldots, t$.

Step 4. Use A_r thus obtained to construct D_r following steps 2 and 3 for SLH in Algorithm 4.1.

contains exactly one scenario. Additionally, each of the 3×3 squares in the dashed lines has exactly two scenarios, because $\lambda = 2$.

We provide a general variance formula of $L_{n,t}$ under SOLH introduced in the last section. The result is a direct consequence of (Hwang et al., 2013, Theorem 1).

Theorem 4.14. *Consider problem* (4.9)*, and suppose that the conditions in Proposition 4.11 hold. Based on an OA$(N, m+1, t, 2)$ with $N = nt$, $n = \lambda t$, and $s = t$ symbols, we have as $s \to \infty$ that*

$$var(L_{n,t}^{SOLH}) = N^{-2} \sum_{|u|\geq 3} M(u, |u|) var[\tilde{G}_u(x^*, \tilde{\xi}^u)] + o(N^{-1}),$$

where u is a subset of $\mathcal{D}$, and $\tilde{\xi}^u$ contains $\tilde{\xi}^k$ for $k \in u$.

Theorem 4.14 shows that the variance of $L_{n,t}$ can be reduced under SOLH after

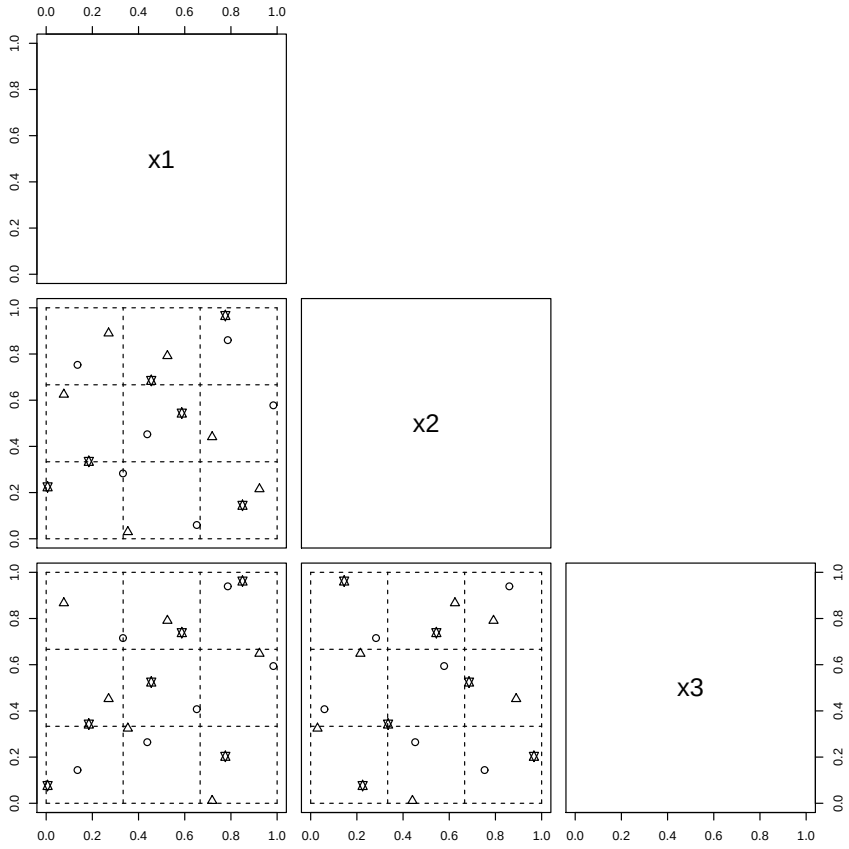

Figure 4.3: Bivariate projections of a sliced orthogonal array based Latin hypercube design $\mathbf{D}$ with batches D_1 (circles), D_2 (triangles), and D_3 (stars).

filtering out the bivariate interactions in $\tilde{G}(x^*, \tilde{\xi})$. This fact remains true for SOLH based on any strength-two orthogonal array even if coincidence defects are present, provided that $s \to \infty$. According to the functional ANOVA decomposition and Theorem 4.12 (i), we have $var(L_{n,t}^{SLH}) = N^{-1} \sum_{|u|\geq 2} var[\tilde{G}_u(x^*, \tilde{\xi}^u)] + o(N^{-1})$, where coefficients of variances due to interactions with more than two variables are all of order N^{-1}. As a result, SOLH based on an orthogonal array with $M(u, |u|) > N$, for

some $u \subseteq \mathcal{D}$ and $|u| \geq 3$, can inflate the variance due to higher-order interactions. This side effect is less significant if the bivariate interactions dominate higher order interactions, which is true in most problems.

In addition, we need to emphasize that each batch under SOLH is based on an OA$(n, m, t, 1)$, which is a Latin hypercube design instead of an ordinary Latin hypercube. We conjecture that Assumption 1 still holds in this case, and the bivariate variance reduction is due to the between-batch negative dependence rather than the within-batch non-ordinary Latin hypercube structure. This conjecture is consistent with our computational observations, as shown below.

Using the SOLH method is not just a matter of picking the desired n, m values of the dimensions of the design matrices and the number of batches t, like it was in the SLH case. Picking the appropriate orthogonal array requires some understanding of the tradeoffs between the different choices.

Consider an OA$(n^2, m+1, n, 2)$. The benefits of using these arrays include (i) each batch is based on an ordinary Latin hypercube, and (ii) there is no coincidence defect. However, we have to solve n batches of n scenarios each. We want n to be large enough to ensure that the quantity x_n^* converges and that the bias $v^* - \mathbb{E}[v_n(D)]$ is small, but solving n batches could be computationally infeasible.

One way around this computational hurdle is to relax the restriction that we use all the batches that are generated by the SOLH method. We can use just the first t batches, and we will call this the sliced partial orthogonal-aray-based Latin hypercube method (SPOLH). Let υ denote the fraction of the number of batches generated by SOLH. SPOLH can perform better than SLH since υ of the variance

due to bivariate interactions can be removed. However, when υ is small (10%, say), it does not guarantee substantial variance reduction from SLH to SPOLH.

We can alternatively use orthogonal arrays with a higher index value (i.e. $\lambda > 1$) allowing us to have a t that is much smaller than n. In this case, there is coincidence defect and picking and generating the right orthogonal arrays with desirable n, m, t values with low $M(u, r)$ can be difficult.

4.6 Experimental Setup

To illustrate the effectiveness of negatively dependent designs for estimation of the lower bound on the objective value in stochastic programs, we perform computational experiments using data sets from the literature. This section provides some details on our implementations and test problems.

We estimate a single lower bound by choosing a design family, obtaining a set of sampled approximations using this design family, and finally solving these approximations. The performance of a sampling scheme is assessed by repeating this process to obtain a number of lower bound estimates. We then calculate the mean and variance of these estimates.

We augmented the SUTIL library (Czyzyk et al., 2000), a C and C++ based library for manipulating stochastic programming instances with a design-based sampling framework. This augmented SUTIL library can take an orthogonal array as input and generate an orthogonal-array-based design family as output. The library can generate other design families from scratch. We used a C library due to Owen

(available at `http://lib.stat.cmu.edu/`) to generate the orthogonal arrays. The SUTIL software produces the extensive form (or deterministic equivalent) linear program of each sampled instance and outputs the linear program as an MPS file. These files are fed into the commercial linear programming solver Cplex v12.5 and solved using the barrier method. All the timed experiments were run on an Intel Xeon X5650 (24 cores @ 2.66Ghz) server with 128GB of RAM, and 16 of the 24 cores available on the machine were used. Other experiments were run on the HTCondor grid (Thain et al., 2005) of the Computer Sciences department at University of Wisconsin-Madison. They required more than a week of wall-clock time to complete.

Our five test problems were drawn from the stochastic programming literature. In fact, they are the same problems that were studied in Linderoth et al. (2006). These problems are specified in SMPS file format (a stochastic-programming extension of MPS), and have finite discrete distributions for their random variables.

20term, first described in Mak et al. (1999), is a model of a motor freight carrier's operations. The first stage variables represent positions of a fleet of vehicles at the beginning of a day. The second-stage variables determine the movements of the fleet through a network to satisfy point-to-point demands for shipments (with penalties for unsatisfied demands), and to end the day with the fleet back in their initial positions.

gbd is derived from an aircraft allocation problem originally described in the textbook of Dantzig (1963, Chapter 28). Four different types of aircraft are to be allocated to routes to maximize the profit under uncertain demand for each

route. There are costs associated with using each aircraft type, and when the capacity does not meet demand.

LandS is a modification by Linderoth et al. (2006) of a simple problem in electrical investment planning described by Louveaux and Smeers (1988). The first-stage variables represent capacities of different new technologies, and the second-stage variables represent production of each of the different modes of electricity from each of the technologies.

ssn (Sen et al., 1994) is a problem from telecommunications network design. The owner of a network sells private-line services between pairs of nodes in the network. During the first stage of the problem, the owner has a budget for adding bandwidth (capacity) to the edges in the network. In the second stage, to satisfy uncertain demands for service between each pair of nodes, short routes between two nodes with sufficient total bandwidth must be identified. Unmet demands incur costs, and the goal of the problem is to minimize the expected costs.

storm is a cargo flight scheduling problem described by Mulvey and Ruszczyński (1993). The goal is to schedule cargo-carrying flights over a set of routes in a network, where the amount of cargo delivered to each node is uncertain. In the first stage, the number of planes of each type on each route is decided. In the second stage, the random variables are the demands on the amounts of cargo to be delivered between nodes. The goal is to minimize (1) the costs that comes from assigning the planes and balancing payloads, and (2) the penalties

associated with unmet demands.

All of these problems fit our monotonicity assumption (Definition 4.3 of § 4.4). The intuition for this property is straightforward – the objective in each problem is to minimize costs and penalties that come from unmet demands. Hence, as the demand increases, the objective value always increases.

We outline some facts about the random variables and optimal solution for each problem in Table 4.2. The random variables for all the problems are independent from each other. For the distribution column, we use 'uniform' to refer to a uniform distribution over all possible values, and 'irregular' for any other distribution. Additionally, we note that the distribution of each of the random variables of LandS approximate a linear function as it has a uniform distribution over evenly spaced points. The upper- and lower-bound estimates (with 95% confidence intervals) are obtained from (Linderoth et al., 2006, Table 4) with $n = 5000$. We will refer to the quantities in this table in our discussion of the computational results.

Table 4.2: Properties of our stochastic programming test problems

	Random Variables			Bounds (95% confidence interval)					
	Number	Possible Values	Distribution	Lower			Upper		
20term	40	2	Uniform	254298.57	±	38.74	254311.55	±	5.56
gbd	5	13 - 17	Irregular	1655.62	±	0.00	1655.628	±	0.00
LandS	3	100	Uniform	225.62	±	0.02	225.624	±	0.005
ssn	86	4 - 7	Irregular	9.84	±	0.10	9.913	±	0.022
storm	117	5	Uniform	15498657.8	±	73.9	15498739.41	±	19.11

4.7 Computational Results

In this section we will summarize the computational results and observations. For each combination of sampling scheme, number of scenarios, and number of batches, we perform the lower bound estimation process 100 times, and compute the mean and standard error of the 100 lower bounds obtained.

Before we proceed with our observations, we should emphasize that while we use the same problems as in Linderoth et al. (2006), our results are not directly comparable. In particular, Linderoth et al. (2006) focus on results based on analysis of the objective values across 10 uncorrelated subproblems of a single replicate (or similarly, across 10 replicates of one subproblem each). We focus on the analysis of the mean SAA objective values between many (potentially correlated) subproblems across 100 replicates. Hence, the results in Linderoth et al. (2006) yield a measure of the quality of a single lower-bound estimate, while our results compare the quality of several different lower bound estimates.

All data used in this analysis — specifically, the objective values obtained by solving each LP corresponding to each batch — can be obtained at the web site for this paper at `http://pages.cs.wisc.edu/~conghan/slhd/`.

Table 4.3 shows the wall-clock time required to solve a single sampled approximation for the five test problems using our timing setup as described in § 4.6. Even though the time to generate a batch of scenarios increases as the sampling method increases in complexity, this time is insignificant compared to the rest of the process and hence was not included in this table. The timings were obtained for each problem and number of scenarios by averaging across 16 subproblems constructed

from all design methods we considered. The timing increases with the number of scenarios at a superlinear rate. Hence, when the underlying problem is difficult and the number of scenarios is sufficiently high, solving many batches could be extremely time consuming even with multiple machines.

Table 4.3: Wall clock times (in seconds) for solving one sample approximation problem with varying numbers of scenarios

	no. of scenarios		
	256	512	1024
20term	4.83	11.12	30.09
gbd	0.03	0.10	0.17
LandS	0.07	0.12	0.25
ssn	10.36	23.80	61.63
storm	16.40	37.85	84.22

In our first set of experiments, we computed the means and standard errors of the lower bound estimates of the objective value over 100 replicates of each sampling method and each number of scenarios. We fixed the number of subproblems at $t = 16$, while varying the number of scenarios per batch according to $n \in \{128, 256, 512, 1024\}$. We tested four different sampling methods: Monte Carlo (MC), independent ordinary Latin hypercube (ILH), Sliced Latin hypercube (SLH) and Sliced Partial Orthogonal Array Based Latin hypercube based on Bush orthogonal arrays (BUSH), which are $OA(n^2, m+1, n, 2)$. Since we are only using 16 out of n possible batches, we are discarding much of the orthogonal array, and therefore not achieving much two-dimensional stratification in our negatively dependent designs. Computational results are summarized in Table 4.4.

We begin with some general observations about the mean of the lower bound estimates. As observed in other experiments (Linderoth et al., 2006; Freimer et al.,

Table 4.4: Mean and variance of estimates of the lower bound with 16 batches, over 100 replicates.

		128 scenarios		256 scenarios		512 scenarios		1024 scenarios	
		Mean	SE	Mean	SE	Mean	SE	Mean	SE
20term	MC	254253.1	244.1	254292.1	160.2	254297.4	95.1	254311.3	69.3
	ILH	254296.7	58.2	254311.1	39.8	254306.4	25.6	254311.0	19.9
	SLH	254285.9	53.3	254303.0	45.0	254307.6	31.5	254310.0	23.8
	BUSH	254296.0	54.8	254299.2	38.3	254307.5	30.2	254312.8	22.1
gbd	MC	1653.207	14.292	1654.061	10.540	1655.124	6.881	1656.837	5.585
	ILH	1655.637	1.094	1655.760	0.670	1655.606	0.260	1655.616	0.157
	SLH	1655.640	0.281	1655.622	0.171	1655.615	0.066	1655.635	0.044
	BUSH	1655.602	0.275	1655.609	0.167	1655.645	0.080	1655.632	0.043
LandS	MC	225.3951	1.2923	225.7044	0.9876	225.5539	0.6106	225.6881	0.4622
	ILH	225.6314	0.0549	225.6233	0.0332	225.6249	0.0278	225.6301	0.0158
	SLH	225.6135	0.0471	225.6248	0.0370	225.6259	0.0255	225.6295	0.0177
	BUSH	225.6159	0.0522	225.6191	0.0319	225.6274	0.0266	225.6295	0.0185
ssn	MC	7.426	0.387	8.403	0.287	9.028	0.188	9.411	0.142
	ILH	8.945	0.267	9.378	0.186	9.635	0.150	9.770	0.103
	SLH	8.877	0.225	9.321	0.203	9.609	0.137	9.775	0.087
	BUSH	8.929	0.258	9.374	0.181	9.656	0.134	9.785	0.091
storm	MC	15498662.4	7441.1	15498518.5	5517.1	15498532.9	3648.5	15498473.7	2838.3
	ILH	15498741.0	454.9	15498678.8	257.0	15498698.9	151.2	15498716.6	98.2
	SLH	15498690.0	245.2	15498683.9	159.0	15498695.4	104.6	15498721.5	79.3
	BUSH	15498699.6	238.7	15498731.9	149.6	15498688.7	115.6	15498709.3	85.4

2012), the Monte Carlo method is significantly worse than Latin Hypercube-based methods in terms of the bias. As expected, the Latin Hypercube-based methods produce statistically-indistinguishable lower bounds.

By comparing with the results of Table 4.2, for all problems except ssn, ILH attains a mean extremely close to the true mean when the number of scenarios is 1024, and is already very close with a smaller number of scenarios. ssn is known to be a challenging problem, requiring at least 512 scenarios to attain a reasonable estimate of the optimum even for the three Latin hypercube schemes. Increasing the number of scenarios beyond 1024 would continue to improve the quality of the estimates. Linderoth et al. (2006) provide a more detailed description of the behavior

of SAA for the ssn problem.

We now turn to the standard error. By this measure, for every problem except ssn, we see that MC performs much worse than the other methods. It is only slightly worse for ssn. This table shows situations in which the negatively dependent designs of SLH and BUSH begin to distinguish themselves from ILH. For gbd and storm, we can see large improvements from ILH to SLH/BUSH. The improvement from ILH to SLH/BUSH is much less pronounced in ssn, being in general smaller than the improvement from MC to ILH, and in one case performing slightly worse than ILH. However, we should note that in all the problems, BUSH and SLH have roughly similar performance. This observation suggests that when we use too few subproblems, the effect of partial orthogonality on performance is not significant.

For 20term and LandS, the Latin hypercube schemes ILH, SLH, and BUSH perform similarly. In the case of 20term, this similarity is unsurprising. The benefits of SLH over ILH come from the increased stratification that comes from the sliced structure, but since each variable can only take on two values, each with probability 0.5, any stratification that divides the probability space into a multiple of two would perform equally well. In the case of LandS, we suspect that the similarity of performance is due to the smoothness of the distribution of the random variables. In fact, the cumulative distribution function is essentially linear. We have found that when we modify the distribution to be more irregular or to to be a uniform distribution over a much smaller set, it tends to drive up the standard error and to cause SLH to have a significantly smaller standard error than ILH.

We now consider a greater number of subproblems for each n and new alterna-

tives for the underlying orthogonal arrays. In addition to the four different sampling methods considered earlier, we show two additional variants: Sliced Orthogonal Array Based Latin hypercube based on Bose-Bush orthogonal arrays (BB), and independent batches taken over several BB (INDBB). Bose-Bush orthogonal arrays have an $\mathrm{OA}(\lambda s^2, m+1, s, 2)$ structure. We pick s to be equal to the number of subproblems, and define $\lambda = n/s$. With these choices, the sliced designs achieve full two-dimensional stratification, making BB an example of SOLH sampling.

We include INDBB in our experiments to help isolate the factors that lead to the stronger performance of BB. If each slice of the BB design has some special structure that leads to improved performance, then INDBB designs should perform better than ILH, and the performance of INDBB should be close to that of BB. However, if the performance gains of BB are primarily due to the better two-dimensional stratification, then we would expect INDBB to perform no better than ILH. The numerical results support the second claim.

Results are shown in Table 4.5. Many of the observations about MC/ILH/SLH/BUSH from Table 4.4 carry over. Also, comparing the standard error of MC/ILH/SLH/BUSH between the two tables, we notice a factor of $\sqrt{2}$ or 2 difference in standard error, depending on whether the number of subproblems was doubled or quadrupled. This factor is consistent with Theorem 4.7. The lower bound estimates are roughly the same in both tables, demonstrating that changing the number of subproblems does not affect the bias.

Table 4.5 shows a considerable advantage for BB over ILH. In fact, BB is better than all other methods tested, except on gbd, where it performs similarly to

Table 4.5: Mean and standard error of estimates of the lower bound with 32 or 64 batches (over 100 replicates)

(scenarios, batches)		(128,32)		(256,32)		(512,64)		(1024,64)	
		Mean	SE	Mean	SE	Mean	SE	Mean	SE
20term	MC	254295.4	164.5	254305.0	116.3	254301.1	54.4	254313.4	36.9
	ILH	254305.7	43.4	254296.4	29.2	254307.0	16.8	254309.1	9.9
	SLH	254294.2	45.8	254306.3	28.5	254306.2	14.3	254308.5	11.3
	BUSH	254293.7	36.5	254305.3	27.8	254306.7	13.6	254309.4	10.1
	BB	254296.1	20.9	254305.3	18.9	254307.3	7.6	254310.3	4.9
	INDBB	254294.4	39.3	254301.1	29.9	254308.3	15.2	254309.1	11.1
gbd	MC	1653.130	9.485	1654.699	7.091	1655.488	3.389	1655.655	2.986
	ILH	1655.550	0.849	1655.658	0.390	1655.633	0.164	1655.628	0.094
	SLH	1655.649	0.169	1655.620	0.066	1655.628	0.017	1655.628	0.011
	BUSH	1655.628	0.163	1655.628	0.066	1655.629	0.017	1655.628	0.011
	BB	1655.614	0.170	1655.626	0.072	1655.629	0.018	1655.627	0.011
	INDBB	1655.618	0.799	1655.582	0.483	1655.618	0.140	1655.641	0.096
LandS	MC	225.6448	0.9108	225.6300	0.6092	225.6431	0.2844	225.5845	0.2202
	ILH	225.6151	0.0344	225.6190	0.0248	225.6255	0.0131	225.6298	0.0088
	SLH	225.6172	0.0351	225.6260	0.0263	225.6253	0.0124	225.6287	0.0087
	BUSH	225.6155	0.0332	225.6247	0.0225	225.6260	0.0116	225.6269	0.0081
	BB	225.6178	0.0068	225.6247	0.0047	225.6270	0.0015	225.6282	0.0010
	INDBB	225.6202	0.0370	225.6220	0.0252	225.6258	0.0123	225.6281	0.0085
ssn	MC	7.426	0.275	8.412	0.197	9.011	0.094	9.389	0.071
	ILH	8.908	0.184	9.378	0.127	9.628	0.073	9.767	0.045
	SLH	8.911	0.198	9.386	0.131	9.614	0.064	9.771	0.054
	BUSH	8.905	0.175	9.390	0.123	9.634	0.064	9.770	0.051
	BB	8.925	0.105	9.408	0.083	9.627	0.029	9.767	0.022
	INDBB	8.938	0.185	9.381	0.134	9.620	0.078	9.763	0.049
storm	MC	15499036.3	5185.4	15498564.6	3653.0	15498659.9	2039.3	15498513.0	1263.0
	ILH	15498674.5	377.4	15498717.2	179.3	15498690.9	68.6	15498715.7	43.3
	SLH	15498658.3	149.0	15498712.7	99.8	15498699.4	53.4	15498718.1	37.0
	BUSH	15498687.3	133.0	15498707.0	104.1	15498701.9	47.0	15498721.9	38.4
	BB	15498686.1	76.1	15498710.4	42.5	15498693.4	22.0	15498720.7	12.6
	INDBB	15498674.6	297.4	15498712.5	201.2	15498695.8	67.9	15498720.7	43.1

SLH/BUSH. On LandS, BB performs about 5-10× better than the other sliced sampling methods. A possible explanation for this huge improvement is that the total number of random variables is just three, so having two-dimensional stratification would cover a large portion of the possible interactions between variables. In the case of ssn, the improvement from ILH to BB is comparable to the improvement from MC to ILH, a bigger factor than is observed for any other problem.

Finally, we turn our attention to running times. The standard error of the estimates between the results for 512 scenarios and 16 slices in Table 4.4 and 256 scenarios and 32 slices in Table 4.5 are similar. Since the timing scales superlinearly in the number of scenarios, the amount of time it takes to solve ct sampled approximations of n scenarios sequentially can be substantially less than solving t sampled approximations of cn. Each batch could also be solved independently and in parallel. This suggests that if computing resources on each machine is limited and using SLH/SPOLH with cn scenarios and t batches is computationally infeasible, using SOLH with with n scenarios and ct batches can be an effective way of reducing the standard error.

We conclude that for a fairly small number of subproblems, the sliced sampling methods perform at least as well as Latin hypercube sampling, and in fact show significant improvement in some cases. Once we increase the number of subproblems and exploit the full "orthogonality" property of the orthogonal arrays, we see a substantial improvement in *all* cases. Thus, if the computational budget will only allow a small number of batches for the given value of n for which a lower bound v_n is being estimated, there is significant computational benefit to using the more sophisticated sampling methods introduced in this work.

4.8 Conclusions and Future Work

In this paper, we propose the use of two types of negatively dependent designs to improve the lower bound of the objective value. Sliced Latin hypercube sampling

is easy to implement since SLH does not impose any restriction on the number of batches t and the number of scenarios in each batch. We introduce the concept of monotonicity for two-stage stochastic linear programs, and provide a non-asymptotic result showing that SLH can be better than ILH for problems with this monotonicity property. On the other hand, we show that SLH is asymptotically equivalent to ILH if the distribution of the random vector has finite support and the approximate solutions converge. Our computational results supports the theory, showing that SLH performs no worse than ILH and in some cases performs significantly better than ILH.

To improve upon SLH, we consider sliced orthogonal array-based Latin hypercube sampling schemes, which achieve stronger negative dependence between batches. The choice of the underlying orthogonal array can make a huge difference in variance reduction. We provide empirical results showing that when we are able to exploit the full orthogonality of the underlying orthogonal array, using Bose-Bush orthogonal arrays (Bose and Bush, 1952), the performance is significantly better than when we use only part of an orthogonal array.

Our work treats Latin hypercube sampling as the baseline method, in part because it was investigated in earlier work (Linderoth et al., 2006). Other sampling methods, such as U sampling (Tang, 1993; Tang and Qian, 2010) and randomized quasi-Monte Carlo (Niederreiter, 1992; Owen, 1995; Homem-de Mello, 2008) could have been used instead for constructing a single SAA problem. We can carry over the idea of negatively dependent designs to these advanced within-batch sampling techniques. For U sampling, a strength-two orthogonal array-based Latin hypercube

design could be generated for each SAA problem. The t underlying strength-two orthogonal arrays can be obtained by slicing a larger strength-three orthogonal array. For randomized quasi-Monte Carlo, we can sample different batches based on the same low-discrepancy sequence such that batches are negatively dependent spontaneously. Comparing with Latin hypercube sampling, both U sampling and randomized quasi-Monte Carlo are extremely complicated to implement as they require more constrains on the selection of batch size n and the number of batches t. A potential research direction in the future for us is to study the theoretical and empirical performance of negatively dependent batches based on these advanced sampling methods.

Chapter 5

Controlling Correlations in Sliced Latin Hypercube Designs

5.1 Background

Qian (2012) proposed sliced Latin hypercube designs for collective evaluations of computer models and ensembles of multiple computer models. An $N \times m$ sliced Latin hypercube designs is a Latin hypercube design that can be partitioned into t smaller ordinary Latin hypercube designs with n runs each where $n = N/t$. A sliced Latin hypercube design can be used to evaluate a computer model in batches where each batch takes input values from one slice of the design, such that the expected output $\mu = E\{f(X)\}$ in Section 1.1 can be estimated accurately based on combined batches or separated batches. It is desirable to control correlations for each slice and the entire design simultaneously. Yang et al. (2013) proposed methods to construct

a sliced orthogonal Latin hypercube design where the whole design is orthogonal and can be divided into slices of smaller orthogonal Latin hypercube designs. This section propose an algorithm to construct a sliced Latin hypercube designs with correlation controlled for the entire design and for every single slice of smaller Latin hypercube designs, which is ready for the applications in Qian (2012). Our construction method is based on a modified version of the RGS algorithm in Section 1.4 and is flexible in run size.

Let SLH denote a scheme that produce an $N \times p$ sliced Latin hypercube design D with t slices $D_1, \ldots, D_t$, where each slice is a smaller Latin hypercube design of n runs. Let $d_{r,i}$, $d_r(:,k)$ and $d_{r,ik}$ denote the ith run, kth column and kth entry in the ith run of D_r, respectively for $i = 1, ..., n$, $k = 1, ..., m$ and $r = 1, ..., t$. The desired $D_1, \ldots, D_t$ can be constructed by Algorithm 4.1. We obtain a sliced Latin hypercube design D by stacking $D_1, \ldots, D_t$ row by row together. We refer the design thus constructed as an *ordinary sliced Latin hypercube design*.

5.2 Modified RGS algorithm

Inspired by the benefit of controlling correlations in ordinary Latin hypercube designs, we propose to control correlations for each slice and the entire sliced Latin hypercube design simultaneously. Let D_{-r} denote the design obtained by removing D_r from D. Let $\rho_{r,k\ell}$ and $\rho_{-r,k\ell}$ denote the sample correlation between the kth and ℓth columns of D_r and that of D_{-r}, respectively for $k, l = 1, \ldots, m$ and $r = 1, \ldots, m$.

Conditional on the correlation of D_{-r} for every r, we propose a modified RGS

algorithm to control the correlation of D_r around zero such that $\rho_{r,k\ell}$ is closer to $-(t-1)\rho_{-r,k\ell}$ and as a result, $|\rho_{k\ell} \approx \rho_{r,k\ell} + (t-1)\rho_{-r,k\ell}|$ can be reduced. A similar idea has been discussed earlier in § 2.3 for controlling correlations in a sequentially refined Latin hypercube design. In practice, we consider setting $\rho_{r,k\ell} = -c\rho_{-r,k\ell}$ with constant c between 0 and $t-1$. The choice of c will be discussed in details in § 5.4.

Given D_{-r}, our proposed algorithm works as follows.

Algorithm 5.1

Forward:

$$\begin{aligned} &\text{for} \quad k = 2, ..., m \\ &\qquad \text{for} \quad \ell = 1, ..., k-1 \\ &\qquad\qquad d_r^\ell \leftarrow \texttt{takeout}(d_r^k, d_r^\ell) - c\rho_{r,k\ell} d_r^k \\ &\text{for} \quad k = 1, ..., m \\ &\qquad d_r^k \leftarrow \{a_r^k[\texttt{rank}(d_r^k)] - .5\}/nt, \end{aligned} \tag{5.1}$$

Backward:

$$\begin{aligned} &\text{for} \quad k = m-1, ..., 1 \\ &\qquad \text{for} \quad \ell = m, ..., k+1 \\ &\qquad\qquad d_r^\ell \leftarrow \texttt{takeout}(d_r^k, d_r^\ell) - c\rho_{k\ell}^{-(r)} d_r^k \\ &\text{for} \quad k = 1, ..., m \\ &\qquad d_r^k \leftarrow \{a_r^k[\texttt{rank}(d_r^k] - .5\}/nt. \end{aligned} \tag{5.2}$$

where the vector $a_r^k = \lceil ntd_r^k \rceil$, $a_r^k[\texttt{rank}(d_r^k)]$ is a permutation of a_r^k such that $\texttt{rank}\{a_r^k[\texttt{rank}(d_r^k)]\} = \texttt{rank}(a_r^k)$.

Compared with Owen's algorithm (Algorithm 1.2), this new algorithm includes an extra term in (5.1) and (5.2). To exam if Algorithm 5.1 can indeed move $\rho_{r,k\ell}$ closer to $-c\rho_{-r,k\ell}$ effectively, define modified root mean square correlation. as

$$\gamma_r(c) = \sqrt{\frac{\sum_{1 \le k < \ell \le q} (\rho_{r,k\ell} - c\rho_{-r,k\ell})^2}{p(p-1)/2}}. \tag{5.3}$$

Note that $\gamma_r(0) = \rho_{rms}(D_r)$.

As pointed out by Owen (1994a) and Tang (1998), the convergence behavior of RGS type algorithms is difficult or unrealistic to prove. We will provide some empirical results instead. For relatively small m, the algorithm usually converges within eight complete passes, which means after alternating the modified forward and backward algorithm eight times, either (i) D_r stops changing or (ii) D_r switches between two designs. This is the same phenomenon observed for the original RGS algorithm Owen (1994a). To study the asymptotic behavior of $\gamma_r(c)$, our algorithm is applied for $n = 10, 20, 50, 100, 200, 500$. Each value of n has 100 replications. We fix $t = 5$, $r = 1$ and $c = 1$ for illustration. Figure 5.1 presents a series of boxplots of $\gamma_1(1)$ versus n on a log-log scales. Ordinary least squares regression results in

$$\log(\gamma_1(1)) = -1.13 \log(n) - 0.29,$$

which has a similar slope as ρ_{rms} in Owen (1994a). The regression line is plotted as a solid reference line.

5.3 Algorithm for correlation-controlled sliced Latin hypercube designs

We propose an algorithm to construct a *correlation-controlled sliced Latin hypercube design* with $\rho_{rms}(D) = o_p(N^{-1/2})$ and $\rho_{rms}(D_r) = o_p(n^{-1/2})$ for $r = 1, \ldots, r$. The algorithm begins with an ordinary sliced Latin hypercube design, and then update

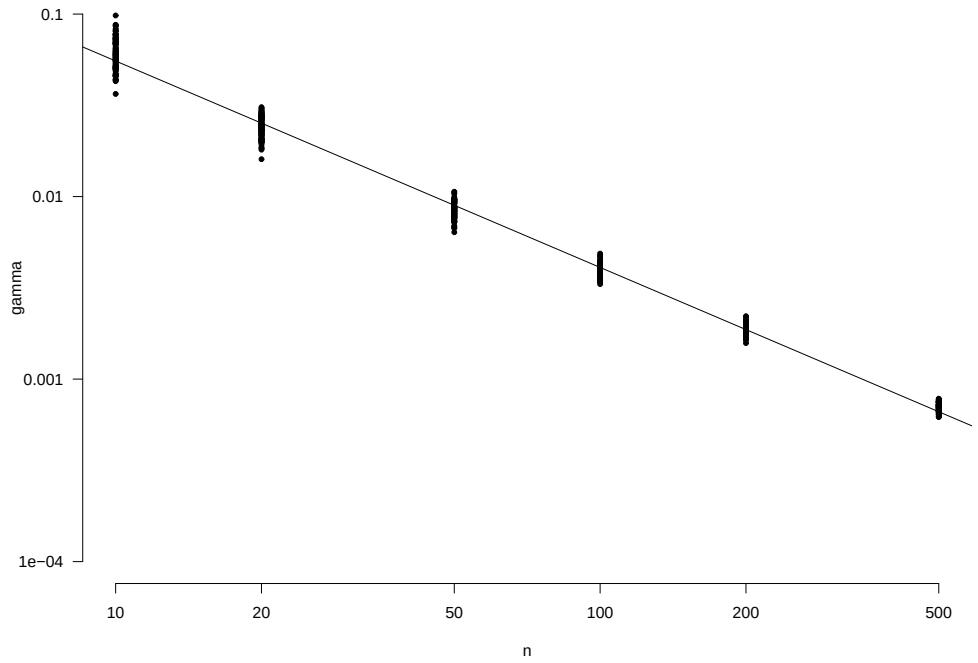

Figure 5.1: Modified root mean square correlation $\gamma_1(1)$ versus sample size n for $p = 9$ dimensional sliced Latin hypercube design with $t = 5$ slices. Dots denote $\gamma_1(1)$ for the modified RGS algorithm. The solid reference line is the least squares regression line of $\log(\gamma_1(1))$ versus $\log(n)$.

each slice independently based on the measure $\gamma_r(0)$ which does not depend on $\rho_{-r,k\ell}$ from other slices. As the slices in an ordinary sliced Latin hypercube design are dependent dimension by dimension, it is reasonable to assume $\rho_{1,k\ell}, \ldots, \rho_{r,k\ell}$ are mutually independent. The initial step guarantees $\rho_{rms}(D_r) = o_p(n^{-1/2})$ for $r = 1, \ldots, t$, hence $\rho_{rms}(D_{-r}) = o_p(N^{-1/2})$ as $\rho_{rms}(D_{-r}) \approx (t-1)^{-1} \sum_{i \neq r} \rho_{rms}(D_i)$. The algorithm then updates each slice based on its modified root mean square correlation based on column-wise correlations of other slices as given in (5.3). The complete

procedure is summarized in Algorithm 5.2.

Algorithm 5.2 Controlling correlations for a sliced Latin hypercube design

1: **g**iven positive integers n, t, p, q_1, q_2, a constant $c \in \mathbb{R}$
2: **g**enerate a $nt \times p$ sliced Latin hypercube designs D with slices $D_1, ..., D_t$ by Algorithm 4.1.
3: **for** $r = 1, \ldots, t$ **do**
4: **u**pdate D_r by Algorithm 5.1 until $\gamma_r(0)$ converges or the number of complete backward and forward steps reaches q_1.
5: **end for**
6: $b \leftarrow 0$
7: **while** $b < t - 1$ or the number of iterations reaches q_2 **do**
8: **for** $r = 1, \ldots, t$ **do**
9: **c**ompute $\rho_{-r,k\ell}$ for any $k \neq \ell$ and $r = 1, \ldots, m$
10: **c**alculate a new D_r by the modified RGS algorithm until $\gamma_r(c)$ converges or the number of complete backward and forward steps reaches q_1.
11: **if** $\rho_{rms}(D)$ decreases based on the new D_r **then**
12: **u**pdate the D_r as the new D_r
13: $b \leftarrow 0$
14: **else**
15: $b \leftarrow b + 1$
16: **end if**
17: **end for**
18: **end while**
19: **r**eturn $D_1, \ldots, D_t$

The new RGS algorithm is deterministic given the constant c since the modified RGS always leads to the same new D_r based on the same set of $D_1, \ldots, D_t$. Thus, no change can be made on line 10 in Algorithm 5.2 if there are no updates on previous $t - 1$ slices.

We use $q_1 = q_2 = 10$ in Algorithm 5.2 unless stated otherwise. These two numbers specify the maximum number of complete passes in new RGS algorithm and the maximum number of while loop starting at line 7.

5.4 Performance

We address the issue of choosing appropriate constant c in Algorithm 5.2. Several candidates of c will be considered and their performance will be compared.

Definition 5.1. *Let CSLH(c) denote a scheme that uses Algorithm 5.2 with constant c to try to produce an $N \times m$ correlation-controlled sliced Latin hypercube design with t slices, where each slice is an $n \times m$ correlation-controlled Latin hypercube design.*

The first candidate we would consider is $c = 0$. In this case, Algorithm 5.2 stops at line 5 and results in $\rho_{rms}(D_r) = O_p(n^{-1})$ and $\rho_{rms}(D) = O_p(t^{-1/2}n^{-1})$ according to the performance of the ordinary RGS algorithm in Section 1.4. Although CSLH(0) can produce a correlation-controlled sliced Latin hypercube design defined in Section 5.3, it only guarantees $\rho_{rms}(D) = O_p(t^{-1/2}n^{-1})$ while $\rho_{rms}(D) = O_p(N^{-1})$ in CLH if we give up the sliced structure.

The performance of CSLH(0) for the entire design D is compromised because we did not update each slice using the information from other slices. As $\rho_{k\ell} \approx t^{-1}\sum_{r=1}^{t}\rho_{r,k\ell}$ and $\rho_{r,k\ell} \approx (t-1)^{-1}\sum_{i\neq r}\rho_{r,k\ell}$, it is desirable to use $c = (t-1)$ such that $\rho_{k\ell} \approx 0$ after the modified RGS algorithm in (5.1) and (5.2).

However, CSLH$(t-1)$ has its drawback. According to line 8 in Algorithm 5.2, the expected $\rho_{1,k\ell}$ is $-(t-1)\rho_{-1k\ell} \approx -\sum_{r=2}^{t}\rho_{r,k\ell} = O_p(t^{1/2}n^{-1})$ in the first iteration. Hence, CSLH$(t-1)$ can cause an inflation in $\rho_{rms}(D_1)$ and maybe $\rho_{rms}(D_2), \ldots$. The third candidate is then $c = \sqrt{t-1}$ such that the expected $\rho_{1,k\ell}$ is $O_p(t^{1/2}n^{-1})$ the same as under CSLH(0). Meanwhile, CSLH$(\sqrt{t-1})$ is still able to reduce $\rho_{rms}(D)$ in CSLH(0) as the expected $\rho_{r,k\ell}$ has the opposite sign of $\rho_{-r,k\ell}$.

We now exam the performance of CSLH(0), CSLH($t-1$) and CSLH($\sqrt{t-1}$) with different n, t and fixed $m = 9$. Results are presented in Tables 5.1 and 5.2. Table 5.1 reports the averages of $\rho_{rms}(D_1)$, $\rho_{rms}(D_2)$ and $\rho_{rms}(D_t)$ based on 100 replicates. According to Table 5.1, CSLH(0) generate slices with similar root mean square correlation because slices are independently optimized. CSLH($t-1$) fails to control correlations in the first several slices especially for relatively small n and large t. On the other hand, CSLH($\sqrt{t-1}$) performs almost as well as CSLH(0). When n is large, we observe that the result in $CSLH(0)$ and CSLH($\sqrt{t-1}$) are equivalent while CSLH($t-1$) is still unable to control the $\rho_{rms}(D_1)$.

Table 5.2 presents some evidence showing that CSLH($t-1$) is most effective in controlling $\rho_{rms}(D)$ followed by CLSH($\sqrt{t-1}$). The difference in $\rho_{rms}(D)$ between three schemes becomes more significant as t grows. Meanwhile comparing with CLSH(0), the computation time required for CSLH($t-1$) and CLSH($\sqrt{t-1}$) is within a reasonable range.

Let D_{rnd} denote a randomly selected slice in D. Figure 5.2 plots the $\rho_{rms}(D)$, the $\rho_{rms}(D_1)$ and the $\rho_{rms}(D_{rnd})$ in different columns on a log-log scale, respectively. When $n = 10$, CSLH($\sqrt{t-1}$) outperforms CSLH(0) in all three comparisons. CSLH($t-1$) is the best scheme in controlling the $\rho_{rms}(D)$ while it is the worst in controlling the average of the $\rho_{rms}(D_{rnd})$ due to the poor performance in $\rho_{rms}(D_1)$. The same results are observed when $n = 100$. Figure 5.3 plots the same comparisons except that comparisons are made with fixed t. Results with $t = 3$ and $t = 20$ are similar. We found CSLH($\sqrt{t-1}$) may fall behind other two schemes when t is large and n is even much larger than t.

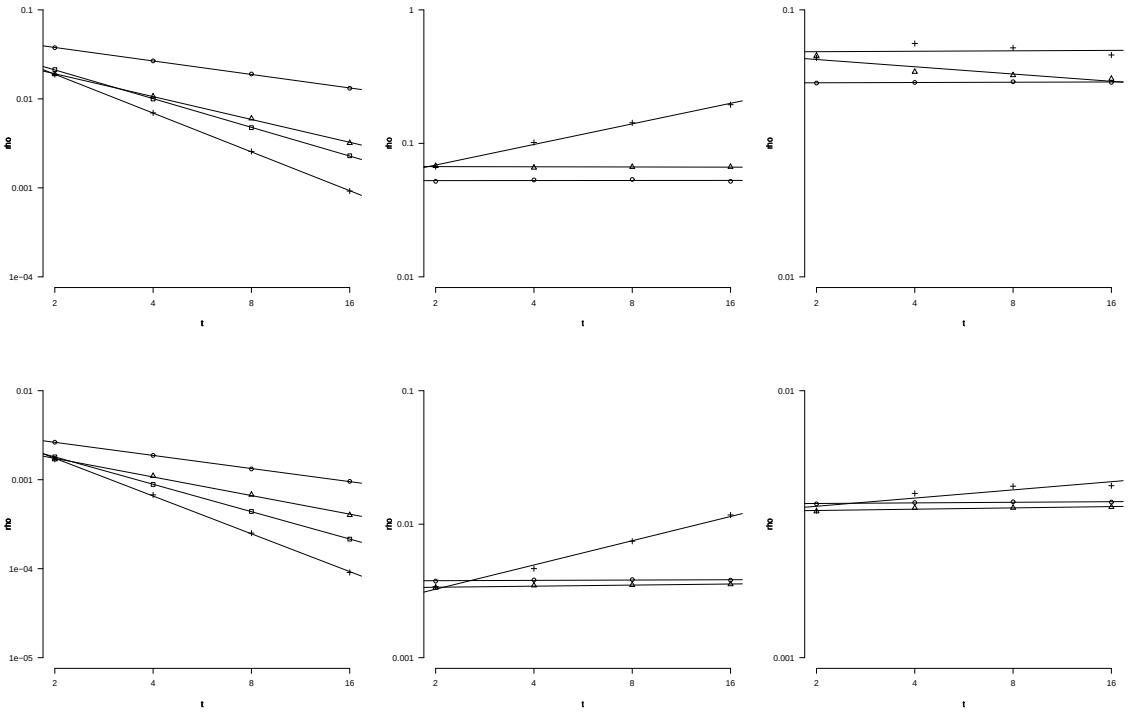

Figure 5.2: The $\rho_{rms}(D)$ (left), the $\rho_{rms}(D_1)$ (middle) and the average of the $\rho_{rms}(D_r)$ (right) versus t for $n = 10$ (top) and $n = 100$ (bottom) on a log-log scale. Means based on different schemes are calculated based on 100 replicates for CLH ($\times$), CSLH(0) ($\square$), CSLH($\sqrt{t-1}$) ($\triangle$) and CSLH($t-1$) ($+$). The solid reference lines are the least squares regression lines of the $\rho_{rms}(D)$ (left), the $\rho_{rms}(D_1)$ (middle) and the $\rho_{rms}(D_r)$ versus $\log(t)$, respectively.

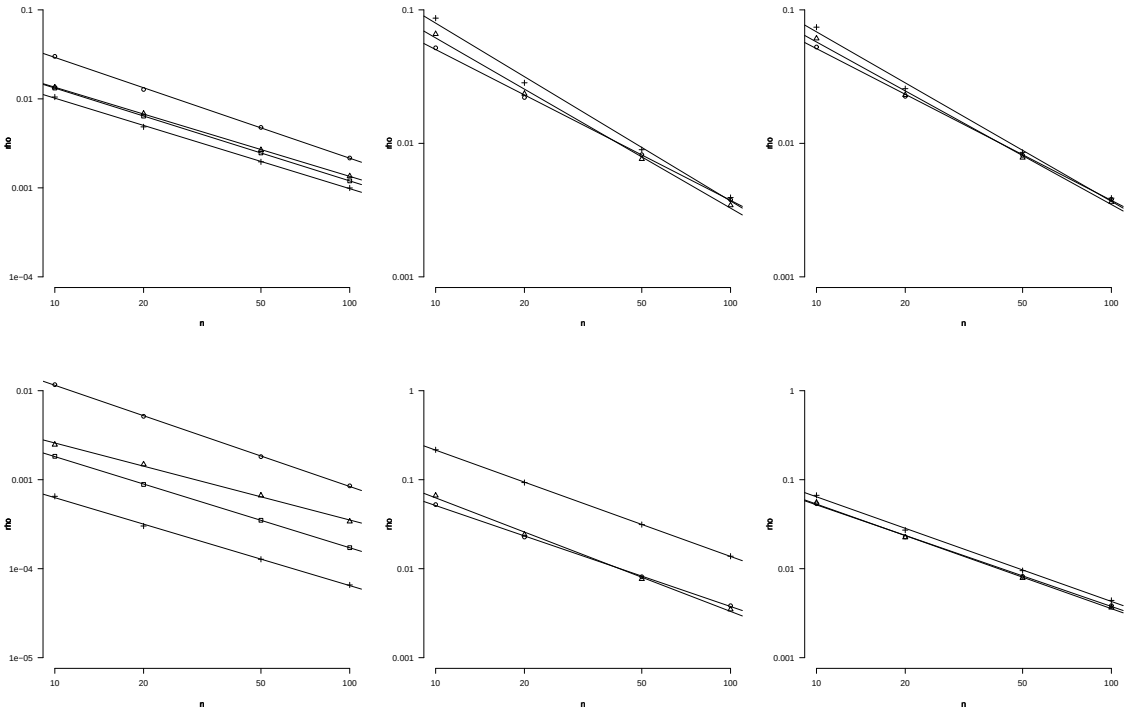

Figure 5.3: The $\rho_{rms}(D)$ (left), the $\rho_{rms}(D_1)$ (middle) and the average of the $\rho_{rms}(D_r)$ (right) versus n for $t = 3$ (top) and $t = 20$ (bottom) on a log-log scale. Means based on different schemes are calculated based on 100 replicates for CLH ($\times$), CSLH(0) ($\square$), CSLH($\sqrt{t-1}$) ($\triangle$) and CSLH($t-1$) ($+$). The solid reference lines are the least squares regression lines of the $\rho_{rms}(D)$ (left), the $\rho_{rms}(D_1)$ (middle) and the $\rho_{rms}(D_{rnd})$ versus $\log(n)$, respectively.

Table 5.1: Average of root mean square correlation for the first, second and last slice under CSLH(0), CSLH($t-1$) and CSLH($\sqrt{t-1}$) based on 100 replicates. The numbers in parentheses are the corresponding standard error. Numerical values are the actual values multiplied by 100

t	Schemes	$n = 10$			$n = 100$		
		$\rho_{rms}(D_1)$	$\rho_{rms}(D_2)$	$\rho_{rms}(D_t)$	$\rho_{rms}(D_1)$	$\rho_{rms}(D_2)$	$\rho_{rms}(D_t)$
3	CSLH(0)	5.333	5.247	5.396	0.382	0.384	0.380
		(0.782)	(0.817)	(0.730)	(0.027)	(0.026)	(0.026)
	CSLH($t-1$)	8.540	6.968	6.513	0.401	0.389	0.389
		(1.225)	(1.109)	(0.883)	(0.039)	(0.028)	(0.027)
	CSLH($\sqrt{t-1}$)	6.661	5.811	5.740	0.347	0.362	0.372
		(0.832)	(0.692)	(0.825)	(0.033)	(0.031)	(0.031)
6	CSLH(0)	5.235	5.260	5.260	0.380	0.385	0.383
		(0.856)	(0.734)	(0.766)	(0.034)	(0.031)	(0.031)
	CSLH($t-1$)	12.600	7.374	5.884	0.600	0.400	0.384
		(0.856)	(0.734)	(0.766)	(0.034)	(0.031)	(0.031)
	CSLH($\sqrt{t-1}$)	6.417	5.818	5.440	0.357	0.357	0.371
		(0.856)	(0.734)	(0.766)	(0.034)	(0.031)	(0.031)
20	CSLH(0)	5.253	5.407	5.459	0.387	0.388	0.385
		(0.770)	(0.786)	(0.727)	(0.026)	(0.031)	(0.027)
	CSLH($t-1$)	21.337	8.481	5.409	1.420	0.412	0.381
		(2.398)	(1.328)	(0.626)	(0.211)	(0.035)	(0.028)
	CSLH($\sqrt{t-1}$)	6.592	6.060	5.323	0.357	0.350	0.377
		(0.863)	(0.761)	(0.706)	(0.036)	(0.030)	(0.034)

5.5 Numerical Illustration

In this section, we compare the performance of the estimator $\hat{\mu}$ in (1.1) based on an entire sliced Latin hypercube design D and an single slice under the three schemes studied in § 5.4. We use $\hat{\mu}$, $\hat{\mu}_1$ and $\hat{\mu}_{rnd}$ to denote the estimator based on D, D_1 and D_{rnd}. We also include the variance reduction performance of $\hat{\mu}$ under LH-CC in § 2.4 as a benchmark. The maximum number of iterations q_1 and q_2 in Algorithm 5.2 are both 10. For the borehole function M8 in § 2.4, consider an experiment to estimate its

Table 5.2: Average of root mean square correlation for the first, second and last slice under CSLH(0), CSLH($t-1$) and CSLH($\sqrt{t-1}$) based on 100 replicates. The numbers in parentheses are the corresponding standard error. The average number of modified RGS iterations and average cpu time (second) for each replicate are given

t	Schemes	$n = 10$			$n = 100$		
		$100\rho_{rms}(D)$	q_2^*	cpu	$1000\rho_{rms}(D)$	q_2^*	cpu
3	CSLH(0)	3.042 (0.381)	0	0.36	2.202 (0.255)	0	0.26
	CSLH($t-1$)	1.047 (0.136)	3.1	0.82	0.993 (0.069)	2.5	0.65
	CSLH($\sqrt{t-1}$)	1.398 (0.180)	2.7	0.73	1.333 (0.082)	2.3	0.58
6	CSLH(0)	2.119 (0.272)	0	0.66	1.546 (0.198)	0	0.53
	CSLH($t-1$)	0.377 (0.052)	3.6	1.69	0.377 (0.027)	3.0	1.39
	CSLH($\sqrt{t-1}$)	0.779 (0.110)	3.1	1.56	0.841 (0.052)	2.8	1.35
20	CSLH(0)	1.214 (0.164)	0	2.33	0.879 (0.101)	0	1.64
	CSLH($t-1$)	0.065 (0.010)	4.1	5.97	0.066 (0.004)	3.3	4.87
	CSLH($\sqrt{t-1}$)	0.258 (0.032)	3.9	6.05	0.339 (0.018)	3.1	4.74

mean output by t machines in parallel, where each machine evaluates the function at n sites. For all methods, $\hat{\mu}$, $\hat{\mu}_1$ and $\hat{\mu}_{rnd}$ are computed 1000 times. Table 5.5 reports root mean squared errors (RMSE) of $\hat{\mu}$, $\hat{\mu}_1$ and $\hat{\mu}_{rnd}$ with different n and t. We find CSLH($t-1$) and CSLH($\sqrt{t-1}$) are more effective than CSLH(0) in reducing the RMSE of μ while CSLH($t-1$) and CSLH($\sqrt{t-1}$) are almost identical. The first slice in CSLH($t-1$) always performs the worst especially when t gets large, but the performance of a randomly selected slice under CSLH($t-1$) and CSLH($\sqrt{t-1}$) are comparable to that under CSLH(0). We also observe that the difference diminishes when n is much larger than t.

Table 5.3: Comparison of the RMSEs of $\hat{\mu}$, $\hat{\mu}_1$ and $\hat{\mu}_{rnd}$ for function M1 under CSLH(0), CSLH($t-1$) and CSLH($\sqrt{t-1}$) based on 1000 replicates.

		$n=10$			$n=20$			$n=50$			$n=100$		
		$t=2$	$t=8$	$t=32$	$t=2$	$t=8$	$t=32$	$t=2$	$t=8$	$t=32$	$t=2$	$t=8$	$t=32$
$\hat{\mu}$	CSLH(0)	0.484	0.246	0.117	0.291	0.140	0.071	0.160	0.083	0.041	0.111	0.055	0.027
	CSLH($t-1$)	0.400	0.191	0.095	0.256	0.127	0.061	0.163	0.076	0.038	0.111	0.054	0.027
	CSLH($\sqrt{t-1}$)	0.405	0.191	0.094	0.257	0.127	0.065	0.154	0.078	0.038	0.111	0.054	0.029
	LH-CC	0.386	0.179	0.086	0.265	0.121	0.062	0.159	0.074	0.040	0.113	0.054	0.028
$\hat{\mu}_1$	CSLH(0)	1.406	1.579	1.588	0.585	0.659	0.692	0.252	0.267	0.270	0.156	0.164	0.161
	CSLH($t-1$)	1.551	1.864	2.689	0.609	0.760	1.1813	0.258	0.286	0.413	0.168	0.164	0.221
	CSLH($\sqrt{t-1}$)	1.479	1.587	1.594	0.600	0.660	0.663	0.247	0.259	0.265	0.166	0.167	0.167
$\hat{\mu}_{rnd}$	CSLH($t-1$)	1.527	1.615	1.690	0.612	0.672	0.686	0.263	0.262	0.283	0.163	0.166	0.160
	CSLH($\sqrt{t-1}$)	1.457	1.623	1.591	0.594	0.649	0.687	0.250	0.253	0.268	0.165	0.165	0.163

Appendix A

Proofs in Chapter 2

Proof of Proposition 2.4

Proof. Take $D_1 \subset \cdots \subset D_u$ from (2.11). Let $\bar{D}_1 \subset \cdots \subset \bar{D}_u$ denote its counterpart obtained in (2.7). Let x_{ik} and $\bar{x}_{ik}$ denote the (i,k)th entries of D_j and $\bar{D}_j$, respectively.

For $j = 1$, D_1 and $\bar{D}_1$ are both ordinary Latin hypercube designs of n_1 runs and hence have the same distribution.

For $j = 2, \ldots, u$, note that for either D_u or $\bar{D}_u$, all columns are independent and generated by the same mechanism. Hence, it suffices to show the result holds for the first column. Let $\boldsymbol{\pi}_u$ be a nest permutation in (2.11) used to generate the first column of D_u. For $i = 1, \ldots, n_{j-1}$, by (2.11)

$$\left\lceil n_j x_{i1} + \frac{u_{i1}}{t_j} \right\rceil = \left\lceil \frac{\pi(i)}{t_j} \right\rceil,$$

which, because $(n_j x_{i1}, n_j x_{i1} + {t_j}^{-1} u_{i,1}]$ contains no integer, equals $\lceil n_j x_{i1} \rceil$. Thus, C_j in

(2.10) has the same distribution as $C_{j,1}$ in (2.4) obtained from $\{\bar{x}_{i1} : i = 1, \ldots, n_{j-1}\}$ and Z_{n_j}/C_j has the same distribution as $\Gamma_{j,1}$ in (2.5). Therefore, $\boldsymbol{\zeta}_j$ in (2.9) as a uniform permutation of Z_{n_j}/C_j has the same distribution as $\{b_{1,k}, \ldots, b_{n_{j-1},1}\}$ in (2.6) being a uniform permutation of $\Gamma_{j,1}$.

Second, for $i = n_{j-1} + 1, \ldots, n_j$, by (2.11) and (2.10), express x_{i1} as

$$\begin{aligned} \frac{\pi(i) - u_{i1}}{n_u} &= \frac{\{\zeta_j(i - n_{j-1}) - 1\}t_j + e_j - u_{i1}}{n_u}, \\ &= \frac{\zeta_j(i - n_{j-1})}{n_j} - \frac{t_j - e_j + u_{i1}}{n_u}, \end{aligned} \tag{A.1}$$

where the u_{i1} are $U[0, 1)$ random variables, the e_j are discrete uniform random variables with support on Z_{t_j}, and the u_{i1} and the e_j are mutually independent. Because $t_j^{-1}(t_j - e_j + u_{i1})$ is a $U[0, 1)$ random variable and is independent of $\zeta_j(i - n_{j-1})$, $\{x_{i1} : i = n_{j-1} + 1, \ldots, n_j\}$ from (A.1) and $\{\bar{x}_{i1} : i = n_{j-1} + 1, \ldots, n_j\}$ from (2.6) have the same distribution. The proof is now completed. □

Proof of Proposition 2.5

Proof. Assume $a, b \in Z_{n_u}$. Let $\boldsymbol{\pi}_u = \{\pi(1), \ldots, \pi(n_u)\}$ be a nested permutation with u layers defined in Section 2.2, where the n_j and the t_j are defined in (2.2) and (2.8), respectively. Note that the jth layer consisting of the first n_j elements of $\boldsymbol{\pi}_u$ forms a permutation on Z_{n_j}.

(i) For $j = 1, \ldots, u$ and $i = 1, \ldots, n_j - n_{j-1}$, by symmetry, $P\{\tau_{j,i} = a\}$ is the same for any $a \in Z_{n_u}$, and hence equals n_u^{-1}.

(ii) For integers $b > 0$ and $t > 0$, let

$$N_{b,t} = \{c \in Z_{n_u} : \lceil c/t \rceil = \lceil b/t \rceil\}$$

be a block of t consecutive integers. Note that $N_{b,t}^c = Z_{n_u}/N_{b,t}$ has $n_u - t$ elements, where the superscript "c" denotes the set complement.

For $j = 1, \ldots, u$, we have that

$$P(\tau_{j,i_1} = a, \tau_{j,i_2} = b) = P(\tau_{j,i_2} = b | \tau_{j,i_1} = a) P(\tau_{j,i_1} = a), \text{ for } a, b \in Z_{n_u}. \qquad \text{(A.2)}$$

Since both τ_{j,i_1} and τ_{j,i_2} of the jth layer belong to the first n_j elements of $\boldsymbol{\pi}_u$, by definition, $\lceil \tau_{j,i_1}/t_j \rceil \neq \lceil \tau_{j,i_2}/t_j \rceil$. Thus, if $\tau_{j,i_1} = a$, then $\tau_{j,i_2} \in N_{a,t_j}^c$.

For $j = 1$, both τ_{1,i_1} and τ_{1,i_2} are in the first layer of $\boldsymbol{\pi}_u$. The condition $\tau_{1,i_1} = a$ implies τ_{1,i_2} cannot take any value in N_{a,t_1}. There is no further restriction for τ_{1,i_2} from previous layers. By the exchangeability of the elements in N_{a,t_1}^c for τ_{1,i_2}, $P(\tau_{1,i_2} = b | \tau_{1,i_1} = a) = (n_u - t_1)^{-1}$ for $a, b \in Z_{n_u}$. Now by (i) and (A.2), (ii) holds by letting $n_0 = 0$ and $(t_0 - 2t_1)/(t_0 - t_1) = 1$ with $t_0 = \infty$.

For $j = 2, \ldots, u$, given $\tau_{j,i_1} = a$, the elements of N_{a,t_j}^c are not equally likely to be taken by τ_{j,i_2} because of the restriction imposed by $\tau_{j,i_1} = a$ in the $(j-1)$th layer of $\boldsymbol{\pi}_u$. More specifically, τ_{j,i_2} is less likely to be a number close to the block $N_{a,t_{j-1}}$ of the $(j-1)$th layer where a resides, as shown below. Partition N_{a,t_j}^c into

$$g_1 = N_{a,t_{j-1}}/N_{a,t_j} = \{b \in Z_{n_u} : \lceil a/t_{j-1} \rceil = \lceil b/t_{j-1} \rceil, \lceil a/t_j \rceil \neq \lceil b/t_j \rceil\}, \qquad \text{(A.3)}$$

which contains $t_{j-1} - t_j$ elements, and

$$g_2 = N^c_{a,t_{j-1}} = \{b \in Z_{n_u} : \lceil a/t_{j-1} \rceil \neq \lceil b/t_{j-1} \rceil, \lceil a/t_j \rceil \neq \lceil b/t_j \rceil\}, \tag{A.4}$$

which contains $n_u - t_{j-1}$ elements. Given $\tau_{j,i_1} = a \in Z_{n_u}$, τ_{j,i_2} must be in either g_1 or g_2. Thus,

$$\sum_{b \in g_1} P(\tau_{j,i_2} = b | \tau_{j,i_1} = a) + \sum_{b \in g_2} P(\tau_{j,i_2} = b | \tau_{j,i_1} = a) = 1. \tag{A.5}$$

By the exchangeability of the elements in g_1 and that of the elements in g_2, for any $a \in Z_{n_u}$, the conditional probability $P(\tau_{j,i_2} = b | \tau_{j,i_1} = a)$ is the same for all $b \in g_1$, denoted by α_1. Similarly, the conditional probability $P(\tau_{j,i_2} = b | \tau_{j,i_1} = a)$ is the same for all $b \in g_2$, denoted by β_1. Plugging the values of α_1 and β_1 in (A.5) gives

$$(t_{j-1} - t_j)\alpha_1 + (n_u - t_{j-1})\beta_1 = 1. \tag{A.6}$$

Now compute α_1. Given $\tau_{j,i_1} = a \in Z_{n_u}$, g_1 in (A.3) can be partitioned into $s_j - 1$ disjoint blocks, denoted by N_{w_i,t_j} for $i = 1, \ldots, s_j - 1$, where $w_1, \ldots, w_{s_j-1}$ are $s_j - 1$ elements of the $(j-1)$th layer of $\boldsymbol{\pi}_u$. By the construction of $\boldsymbol{\pi}_u$, precisely one element of $w_1, \ldots, w_{s_j-1}$, denoted as w_e, belongs to $N_{a,t_{j-1}} (\supset N_{a,t_j})$.

Given $\tau_{j,i_1} = a \in Z_{n_u}$, for any $b \in g_1$ in (A.3),

$$\{\tau_{j,i_2} = b | \tau_{j,i_1} = a\} = T_1 \cap T_2 \cap T_3,$$

where T_1 denotes the event that N_{b,t_j} equals N_{w_i,t_j} for some $i = 1, \ldots, s_j - 1$ except e, $T_2 = \{\tau_{j,i_2} \in N_{b,t_j}\}$ and $T_3 = \{\tau_{j,i_2} = b\}$. Clearly, $P(T_1) = 1 - (s_j - 1)^{-1}$. Because given T_1, τ_{j,i_2} is equally likely to be contained in any N_{i,t_j} for $i \in \{\tau_{j,1}, \ldots, \tau_{j,n_j - n_{j-1}}\}/\{a\}$, $P(T_2|T_1) = (n_j - n_{j-1} - 1)^{-1}$. By the exchangeability of the elements of N_{b,t_j}, $P(T_3|T_1, T_2) = t_j^{-1}$. A conditioning argument shows that

$$\alpha_1 = P(T_3|T_1, T_2)P(T_2|T_1)P(T_1) = \frac{t_{j-1} - 2t_j}{\{n_u - (n_{j-1} + 1)t_j\}(t_{j-1} - t_j)},$$

which combined with (A.6) yields

$$\beta_1 = \frac{t_{j-1} - t_j}{\{n_u - (n_{j-1} + 1)t_j\}t_{j-1}}.$$

Note that $\alpha_1 < \beta_1$. Plugging in the values of α_1 and β_1 in (A.2) proves (ii).

(iii) For $j, \ell = 1, \ldots, u$ with $j < \ell$ and $a, b \in Z_{n_u}$, we have that

$$P(\tau_{j,i_1} = a, \tau_{\ell,i_2} = b) = P(\tau_{\ell,i_2} = b|\tau_{j,i_1} = a)P(\tau_{j,i_1} = a). \tag{A.7}$$

Because both τ_{j,i_1} (of the jth layer of $\boldsymbol{\pi}_u$) and τ_{ℓ,i_2} (of the ℓth layer of $\boldsymbol{\pi}_u$) are among the first n_ℓ elements of $\boldsymbol{\pi}_u$, $\lceil \tau_{j,i_1}/t_\ell \rceil \neq \lceil \tau_{\ell,i_2}/t_\ell \rceil$. Thus, if $\tau_{j,i_1} = a$, then $\tau_{j,i_2} \in N^c_{a,t_\ell}$. Partition N^c_{a,t_ℓ} into

$$g_3 = N_{a,t_{\ell-1}}/N_{a,t_\ell} = \{b : \lceil a/t_{\ell-1} \rceil = \lceil b/t_{\ell-1} \rceil, \lceil a/t_\ell \rceil \neq \lceil b/t_\ell \rceil, b \in Z_{n_u}\},$$

which contains $t_{\ell-1} - t_\ell$ elements, and

$$g_4 = N^c_{a,t_{\ell-1}} = \{b : \lceil a/t_{\ell-1} \rceil \neq \lceil b/t_{\ell-1} \rceil, \lceil a/t_\ell \rceil \neq \lceil b/t_\ell \rceil, b \in Z_{n_u}\},$$

which contains $n_u - t_{\ell-1}$ elements. Given $\tau_{j,i_1} = a \in Z_{n_u}$, τ_{ℓ,i_2} must be in either g_3 or g_4. Thus,

$$\sum_{b \in g_3} P(\tau_{j,i_2} = b | \tau_{j,i_1} = a) + \sum_{b \in g_4} P(\tau_{j,i_2} = b | \tau_{j,i_1} = a) = 1. \tag{A.8}$$

By the exchangeability of the elements in g_3 and that of the elements in g_4, for any $a \in Z_{n_u}$, the conditional probability $P(\tau_{\ell,i_2} = b | \tau_{j,i_1} = a)$ is the same for all $b \in g_3$, denoted by α_2. Similarly, the conditional probability $P(\tau_{\ell,i_2} = b | \tau_{j,i_1} = a)$ is the same for all $b \in g_4$, denoted by β_2. Plugging α_2 and β_2 into (A.8) gives

$$(t_{\ell-1} - t_\ell)\alpha_2 + (n_u - t_{\ell-1})\beta_2 = 1. \tag{A.9}$$

Express α_2 as

$$P(\tau_{\ell,i_2} = b | \tau_{\ell,i_2} \in N_{b,t_\ell}, \tau_{j,i_1} = a) P(\tau_{\ell,i_2} \in N_{b,t_\ell} | \tau_{j,i_1} = a). \tag{A.10}$$

Since $j < \ell$, τ_{j,i_1} belongs to the $(\ell-1)$th layer of $\boldsymbol{\pi}_u$, which implies that N_{b,t_ℓ} must equal $N_{\tau_{\ell,i},t_\ell}$ for some $i \in Z_{n_\ell - n_{\ell-1}}$. By the exchangeability of $\tau_{\ell,1}, \ldots, \tau_{\ell,n_\ell - n_{\ell-1}}$,

$$P(\tau_{\ell,i_2} \in N_{b,t_\ell} | \tau_{j,i_1} = a) = (n_\ell - n_{\ell-1})^{-1}. \tag{A.11}$$

Similarly, by the exchangeability of the elements of N_{b,t_ℓ},

$$P(\tau_{\ell,i_2} = b | \tau_{\ell,i_2} \in N_{b,t_\ell}, \tau_{j,i_1} = a) = t_\ell^{-1},$$

which together with (A.11) and (A.10) yields $\alpha_2 = (n_u - t_\ell n_{\ell-1})^{-1}$. Plugging the value of α_2 into (A.9) gives $\beta_2 = n_u^{-1}$. By (A.7) and the result in (i), the conclusion in (iii) follows. □

Proof of Proposition 2.6

Proof. (i) Consider the case with $i_1, i_2 = n_{j-1}+1, \ldots, n_j$ and $j = 1, \ldots, u$. By (2.12), express $P(x_{i_1,k} \leq v_1, x_{i_2,k} \leq v_2)$ as

$$\frac{n_j}{n_j - n_{j-1} - 1} \left\{ c_{j,0} v_1 v_2 - c_{j,1} \int_0^{v_1} \int_0^{v_2} \delta_{n_{j-1}}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} \right.$$
$$\left. - c_{j,2} \int_0^{v_1} \int_0^{v_2} \delta_{n_j}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} \right\}, \tag{A.12}$$

where $c_{j,0}$, $c_{j,1}$, and $c_{j,2}$ are defined in (2.12). By the theorem in McKay et al. (1979),

$$- \int_0^{v_1} \int_0^{v_2} \delta_{n_j}(x_{1,k}, x_{2,k}) dx_{1,k} dx_{2,k} \leq - \frac{v_1 v_2}{n_j},$$

which implies that (A.12) is no greater than

$$\frac{n_j}{n_j - n_{j-1} - 1} \left(c_{j,0} v_1 v_2 - c_{j,1} \frac{v_1 v_2}{n_{j-1}} - c_{j,2} \frac{v_1 v_2}{n_j} \right) = v_1 v_2,$$

which proves (2.13).

(ii) Consider the case with $i_1 = n_{j-1}+1, \ldots, n_j$ and $i_2 = n_{\ell-1}+1, \ldots, n_\ell$, $j < \ell$. Without loss of generality, assume $v_1 \le v_2$. By (2.12), express $P(x_{i_1,k} \le v_1, x_{i_2,k} \le v_2)$ as

$$v_1 v_2 - \frac{e_{\ell,1}}{e_{\ell,0}} \int_0^{v_1}\int_0^{v_2} \delta_{n_{\ell-1}}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} - \frac{e_{\ell,2}}{e_{\ell,0}} \int_0^{v_1}\int_0^{v_2} \delta_{n_\ell}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k}. \tag{A.13}$$

Let $\lambda = \lfloor n_{\ell-1} v_1 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Let $v_{10} = v_1 - n_{\ell-1}^{-1}\lambda$ and $v_{20} = v_2 - n_{\ell-1}^{-1}\lambda$, where $0 < v_{10} < n_{\ell-1}^{-1}$.

For $i = \ell - 1$ or ℓ, first observe that

$$\begin{aligned} &\int_0^{v_1}\int_0^{v_2} \delta_{n_i}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} \\ =&\int_0^{\frac{\lambda}{n_i}}\int_0^{\frac{\lambda}{n_i}} \delta_{n_i}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} + \int_{\frac{\lambda}{n_i}}^{v_1}\int_{\frac{\lambda}{n_i}}^{v_2} \delta_{n_i}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k}. \end{aligned} \tag{A.14}$$

Second, observe that

$$\begin{aligned} &-\frac{e_{\ell,1}}{e_{\ell,0}} \int_0^{\frac{\lambda}{n_{\ell-1}}}\int_0^{\frac{\lambda}{n_{\ell-1}}} \delta_{n_{\ell-1}}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} \\ =&\frac{e_{\ell,2}}{e_{\ell,0}} \int_0^{\frac{\lambda}{n_{\ell-1}}}\int_0^{\frac{\lambda}{n_{\ell-1}}} \delta_{n_\ell}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k}. \end{aligned} \tag{A.15}$$

Applying (A.14) and (A.15) to (A.13) gives

$$v_1 v_2 + \frac{1}{e_{\ell,0}} \int_0^{v_{10}}\int_0^{v_{20}} \left\{ \frac{t_\ell}{t_{\ell-1}} \delta_{n_{\ell-1}}(x_{i_1,k}, x_{i_2,k}) - \delta_{n_\ell}(x_{i_1,k}, x_{i_2,k}) \right\} dx_{i_1,k} dx_{i_2,k}. \tag{A.16}$$

Similar to the proof of Theorem 1 of Qian (2009) for a pair of nested Latin hypercube

designs based on nested permutations, we now show that the integral in (A.16) is nonpositive case by case.

Case 1: $v_{20} \geq \frac{1}{n_{\ell-1}}$

The integral equals

$$\frac{t_\ell}{t_{\ell-1}} v_{10} \frac{1}{n_{\ell-1}} - v_{10} \frac{1}{n_\ell} = 0.$$

Case 2: $0 \leq v_{20} < \frac{1}{n_{\ell-1}}$, $\lceil n_\ell v_{20} \rceil \neq \lceil n_\ell v_{10} \rceil$

The integral equals

$$\frac{t_\ell}{t_{\ell-1}} v_{10} v_{20} - v_{10} \frac{1}{n_\ell} = \frac{t_\ell}{t_{\ell-1}} v_{10} (v_{20} - \frac{1}{n_{\ell-1}}) \leq 0.$$

Case 3: $0 \leq v_{20} < \frac{1}{n_{\ell-1}}$, $\lceil n_\ell v_{20} \rceil = \lceil n_\ell v_{10} \rceil$

Let $p = \lfloor n_\ell v_{20} \rfloor = \lfloor n_\ell v_{10} \rfloor$, $r_1 = n_\ell v_{10} - p$, and $r_2 = n_\ell v_{20} - p$. Observe that

$$\int_0^{v_{10}} \int_0^{v_{20}} \delta_{n_{\ell-1}}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} = v_{10} v_{20} = \frac{(r_1 + p)(r_2 + p)}{n_\ell^2},$$

that

$$\int_0^{v_{10}} \int_0^{v_{20}} \delta_{n_\ell}(x_{i_1,k}, x_{i_2,k}) dx_{i_1,k} dx_{i_2,k} = \frac{p + r_1 r_2}{n_\ell^2},$$

and that $\frac{t_{\ell-1}}{t_\ell} \geq p + 1$. Then the integral under consideration is no greater than

$$-\frac{p(1 - r_1)(1 - r_2)}{(p + 1) n_\ell^2} \leq 0,$$

which completes the proof. □

Proof of Theorem 2.8

Proof. (i) Take $D_1 \subset \cdots \subset D_u$ from (2.7) with the ith run of D_u denoted by $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iq})$. By using an exchangeability argument, express $var_{SL}(\hat{\mu}_j)$ as

$$n_j^{-1} var\{f(\boldsymbol{x}_1)\} + n_j^{-2} \sum_{r=1}^{j} n_r(n_r - 1) cov_{SL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\}$$
$$+ n_j^{-2} \sum_{1 \le r < \ell \le j} 2(n_r - n_{r-1})(n_\ell - n_{\ell-1}) cov_{SL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{\ell-1}+1})\}.$$

It now suffices to show that all the covariances above are nonpositive. By Proposition 2.6, for $k = 1, \ldots, q$, both $(x_{n_{r-1}+1,k}, x_{n_{r-1}+2,k})$ and $(x_{n_{r-1}+1,k}, x_{n_{\ell-1}+1,k})$ are negatively quadrant dependent pairs. This result together with Theorem 1 of Lehmann (1966) and the monotonicity assumption on f imply the desired result. □

Proof of Theorem 2.9

Proof. (i) When $D_1, \ldots, D_u$ are taken under the IL scheme, E_r and E_ℓ are two independent ordinary Latin hypercube designs for $r < \ell$. Based on the same exchangeability argument as in the proof of Theorem 2.8, express $var_{IL}(\hat{\mu}_j)$

$$n_j^{-1} var\{f(\boldsymbol{x}_1)\} + n_j^{-2} \sum_{r=1}^{j} n_r(n_r - 1) cov_{IL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\}$$
$$+ n_j^{-2} \sum_{1 \le r < \ell \le j} 2(n_r - n_{r-1})(n_\ell - n_{\ell-1}) cov_{IL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{\ell-1}+1})\}.$$

As $cov_{SL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\} \le cov_{SL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\} = 0$ for $r < \ell$ by the fact that E_r and E_ℓ are independent, it suffices to show that E_r under the SL

scheme is an ordinary Latin hypercube design, such that

$$cov_{SL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\} = cov_{IL}\{f(\boldsymbol{x}_{n_{r-1}+1}), f(\boldsymbol{x}_{n_{r-1}+2})\}$$

for every $r = 1, \ldots, u$. When $s_2 = \ldots = s_u = 2$, E_1 is an ordinary Latin hypercube design and E_r is an $n_{r-1} \times q$ matrix for $r = 2, \ldots, u$. Consider $\boldsymbol{\pi}_u$ defined in Section 2.2 and $E_r = (e_{ik})$ in (2.6), the kth element in the ith row of E_r is

$$e_{ik} = \frac{\pi_{r,i} - u_{r,i}}{t_{r-1} n_{r-1}}, \quad i = 1, \ldots, n_{r-1},\ k = 1, \ldots, q,$$

where the $u_{r,i}$ are independent $U[0, 1)$ random variables, and the $\pi_{r,i}$ and the $u_{r,i}$ are mutually independent. With some abuse of nation, we use the notation $\boldsymbol{\pi}_u$ for each k while they are different independent permutations for different k.

Let $d_{r,i}$ denote $\lceil \pi_{r,i}/t_{r-1} \rceil$. By the definition of nested permutation $\boldsymbol{\pi}_u$,

$$\{\lceil \pi(1)/t_{r-1} \rceil, \ldots, \lceil \pi(n_{r-1})/t_{r-1} \rceil\}$$

is a permutation on $Z_{n_{r-1}}$ and $\{\lceil \pi(1)/t_{r-1} \rceil, \ldots, \lceil \pi(n_r)/t_{r-1} \rceil\}$ is a permutation on $\{Z_{n_{r-1}}, Z_{n_{r-1}}\}$. As a result, $\{d_{r,1}, \ldots, d_{r,n_{r-1}}\}$ is a permutation on $Z_{n_{r-1}}$. Let $\eta_{r,i} = t_{r-1} d_{r,i} - \pi_{r,i}$, then $e_{ik} = \frac{d_{r,i} - (\eta_{r,i} + u_{r,i})/t_{r-1}}{n_{r-1}}$. By Proposition 2.5(i), $\eta_{r,i}$ is a discrete uniform random variable on $\{0, 1, \ldots, t_{r-1} - 1\}$ and the $\eta_{r,i}$ and the $d_{r,i}$ are mutually independent. It is straightforward to show that the $(\eta_{r,i} + u_{r,i})/t_{r-1}$ are independent $U[0, 1)$ random variables, and the $(\eta_{r,i} + u_{r,i})/t_{r-1}$ and the $d_{r,i}$ are mutually independent. By (1.3), E_r is an ordinary Latin hypercube design with n_{r-1}

runs, which completes the proof.

(ii) Notice that the expected value of $f_k(x_k)$ is estimated only based on the kth column of the design. Hence, it is sufficient to show the result holds when f is a function with input $x \in (0,1]$. Clearly, IL and SL are equivalent in each dimension. We will use the same technique as in Theorem 3 of Stein (1987). Define $I(i,n) = ((i-1)/n, i/n]$. It can be shown that

$$\begin{aligned} var_{SL}(\hat{\mu}_j) = var_{LH}(\hat{\mu}_j) &= \frac{1}{n_j}\left\{\int_0^1 f(x)^2 dx - n_j \int_0^1 f(x_1)f(x_2)\delta_{n_j}(x_1,x_2)dx_1dx_2\right\} \\ &= \frac{1}{n_j}\int_0^1 f(x)^2 dx - \sum_{i=1}^{n_j}\left(\int_{I(i,n_j)} f(x)dx\right)^2. \end{aligned} \tag{A.17}$$

As $s_i = 2$ for $i = 2,\ldots,u$, the design at the j stage under the IL scheme consists of j ordinary Latin hypercube designs with size $n_j/2^{j-1}, n_j/2^{j-1}, n_j/2^{j-2}, \ldots, n_j/2$, respectively. Based on (A.17), we have

$$var_{IL}(\hat{\mu}_j) = \frac{1}{n_j}\int_0^1 f(x)^2 dx - n_j^{-2}\sum_{\ell=1}^{j}(n_j/2^{j-\ell+1})^2 \sum_{i=1}^{n_j/2^{j-\ell+1}}\left(\int_{I(i,n_j/2^{j-\ell+1})} f(x)dx\right)^2 \tag{A.18}$$

Notice that

$$I(i, n_j/2^{j-\ell+1}) = \cup_{k=(i-1)2^{j-\ell+1}+1}^{i2^{j-\ell+1}} I(k, n_j)$$

for any $\ell = 1,\ldots,j$, $i = 1,\ldots,n_j/2^{j-\ell+1}$. By Jensen's inequality, we have

$$\left(\int_{I(i,n_j/2^{j-\ell+1})} f(x)dx\right)^2 \le 2^{-(j-\ell+1)} \sum_{k=(i-1)2^{j-\ell+1}+1}^{i2^{j-\ell+1}}\left(\int_{I(k,n_j)} f(x)dx\right)^2.$$

Substituting the above inequality into (A.18) completes the proof. □

Appendix B

Proofs in Chapter 3

Proof of Proposition 3.5

Proof. In the proof of Lemma 4 in Tang (1993) the conditional probabilities in H_2 and H_3 are the same. Here H_2 and H_3 have different conditional probabilities because the columns of B are linked to a_{m+1} in the proposed method. As b_k are obtained based on a symbol permutation of a_{m+1}, $b_{jk} \neq b_{ik}$ is equivalent to $a_{j(m+1)} = a_{i(m+1)}$ while $b_{jk} = b_{ik}$ is equivalent to $a_{j(m+1)} \neq a_{i(m+1)}$ for any $i \neq j$. As A is an OA$(s^2, m+1, s, 2)$, among the remaining $n-1$ rows given the first row, $m(s-1)$ rows are in H_1 where $s-1$ rows with $a_{jk} = a_{1k}$ for each $k \in \mathcal{D}$. Similarly there are $s-1$ other entries in a_{m+1} taking the same value as $a_{1(m+1)}$, thus $s-1$ rows are in H_2. Finally, the remaining $(s-m)(s+1)$ rows are in H_3. As a result, $pr\{(i,j) \in H_1 \mid d_{jk}, k = 1, ...m\} = m/(s+1)$, $pr\{(i,j) \in H_2 \mid d_{jk}, k = 1, ...m\} = 1/(s+1)$, $pr\{(i,j) \in H_3 \mid d_{jk}, k = 1, ...m\} = (s-m)/(s+1)$. The result now follows by applying Table 3 and the definition of conditional probability. □

Proof of Theorem 3.6

Proof. (i) Note that $a_{i1}, \ldots, a_{im}, b_{i1}, \ldots, b_{im}$ and $\eta_{i1}, \ldots, \eta_{im}$ used in the generation of X_i are mutually independent. According to the construction of correlation-controlled U designs, a_{ik} and b_{ik} both are discrete uniform random variables on Z_s while η_{ik} is uniformly distributed on $[0, 1)$ such that X_i is uniformly distributed on $(0, 1]^m$ for $i = 1, \ldots, n$. As a result, $E(\hat{\mu}) = \mu = E\{f(X)\}$.

(ii) Without loss of generality, assume $\mu = 0$. Note that

$$\begin{aligned} var(\hat{\mu}) &= n^{-2}\left[\sum_{i=1}^{n} var\{f(X_i)\} + \sum_{i\neq j}^{n} cov\{f(X_i), f(X_j)\}\right] \\ &= n^{-1}var\{f(X)\} + (1 - n^{-1})cov\{f(X_1), f(X_2)\}, \end{aligned} \tag{B.1}$$

which holds as the runs X_i are exchangeable. Let u denote a subset of axes in $\mathcal{D}$, according to the functional analysis of variance decomposition defined in §1.2,

$$cov\{f(X_1), f(X_2)\} = \sum_{t=1}^{m}\sum_{|u|=t} E\{f_u(X_1^u)f_u(X_2^u)\}, \tag{B.2}$$

where X_i^u denotes a vector consisting of X_i^k for $k \in u$. More details can be found in Theorem 1 of Owen (1994b). For any $|u| = 1$, Proposition 3.4(ii), (1.3) and the continuity of f together imply that

$$E\{f_u(X_1^u)f_u(X_2^v)\} = \{-s^{-2} + o(s^{-2})\}var\{f_u(X^u)\}. \tag{B.3}$$

For any $|u| \geq 2$, we have

$$cov\{f_u(X_1^u), f_u(X_2^u)\} = E[E\{f_u(X_1^u) \mid d_{1k}, k \in u\}E\{f_u(X_2^u) \mid d_{1k}, k \in u\}]. \quad \text{(B.4)}$$

By Proposition 3.5,

$$\begin{aligned} E\{f_u(X_2^u) \mid d_{1k}, k \in u\} =& \frac{(s-1)^{|u|} + |u|(s-1)^{|u|-1} + 1}{(s+1)(s-1)^{|u|-1}} \int_{x \in H_1} f_u(x^u)dw \\ &+ \frac{(s-1)^{|u|} + |u|(s-1)^{|u|-1} + 1}{s+1} \int_{x \in H_2} f_u(x^u)dw \\ &+ \frac{(s-|u|)\{(s-1)^{|u|} + |u|(s-1)^{|u|-1} + 1\}}{(s+1)(s-1)^{|u|}} \int_{x \in H_3} f_u(x^u)dw. \end{aligned}$$

As $s \to \infty$, the set H_4^C becomes the unit cube $(0,1]^{|u|}$. Thus, w has density $1 + o(1)$ with respect to the Lebesgue measure on H_4^C. Since $\int f_u(x^u)dF_v = 0$ for any $v \in u$,

$$\begin{aligned} E\{f_u(X_2^u) \mid d_{1k}, k \in u\} =& - s^{|u|} \sum_{k,l \in u, k<l} E\{f_u(X_1^u) \mid a_{1k}, a_{1l}\} + o(s^{-2}) \\ &+ \{s^{-1} + o(s^{-1})\}E\{f_u(X_2^u) \mid d_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\}. \end{aligned} \quad \text{(B.5)}$$

The last term in (B.5) does not appear in the proof of Theorem 1 of U designs in Tang (1993). This term is due to the special permutations for the auxiliary array B in step 2 of the construction of correlation-controlled U designs. To complete the proof, it suffices to show that $E[E\{f_u(X_1^u) \mid d_{1k}, k \in u\}E\{f_u(X_2^u) \mid d_{1k}, a_{2k} \neq$

$a_{1k}, b_{2k} = b_{1k}, k \in u\}] = o(s^{-1})$. Conditioning gives

$$\begin{aligned}
&E[E\{f_u(X_1^u) \mid d_{1k}, k \in u\}E\{f_u(X_2^u) \mid d_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\}] \\
=&E[f_u(X_1^u)E\{f_u(X_2^u) \mid d_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\}] \\
=&E\left(E[f_u(X_1^u)E\{f_u(X_2^u) \mid b_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\} \mid a_{1k}, k \in u]\right). \qquad \text{(B.6)}
\end{aligned}$$

Given a_{1k} for $k \in u$, simplify the inner conditional expectation in (B.6) as

$$\begin{aligned}
&E[f_u(X_1^u)E\{f_u(X_2^u) \mid b_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\} \mid a_{1k}, k \in u] \\
=&s^{-|u|} \sum_{b_{1k} \in Z_s, k \in u} E\{f_u(X_1^u) \mid a_{1k}, b_{1k}, k \in u\}E\{f_u(X_2^u) \mid a_{1k}, b_{1k}, a_{2k} \neq a_{1k}, b_{2k} = b_{1k}, k \in u\} \\
=&s^{-|u|} \sum_{b_{1k} \in Z_s, k \in u} [E\{f_u(X_1^u) \mid a_{1k}, k \in u\} + O(s^{-1})][E\{f_u(X_2^u) \mid a_{1k}, a_{2k} \neq a_{1k}, k \in u\} \\
&+ O(s^{-1})] \\
=&E\{f_u(X_1^u) \mid a_{1k}, k \in u\}E\{f_u(X_2^u) \mid a_{1k}, a_{2k} \neq a_{1k}, k \in u\} + o(s^{-1}) \qquad \text{(B.7)} \\
=&o(s^{-1})[E\{f_u(X_1^u) \mid a_{1k}, k \in u\}]^2 + o(s^{-1}) \qquad \text{(B.8)} \\
=&o(s^{-1}), \qquad \text{(B.9)}
\end{aligned}$$

where the Lipschitz continuity of f establishes (B.7), and (B.8) holds since

$$E\{f_u(X_2^u) \mid a_{1k}, a_{2k} \neq a_{1k}, k \in u\} = \{-(s-1)\}^{-|u|}E\{f_u(X_1^u) \mid a_{1k}, k \in u\},$$

using the fact that $\int f_u(x^u)dF_k = 0$ for any $k \in u$.

Substituting (B.5), (B.6) and (B.9) into (B.4), we have that for any $u \subset D$ with

$|u| \geq 2$,

$$\begin{aligned} & E[E\{f_u(X_1^u) \mid d_{1k}, k \in u\} E\{f_u(X_2^u) \mid d_{1k}, k \in u\}] \\ =& - s^{|u|} \sum_{k,l \in u, k<l} var[E\{f_u(X_1^u) \mid a_{1k}, a_{1l}\}] + o(s^{-2}). \end{aligned} \tag{B.10}$$

Substituting (B.3) and (B.10) into (B.2),

$$\begin{aligned} & cov\{f(X_1), f(X_2)\} \\ =& - s^{-2} \sum_{k=1}^{m} var\{f_{\{k\}}(X^k)\} - s^{-2} \sum_{|u|=2} \sum_{k,l \in u, k<l} var[E\{f_u(X_1^u) \mid a_{1k}, a_{1l}\}] + o(s^{-2}) \\ =& - s^{-2} \sum_{k=1}^{m} var\{f_{\{k\}}(X^k)\} - s^{-2} \sum_{k<l}^{m} var[E\{f(X_1) \mid a_{1k}, a_{1l}\} - E\{f(X_1) \mid a_{1k}\} \\ & - E\{f(X_1) \mid a_{1l}\}] + o(s^{-2}) \\ =& - s^{-2} \sum_{k=1}^{m} var\{f_{\{k\}}(X^k)\} - s^{-2} \sum_{k<l}^{m} var\{f_{\{k,l\}}(X^k, X^l)\} + o(s^{-2}). \end{aligned} \tag{B.11}$$

Here, (B.11) holds due to the continuity of f. By substituting (B.11) in (B.1), Theorem 3.6 follows as $n = s^2$. □

Proof of Proposition 3.7

Proof. (i) As a_k and a_l are orthogonal, a_{ik} and a_{jk} are independent and both have s choices on Z_s with equal probabilities given $a_{il} \neq a_{jl}$ for some $l = 1, \ldots, m$ and $l \neq k$.

(ii) As a_k and a_l are orthogonal, a_{ik} and a_{jk} must be different given $a_{il} \neq a_{jl}$ for some $l = 1, \ldots, m$ and $l \neq k$. There are $s(s-1)$ possible level combinations for a_{ik}

and a_{jk} which are equally likely.

(iii) By step 2 in the construction of U designs, the b_{ik} and b_{jk} are chosen such that $b_{ik} \neq b_{jk}$ if $a_{ik} = a_{jk}$. Hence, b_k and a_k are orthogonal. The result follows (i) by changing a_{ik}, a_{jk}, a_{il} and a_{jl} into b_{ik}, b_{jk}, a_{ik} and a_{jk}, respectively.

(iv) Using the same argument in (iii), the result follows (ii). □

Proof of Lemma 3.8

Proof. (i) For any $k \neq l$,

$$var(\sum_{i=1}^{n} \bar{a}_{ik}\bar{b}_{il}) = \sum_{i \neq j}^{n} E(\bar{a}_{ik}\bar{a}_{jk}\bar{b}_{il}\bar{b}_{jl}) + \sum_{i=1}^{n} E(\bar{a}_{ik}^2\bar{b}_{il}^2), \tag{B.12}$$

where $\bar{a}_{ik} = (a_{ik} - 1/2)/s - 1/2$ and $\bar{b}_{ik} = (b_{ik} - 1/2)/s - 1/2$.

By Proposition 5(i)-(iv), for any $i \neq j$, we have

$$\begin{aligned} E(\bar{a}_{ik}\bar{a}_{jk}\bar{b}_{il}\bar{b}_{jl}) =& E(\bar{a}_{ik}\bar{a}_{jk} \mid a_{il} = a_{jl})E(\bar{b}_{il}\bar{b}_{jl} \mid a_{il} = a_{jl})pr(a_{il} = a_{jl}) \\ &+ E(\bar{a}_{ik}\bar{a}_{jk} \mid a_{il} \neq a_{jl})E(\bar{b}_{il}\bar{b}_{jl} \mid a_{il} \neq a_{jl})pr(a_{il} \neq a_{jl}) \\ =& n^{-2}(s+1)/144, \end{aligned} \tag{B.13}$$

where $pr(a_{il} \neq a_{jl}) = 1/(s+1)$.

As a_{ik} and b_{il} are two independent discrete random variables uniformly distributed on Z_s,

$$E(\bar{a}_{ik}^2\bar{b}_{il}^2) = \{E(\bar{a}_{ik}^2)\}^2 = n^{-2}(n-1)^2/144. \tag{B.14}$$

Substituting (B.13) and (B.14) to (B.12) completes the proof.

(ii) Similarly, for any $k \neq l$,

$$var(\sum_{i=1}^{n} \bar{b}_{ik}\bar{b}_{il}) = \sum_{i \neq j}^{n} E(\bar{b}_{ik}\bar{b}_{jk}\bar{b}_{il}\bar{b}_{jl}) + \sum_{i=1}^{n} E(\bar{b}_{ik}^2\bar{b}_{il}^2), \tag{B.15}$$

By Proposition 5(iii) and (iv), for any $i \neq j$

$$\begin{aligned} E(\bar{b}_{ik}\bar{b}_{jk}\bar{b}_{il}\bar{b}_{jl}) =& E(\bar{b}_{ik}\bar{b}_{jk} \mid a_{ik} \neq a_{jk})E(\bar{b}_{il}\bar{b}_{jl} \mid a_{il} \neq a_{jl})pr(a_{ik} \neq a_{jk}, a_{il} \neq a_{jl}) \\ &+ E(\bar{b}_{ik}\bar{b}_{jk} \mid a_{ik} = a_{jk})E(\bar{b}_{il}\bar{b}_{jl} \mid a_{il} \neq a_{jl})pr(a_{ik} = a_{jk}, a_{il} \neq a_{jl}) \\ &+ E(\bar{b}_{ik}\bar{b}_{jk} \mid a_{ik} \neq a_{jk})E(\bar{b}_{il}\bar{b}_{jl} \mid a_{il} = a_{jl})pr(a_{ik} \neq a_{jk}, a_{il} = a_{jl}) \\ =& 0. \end{aligned} \tag{B.16}$$

As b_{ik} and b_{il} are two independent discrete random variables uniformly distributed on Z_s,

$$E(\bar{b}_{ik}^2\bar{b}_{il}^2) = n^{-2}(n-1)^2/144. \tag{B.17}$$

Substituting (B.16) and (B.17) to (B.15) completes the proof. □

Proof of Theorem 3.9

Proof. It suffices to show that $var(\rho_{kl}) = n^{-2}/72 + O(n^{-5/2})$ in part (i) and $var = n^{-5/2}/144 + O(n^{-3})$ in part (ii) for any $1 \leq k < l \leq m$.

(i) By (1.13), (3.2) and Proposition 3.2(i),

$$\begin{aligned} var(\rho_{kl}) =& var\left[n^{-1}\sum_{i=1}^{n}\left\{\bar{a}_{ik}+s^{-1}\bar{b}_{ik}-n^{-1}\left(\eta_{ik}-1/2\right)\right\}\left\{\bar{a}_{il}+s^{-1}\bar{b}_{il}-n^{-1}\left(\eta_{il}-1/2\right)\right\}\right] \\ =& 2n^{-3}var\left(\sum_{i=1}^{n}\bar{a}_{ik}\bar{b}_{il}\right)+n^{-4}var\left(\sum_{i=1}^{n}\bar{b}_{ik}\bar{b}_{il}\right)+2n^{-4}var\left\{\sum_{i=1}^{n}\bar{a}_{ik}(\eta_{il}-1/2)\right\} \\ &+2n^{-5}var\left\{\sum_{i=1}^{n}\bar{b}_{ik}(\eta_{il}-1/2)\right\}+n^{-6}var\left\{\sum_{i=1}^{n}\left(\eta_{ik}-1/2\right)\left(\eta_{il}-1/2\right)\right\}. \end{aligned} \tag{B.18}$$

The last three variance components in (B.18) are all $O(n)$. The result now follows by Lemma 3.8.

(ii) By (1.13), (3.2) and Proposition 3.2(ii),

$$\begin{aligned} var(\rho_{kl}) =& n^{-2}var\left(n^{-1}\sum_{i=1}^{n}\bar{b}_{ik}\bar{b}_{il}\right)+2n^{-4}var\left\{\sum_{i=1}^{n}\bar{a}_{ik}(\eta_{il}-1/2)\right\} \\ &+2n^{-5}var\left\{\sum_{i=1}^{n}\bar{b}_{ik}(\eta_{il}-1/2)\right\}+n^{-6}var\left\{\sum_{i=1}^{n}\left(\eta_{ik}-1/2\right)\left(\eta_{il}-1/2\right)\right\}. \end{aligned} \tag{B.19}$$

Since all the b_k are selected based on a permutation of a_{m+1} in correlation-controlled U designs, the concatenation of b_k and b_l in correlation-controlled U designs consists of s copies of a Latin hypercube with s levels. As a result, the first variance in (B.19) equals $n^{-2}(s+1)(n-1)/144$ by Theorem 1 of Owen (1994b), which completes the proof.

□

Appendix C

Proofs in Chapter 4

Proof of Proposition 4.1

Proof. (i) According to (4.8) and Step 2 in the above construction, B_r is the same as A_r in Step 1 *prior to* the replacement step, and the result follows.

(ii) Note that $b_{r,ik} = \left\lceil \frac{a_{r,ik}-\gamma_{r,ik}}{t} \right\rceil = \left\lceil \frac{a_{r,ik}}{t} \right\rceil$ where the $a_{r,ik}$ are values in A_r after the replacement in Step 2 of the construction above. By (4.7) and (4.8), we have

$$\theta_{r,ik} = b_{r,ik} - \frac{a_{r,ik}}{t} + \frac{\gamma_{r,ik}}{t} = \left(t \left\lceil \frac{a_{r,ik}}{t} \right\rceil - a_{r,ik} \right) /t + \gamma_{r,ik}/t. \tag{C.1}$$

According to Step 2 of the construction, for each $r = 1, 2, \ldots, t$, $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$, the quantity $t\lceil \frac{a_{r,ik}}{t} \rceil - a_{r,ik}$ is randomly selected among Z_t and is independent of B_r. Since the $\gamma_{r,ik}$ are independent of the $a_{r,ik}$, the claim is proved.

(iii) For each $r = 1, \ldots, t$ and $k = 1, \ldots, m$, the quantities $\{t \left\lceil \frac{a_{r,1k}}{t} \right\rceil - a_{r,1k}, t \left\lceil \frac{a_{r,2k}}{t} \right\rceil -$

$a_{r,2k}, \ldots, t\left\lceil \frac{a_{r,nk}}{t} \right\rceil - a_{r,nk}\}$ are independently and randomly selected among n different Z_t's, respectively. Thus, the $\theta_{r,ik}$ are mutually independent within Θ_r. In other words, the $t\lceil \frac{a_{r,ik}}{t} \rceil - a_{r,ik}$, $i = 1, 2, \ldots, n$ are mutually independent discrete uniform random variables on Z_t, such that the $\theta_{r,ik}$ are $U[0, 1)$ random variables, by (C.1).

(iv) It suffices to show that $t\left\lceil \frac{a_{r,1k}}{t} \right\rceil - a_{r,1k}$ and $t\left\lceil \frac{a_{r,nk}}{t} \right\rceil - a_{s,nk}$ are dependent if and only if $B_{r,ik} = B_{s,ik}$. That is true because $t\left\lceil \frac{a_{r,1k}}{t} \right\rceil - a_{r,1k}$ and $t\left\lceil \frac{a_{r,nk}}{t} \right\rceil - a_{s,nk}$ are selected from the same Z_t when $B_{r,ik} = B_{s,ik}$.

(v) The result follows directly from (i), (ii), (iv), and the definition of the ordinary Latin hypercube design.

□

Proof of Lemma 4.5

Proof. Given two ordinary Latin hypercubes B_r and B_s in (4.8), let $D_r = (B_r - \Theta_r)/n$ and $D_s = (B_s - \Theta_s)/n$ denote two slices generated by (4.7). Since the underlying Latin hypercubes are fixed for D_r and D_s, the only random parts in the definition are Θ_r and Θ_s. Define $H(\Delta_r; B_r) = v_n(D_r) = v_n(B_r/n - \Theta_r/n)$ and $H(\Delta_s; B_s) = v_n(D_s) = v_n(B_s/n - \Theta_s/n)$ as in Definition 4.3, where Δ_r and Δ_s are $n \times m$ matrices with the (k, ℓ)th entry defined as $\delta_{r,\ell k} = \theta_{r,ik}$ and $\delta_{s,\ell k} = \theta_{s,ik}$ such that $B_{r,ik} = B_{s,ik} = \ell$ for $\ell = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$. By Proposition 4.1 (iii) and (iv), we can find nm independent pairs of random variables: (δ_r, δ_s) for $\ell = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$. By (Lehmann, 1966, Theorem 2), the monotonicity assumption of

$v_n(D)$, and the proof of (Qian, 2012, Theorem 1), we have

$$\mathbb{E}\left[H(\Delta_r; B_r)H(\Delta_s; B_s)\right] \leq \mathbb{E}\left[H(\Delta_r; B_r)\right] \mathbb{E}\left[H(\Delta_r; B_r)\right],$$

which is equivalent to

$$\mathbb{E}\left[v_n(D_r)v_n(D_s)|B_r, B_s\right] \leq \mathbb{E}\left[v_n(D_r)|B_r\right] \mathbb{E}\left[v_n(D_r)|B_s\right].$$

Taking expectations of both sides gives

$$\mathbb{E}\left[v_n(D_r)v_n(D_s)\right] \leq \mathbb{E}\left\{\mathbb{E}\left[v_n(D_r)|B_r\right]\right\} \mathbb{E}\left\{\mathbb{E}\left[v_n(D_s)|B_s\right]\right\} = \mathbb{E}\left[v_n(D_r)\right] \mathbb{E}\left[v_n(D_s)\right].$$

The last equality holds because B_r and B_s are independent, by Proposition 4.1 (i). □

Proof of Theorem 4.7

Proof. (i) When $D_1, D_2, \ldots, D_t$ are independent under ILH, $cov[v_n(D_r), v_n(D_s)] = 0$ for any $r \neq s$. Thus, we have $var(L_{n,t}^{ILH}) = t^{-1}var[v_n(D)]$.

(ii) When $D_1, D_2, \ldots, D_t$ are sampled under SLH, we have

$$\begin{aligned} var(L_{n,t}^{SLH}) =& var\left[t^{-1}\sum_{i=1}^{t} v_n(D_i)\right] \\ =& t^{-2}\sum_{i=1}^{t} var\left[v_n(D_i)\right] + t^{-2}\sum_{1\leq r<s\leq t} cov\left[v_n(D_r)v_n(D_s)\right] \\ =& var(L_{n,t}^{ILH}) + (1-t^{-1})cov\left[v_n(D_1)v_n(D_2)\right], \end{aligned} \tag{C.2}$$

where the last equality in (C.2) holds because batches are exchangeable. Let $H(\Delta_1; B_1) = v_n(D_1)$ and $H(\Delta_2; B_2) = v_n(D_2)$. Without loss of generality, assume $\mathbb{E}[v_n(D)] = 0$. We have

$$\begin{aligned}
cov[v_n(D_1)v_n(D_2)] =& \mathbb{E}\left[v_n(D_1)v_n(D_2)\right] \\
=& \mathbb{E}\left\{\mathbb{E}\left[v_n(D_1)v_n(D_2)|B_1, B_2\right]\right\} \\
=& \mathbb{E}\left\{\mathbb{E}\left[H(\Delta_1; B_1)H(\Delta_2; B_2)\right]\right\} \\
=& \mathbb{E}\left[\sum_{\ell=1}^{n}\sum_{k=1}^{m} -t^{-1}\int H_{\{\ell k\}}(\delta_{\ell k}; B_1)H_{\{\ell k\}}(\delta_{\ell k}; B_2)d\delta_{\ell k}\right] + o(t^{-1}) \\
=& -t^{-1}\sum_{\ell=1}^{n}\sum_{k=1}^{m}\int\left\{\mathbb{E}\left[H_{\{\ell k\}}(\delta_{\ell k}; B)\right]\right\}^2 d\delta_{\ell k} + o(t^{-1}), \qquad \text{(C.3)}
\end{aligned}$$

where the second-last equality is from (Qian, 2009, Lemma 1). The result follows by substituting (C.3) into (C.2). The functional ANOVA decomposition is properly defined because the quantities $\delta_{\ell k}$ are independent, according to Proposition 4.1 (iii).

(iii) Assuming again that $\mathbb{E}[v_n(D)] = 0$, we have

$$\begin{aligned}
var[v_n(D)] =& \mathbb{E}\left\{var\left[v_n(D)|B\right]\right\} + var\left\{\mathbb{E}\left[v_n(D)|B\right]\right\} \\
=& \mathbb{E}\left\{var\left[H(\Delta;B)\right]\right\} \\
=& \mathbb{E}\left\{\sum_{\ell=1}^{n}\sum_{k=1}^{m}\int \left[H_{\{\ell k\}}(\delta_{\ell k};B)\right]^2 d\delta_{\ell k}\right\} \\
=& \sum_{\ell=1}^{n}\sum_{k=1}^{m}\int \mathbb{E}\left\{\left[H_{\{\ell k\}}(\delta_{\ell k};B)\right]^2\right\} d\delta_{\ell k} \\
=& \sum_{\ell=1}^{n}\sum_{k=1}^{m}\int var\left[H_{\{\ell k\}}(\delta_{\ell k};B)\right] + \left\{\mathbb{E}\left[H_{\{\ell k\}}(\delta_{\ell k};B)\right]\right\}^2 d\delta_{\ell k} \\
=& \sum_{\ell=1}^{n}\sum_{k=1}^{m}\int \left\{\mathbb{E}\left[H_{\{\ell k\}}(\delta_{\ell k};B)\right]\right\}^2 d\delta_{\ell k}. \qquad \text{(C.4)}
\end{aligned}$$

The second equality holds because $\mathbb{E}[v_n(D)|B]$ is the same regardless of the underlying ordinary Latin hypercube B when $v_n(D)$ is additive. The third equation holds due to the functional ANOVA decomposition on $H(\Delta;B)$. We complete the proof by combining (C.4) with the result in (ii). □

Proof of Lemma 4.8

Proof. For any ordinary Latin hypercube design D, let x_n be the arg min of the approximation problem, that is,

$$v_n(D) = \min_{x\in X}\frac{1}{n}\sum_{i=1}^{n} G(x,\xi_i) = \frac{1}{n}\sum_{i=1}^{n} G(x_n,\xi_i),$$

where ξ_i is the ith scenario of D. Without loss of generality, increase only ξ_{1k} to obtain a new design D^* with scenarios $\xi_1^*, \xi_2, \ldots, \xi_n$. Let x_n^* be the arg min of the

approximation problem with design D^*, that is,

$$v_n(D^*) = \min_{x \in X} \frac{1}{n} \left[G(x, \xi_1^*) + \sum_{i=2}^{n} G(x, \xi_i) \right] = \frac{1}{n} \left[G(x_n^*, \xi_1^*) + \sum_{i=2}^{n} G(x_n^*, \xi_i) \right].$$

If either (i) or (ii) is satisfied, the value of x does not affect whether $G(\cdot, \xi_1)$ is increasing or decreasing in ξ_{1k}. Supposing that $G(\cdot, \xi_1)$ is increasing in ξ_{1k}, we have

$$v_n(D^*) = \frac{1}{n} \left[G(x_n^*, \xi_1^*) + \sum_{i=2}^{n} G(x_n^*, \xi_i) \right] \geq \frac{1}{n} \sum_{i=1}^{n} G(x_n^*, \xi_i) \geq v_n(D),$$

which implies that $v_n(D)$ is increasing in ξ_{1k}. Similarly, if $f(\cdot, \xi_1)$ is decreasing in ξ_{1k}, then

$$v_n(D^*) \leq \frac{1}{n} \left[G(x_n, \xi_1^*) + \sum_{i=2}^{n} G(x_n, \xi_i) \right] \leq \frac{1}{n} \sum_{i=1}^{n} G(x_n, \xi_i) = v_n(D),$$

which implies $v_n(D)$ is decreasing in ξ_{1k}. □

Proof of Proposition 4.11

Proof. Let $x_n^*(D_r)$ denote an optimal solution to the SAA problem with scenarios given by D_r for $r = 1, 2, \ldots, t$, we have

$$L_{n,t} = t^{-1} \sum_{r=1}^{t} v_n(D_r) = (nt)^{-1} \sum_{r=1}^{t} \sum_{i=1}^{n} \tilde{G}(x_n^*(D_r), \tilde{\xi}_{r,i}).$$

Because $x_n^*(D_r) = x^*$ w.p.1 for n sufficiently large by Proposition 4.10 and $\tilde{G}(\cdot, \tilde{\xi})$ is continuous, the result follows according to the Continuous Convergence Theorem (Mann and Wald, 1943). □

Proof of Theorem 4.12

Proof. (i) Since $D_1, D_2, \ldots, D_t$ are independent ordinary Latin hypercube designs sampled under ILH, we immediately have that $var(L_{n,t}^{ILH}) = t^{-1}var\left[v_n(D_1)\right]$, by exchangeability of batches. By (Stein, 1987, Corollary 1), we have

$$var\left[v_n(D_1)\right] = n^{-1}var\left[\tilde{G}(x^*, \tilde{\xi})\right] - n^{-1}\sum_{k=1}^{m} var\left[\tilde{G}_{\{k\}}(x^*, \tilde{\xi}^k)\right] + o(n^{-1}),$$

and the result follows.

(ii) The result holds as a consequence of (Qian, 2012, Theorem 2).

(iii) Since any batch sampled under SLH is statistically equivalent to an ordinary Latin hypercube design, by Proposition 4.1 (v), the variances $var\left[v_n(D_i)\right]$ are the same as those under ILH. Due to exchangeability of batches and scenarios within the same batch, it suffices to show that under SLH, we have $cov\left[\tilde{G}(x^*, \tilde{\xi}_{1,1}), \tilde{G}(x^*, \tilde{\xi}_{2,1})\right] \leq 0$, where $\tilde{\xi}_{r,i}$ denote the ith scenario in D_r. For $0 < z_1, z_2 \leq 1$ and an integer p, define

$$\alpha_p(z_1, z_2) = \begin{cases} 1, & \lceil pz_1 \rceil = \lceil pz_1 \rceil \\ 0, & \text{o.w.} \end{cases}$$

By (Qian, 2012, Lemma 1 (iii)), the joint probability density function of $\tilde{\xi}_{1,1}$ and $\tilde{\xi}_{2,1}$ is

$$\left(\frac{t}{t-1}\right)^m \prod_{k=1}^{m}\left[1 - t^{-1} + t^{-1}\alpha_n(\tilde{\xi}_{1,1k}, \tilde{\xi}_{2,1k}) - \alpha_{nt}(\tilde{\xi}_{1,1k}, \tilde{\xi}_{2,1k})\right]. \tag{C.5}$$

Let $I(i,n)$ denote the interval $((i-1)/n, i/n]$. Without loss of generality, assume that $\mathbb{E}[\tilde{G}(x^*, \tilde{\xi})] = 0$, where $\tilde{G}(x^*, \tilde{\xi}) = \sum_{k=1}^{m} \tilde{G}_{\{k\}}(x^*, \tilde{\xi}^k)$. Following the proof of

(Stein, 1987, Theorem 1), we can express $cov\left[\tilde{G}(x^*,\tilde{\xi}_{1,1}),\tilde{G}(x^*,\tilde{\xi}_{2,1})\right]$ as

$$\begin{aligned}
&cov\left[\tilde{G}(x^*,\tilde{\xi}_{1,1}),\tilde{G}(x^*,\tilde{\xi}_{2,1})\right]\\
=&\frac{t}{t-1}\sum_{k=1}^{m}\left(\frac{1}{t}\int\tilde{G}_{\{k\}}(x^*,\tilde{\xi}_{1,1k})\tilde{G}_{\{k\}}(x^*,\tilde{\xi}_{2,1k})\alpha_n(\tilde{\xi}_{1,1k},\tilde{\xi}_{2,1k})d\tilde{\xi}_{1,1k}d\tilde{\xi}_{2,1k}\right.\\
&\left.-\int\tilde{G}_{\{k\}}(x^*,\tilde{\xi}_{1,1k})\tilde{G}_{\{k\}}(x^*,\tilde{\xi}_{2,1k})\alpha_{nt}(\tilde{\xi}_{1,1k},\tilde{\xi}_{2,1k})d\tilde{\xi}_{1,1k}d\tilde{\xi}_{2,1k}\right)\\
=&\frac{t}{t-1}\sum_{k=1}^{m}\left[\frac{1}{t}\sum_{i=1}^{n}\left(\int_{I(i,n)}\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)d\tilde{\xi}^k\right)^2\right.\\
&\left.-\sum_{i=1}^{nt}\left(\int_{I(i,nt)}\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)d\tilde{\xi}^k\right)^2\right] \qquad \text{(C.6)}
\end{aligned}$$

Notice that $I(i,n)=\cup_{j=(i-1)t+1}^{it}I(j,nt)$ for any $i=1,2,\ldots,n$. By Jensen's inequality, for each $k=1,2,\ldots,m$ and $i=1,2,\ldots,n$, we have

$$\frac{1}{t}\left(\int_{I(i,n)}\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)d\tilde{\xi}^k\right)^2\leq\sum_{j=(i-1)t+1}^{it}\left(\int_{I(j,nt)}\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)d\tilde{\xi}^k\right)^2.$$

The proof of the first claim is completed by substituting this inequality into (C.6).

To prove the second claim, we define

$$\zeta_i^n(\tilde{\xi}^k)=\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)-\int_{I(i,n)}\tilde{G}_{\{k\}}(x^*,\tilde{\xi}^k)d\tilde{\xi}^k.$$

Again, by following the proof of (Stein, 1987, Theorem 1), we have

$$var[v_n(D_1)]=\frac{1}{n}\sum_{k=1}^{m}\sum_{i=1}^{n}\left[\zeta_i^n(\tilde{\xi}^k)\right]^2.$$

Because each Ξ_k is finite, the number of non-zero values of $\zeta_i^n(\tilde{\xi}^k)$ is also finite. Hence, $var(L_{n,t}^{ILH}) = t^{-1}var[v_n(D_1)] = O(n^{-2}t^{-1})$. Under SLH, using the same arguments as (C.6), we have

$$cov\left[\tilde{G}(x^*, \tilde{\xi}_{1,1}), \tilde{G}(x^*, \tilde{\xi}_{2,1})\right] = \frac{t}{t-1}\sum_{k=1}^{m}\left[\frac{1}{nt}\sum_{i=1}^{nt}(\zeta_i^{nt}(\tilde{\xi}^k))^2 - \frac{1}{nt}\sum_{i=1}^{n}(\zeta_i^{n}(\tilde{\xi}^k))^2\right],$$

and

$$\begin{aligned}
var(L_{n,t}^{SLH}) =& t^{-1}var\left[v_n(D_1)\right] + \frac{t-1}{t}cov\left[\tilde{G}(x^*, \tilde{\xi}_{1,1}), \tilde{G}(x^*, \tilde{\xi}_{2,1})\right] \\
=& \frac{1}{nt}\sum_{k=1}^{m}\sum_{i=1}^{n}(\zeta_i^n(\tilde{\xi}^k))^2 + \sum_{k=1}^{m}\left[\frac{1}{nt}\sum_{i=1}^{nt}(\zeta_i^{nt}(\tilde{\xi}^k))^2 - \frac{1}{nt}\sum_{i=1}^{n}(\zeta_i^{n}(\tilde{\xi}^k))^2\right] \\
=& \frac{1}{nt}\sum_{k=1}^{m}\sum_{i=1}^{nt}\left(\zeta_i^{nt}(\tilde{\xi}^k)\right)^2 \\
=& O(n^{-2}t^{-2}).
\end{aligned}$$

□

Proof of Proposition 4.13

Proof. Divide $(0, 1]$ into the n equal subintervals as $(0, 1/n], (2/n, 3/n], \ldots, ((n-1)/n, 1]$, and let $I(i, n) = ((i-1)/n, i/n]$, as before. Assume $p_{(0)k} = 0$. For any $k = 1, 2, \ldots, m$, when $np_{(1)k}$ is an integer, we have $\xi^k = F_k^{-1}(\tilde{\xi}^k)$ equals the smallest possible value in Ξ_k, provided that $\tilde{\xi}^k$ is drawn from $I(1, n), I(2, n), \ldots, I(np_{(1)k}, n)$. Similarly, when $np_{(2)k}$ is an integer, we have that $\xi^k = F_k^{-1}(\tilde{\xi}^k)$ equals the second smallest value in Ξ_k, provided that $\tilde{\xi}^k$ is taken from $I(np_{(1)k} + 1, n), I(np_{(1)k} + 2, n), \ldots, I(np_{(2)k}, n)$, and so on. As a result, $\xi_1, \xi_2, \ldots, \xi_n$ is determined only by B.

That is, given b_{ik}, $\tilde{\xi}_{ik}$ is taken from $I(b_{ik}, n)$ and $\xi_{ik} = F_k^{-1}(\tilde{\xi}_{ik})$ is the ℓth smallest value in Ξ_k, where ℓ is an integer and $np_{(\ell-1)k} + 1 \leq b_{ik} \leq n_{(\ell)k}$. Under SLH, $v_n(D_r)$ and $v_n(D_s)$ would be independent for any $r \neq s$, because they depend on B_r and B_s, which are independent according to Proposition 4.1 (i). □

Proof of Theorem 4.14

Proof. (Hwang et al., 2013, Theorem 1) proves this result for continuous $\tilde{G}(x^*, \cdot)$. We extend it to cases in which $\tilde{G}(x^*, \cdot)$ is a step function (since ξ has finite support) by applying (Loh, 1996, Lemma 3). □

References

Adams, B. M., M. S. Ebeida, M. S. Eldred, J. D. Jakeman, and L. P. Swiler. 2014. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Version 5.4 User's Manual, Sandia National Labs., Albuquerque, MN (US) Sandia National Labs., Livermore, CA (US).

Addelman, S., and O. Kempthorne. 1961. Some main-effect plans and orthogonal arrays of strength two. *Ann. Math. Statist.* 32:1167–1176.

Bayraksan, G., and D. P. Morton. 2006. Assessing solution quality in stochastic programs. *Mathematical Programming* 108(2-3):495–514.

Bingham, D., R. R. Sitter, and B. Tang. 2009. Orthogonal and nearly orthogonal designs for computer experiments. *Biometrika* 96:51–65.

Bose, R. C., and K. A. Bush. 1952. Orthogonal arrays of strength two and three. *Ann. Math. Statist.* 23:508–523.

Bush, K. A. 1952. Orthogonal arrays of index unity. *Ann. Math. Statist.* 23:426–34.

Butler, N. A. 2001. Optimal and orthogonal Latin hypercube designs for computer experiments. *Biometrika* 88:847–857.

Chen, J., and P. Z. G. Qian. 2014. Latin hypercube designs with controlled correlations and multi-dimensional stratification. *Biometrika* 101:319–332.

Czyzyk, J., J. T. Linderoth, and J. Shen. 2000. Sutil: A utility library for handling stochastic programs. http://coral.ie.lehigh.edu/ sutil/.

Dantzig, G. B. 1963. *Linear programming and extensions.* Princeton University Press, Princeton. New Jersey.

Donnelly, T. 2010. Comparison of prediction accuracy of surrogate models developed using nested Latin hypercube designs. SAS Institute Inc., NDIA 26th Annual – National Test & Evaluation Conference.

Drew, S. S., and T. Homem-de Mello. 2005. *Some large-deviations results for latin hypercube sampling.* In WSC05: Proc. 37th Conf. Winter Simul, pp. 673–81. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Fang, K. F., R. Z. Li, and A. Sudjianto. 2005. *Design and modeling for computer experiments.* New York: Chapman & Hall/CRC Press.

Freimer, M. B., J. T. Linderoth, and D. J. Thomas. 2012. The impact of sampling methods on bias and variance in stochastic linear programs. *Computational Optimization and Applications* 51:51–75.

Gerstner, T., and M. Griebel. 1998. Numerical integration using sparse grids. *Numer. Alg.* 18:209–32.

He, X., and P. Z. G. Qian. 2011. Nested orthogonal array-based Latin hypercube designs. *Biometrika* 98:721–731.

Hedayat, A. S., N. J. A. Sloane, and John Stufken. 1999. *Orthogonal arrays: Theory and applications*. New York: Springer.

Hwang, Y., X. He, and P. Z. G. Qian. 2013. Sliced orthogonal array based latin hypercube designs. Submitted.

Iman, R. L., and W. J. Conover. 1982. A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics, Part B-Simulation and Computation* 11:311–34.

Jones, D. R., M. Schonlau, and Welch W. J. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13:455–492.

Joseph, V. R., and Y. Hung. 2008. Orthogonal-maximin Latin hypercube designs. *Statistica Sinica* 18:171–86.

Koehler, J., and A. B. Owen. 1996. Computer experiments. In *Handbook of statistics, 13: Design and analysis of experiments*, ed. S. Ghosh and C.R. Rao, 261–308. Amsterdam: North-Holland.

Lehmann, E. L. 1966. Some concepts of dependence. *Annals of Mathematical Statistics* 37:1137–1153.

Leoppky, J. L., L. M. Moore, and B. J. Williams. 2010. Batch sequential designs for computer experiments. *Journal of Statistical Planning and Inference* 140:1452–1464.

Lin, C. D., D. Bingham, R. R. Sitter, and B. Tang. 2010. A new and flexible method for constructing designs for computer experiments. *Ann. Statist.* 38:1460–77.

Lin, C. D., R. Mukerjee, and B. Tang. 2009. Construction of orthogonal and nearly orthogonal Latin hypercubes. *Biometrika* 96:243–7.

Linderoth, J. T., A. Shapiro, and S. J. Wright. 2006. The empirical behavior of sampling methods for stochastic programming. *Ann. Oper. Res.* 142:215–241.

Loeppky, J. L., L. M. Moore, and B. J. Williams. 2011. Projection array based designs for computer experiments. *Journal of Statistical Planning and Inference* 142:1493–1505.

Loh, W. L. 1996. On Latin hypercube sampling. *Ann. Statist.* 24:2058–80.

Louveaux, F. V., and Y. Smeers. 1988. Optimal investments for electricity generation: a stochastic model and a test problem. *in Y. Ermoliev and R. J-B. Wets (eds.), Numerical techniques for stochastic optimization problems* 445–452.

Mak, W. K., D. P. Morton, and R. K. Wood. 1999. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* 24:47–56.

Mann, H.B., and A. Wald. 1943. On stochastic limit and order relationships. *The Annals of Mathematical Statistics* 14:217–226.

McKay, M. D., W. J. Conover, and R. J. Beckman. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21:239–45.

Mease, D., and D. Bingham. 2006. Latin hyperrectangle sampling for computer experiments. *Technometrics* 48:467–477.

Homem-de Mello, T. 2008. On rates of convergence for stochastic optimization problems under non-independent and identically distributed sampling. *SIAM J. Optim.* 19:524–51.

Morris, M., and T. Mitchell. 1995a. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference* 43:381–402.

Morris, M. D., and T. J. Mitchell. 1995b. Exploratory designs for computer experiments. *J. Statist. Plann. Inference* 43:381–402.

Morris, M. D., T. J. Mitchell, and D. Ylvisaker. 1993. Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics* 35:243–55.

Mulvey, J. M., and A. Ruszczyński. 1993. A new scenario decomposition method for large-scale stochastic optimization. *Operations Research* 43(3):477–490.

Nguyen, N. K. 1996. A note on the construction of near-orthogonal arrays with mixed levels and economic run size. *Technometrics* 38:279–83.

Niederreiter, H. 1992. *Random number generation and quasi-monte carlo methods.* SIAM, Philadelphia.

Norkin, V., G. Pflug, and A. Ruszczyński. 1998. A branch and bound method for stochastic global optimization. *Mathematical Programming* 83:425–450.

Owen, A. B. 1992. A central limit theorem for Latin hypercube sampling. *Journal of The Royal Statistical Society Series B* 54:541–551.

———. 1994a. Controlling correlations in Latin hypercube samples. *J. Am. Statist. Assoc.* 89:1517–22.

———. 1994b. Lattice sampling revisited: Monte Carlo variance of means over randomized orthogonal arrays. *Ann. Statist.* 22:930–45.

———. 1995. *Randomly permuted* (t, m, s)*-nets and* (t, s)*-sequences*, vol. 106 of *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics*, 299–317. New York: Springer.

———. 1997. Scrambled net variance for integrals of smooth functions. *Annals of Statistics* 25:1541–1562.

Pang, F., M. Q. Liu, and D. K. J. Lin. 2009. A construction method for orthogonal Latin hypercube designs with prime power levels. *Statistica Sinica* 19:1721–1728.

Patterson, H. D. 1954. The errors of lattice sampling. *J. R. Statist. Soc. B* 16: 140–9.

Qian, P. Z. G. 2009. Nested Latin hypercube designs. *Biometrika* 96:957–70.

———. 2012. Sliced Latin hypercube designs. *J. Am. Statist. Assoc.* 107(497): 393–9.

Rao, C. R. 1946. Hypercubes of strength d leading to confounded designs in factorial experiments. *Bull. Calcutta Math. Soc.* 38:67–78.

———. 1947. Factoial experiments derivable from combinatorial arrangements of arrys. *J. Royal Statist. Soc. (Suppl.)* 9:128–139.

———. 1949. On a class of arrangements. *Proc. Edinburgh Math. Soc.* 8:119–125.

Rennen, G., B. Husslage, E. R. Van Dam, and D. Den Hertog. 2010. Nested maximin Latin hypercube designs. *Structural and Multidisciplinary Optimization* 41:371–395.

Rosenblatt, M. 1953. Remark on a multivariate transformation. *Ann. Math. Stat.* 23:470–472.

Sallaberry, C. J., J. C. Helton, and S. C. Hora. 2008. Extension of Latin hypercube samples with correlated variables. *Reliability Engineering and System Safety* 93: 1047–1059.

Santner, T. J., B. J. Williams, and W. I. Notz. 2003. *The design and analysis of computer experiments*. New York: Springer.

Sen, S., R. D. Doverspike, and S. Cosares. 1994. Network planning with random demand. *Telecommunication Systems* 3(1):11–30.

Shapiro, A. 1991. Asymptotic analysis of stochastic programs. *Ann. Oper. Res.* 30: 169–186.

Shapiro, A., D. Dentcheva, and A. Ruszczynski. 2009. *Lectures on stochastic programming: Modeling and theory*. SIAM.

Stein, M. 1987. Large-sample properties of simulations using Latin hypercube sampling. *Technometrics* 29:143–51.

Steinberg, D. M., and D. K.J. Lin. 2006. A construction method for orthogonal Latin hypercube designs. *Biometrika* 93:279–288.

Sun, F. S., M. Q. Liu, and D. K. J. Lin. 2009. Construction of orthogonal Latin hypercube designs. *Biometrika* 96:971–4.

Tang, B. 1993. Orthogonal array-based Latin hypercubes. *J. Am. Statist. Assoc.* 88:1392–7.

———. 1998. Selecting Latin hypecubes using correlation criteria. *Statistica Sinica* 8:965–77.

Tang, Q., and P. Z. G. Qian. 2010. Enhancing the sample average approximation method with U designs. *Biometrika* 97:947–960.

Thain, Douglas, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience* 17(2-4): 323–356.

Tong, C. 2006. Refinement strategies for stratified sampling methods. *Reliability Engineering and System Safety* 91:1257–1265.

Wang, G. G. 2003. Adaptive response surface method using inherited Latin hypercube design points. *Journal of Mechanical Design* 125:210–220.

Wang, J. R., and C. F. J. Wu. 1992. Nearly orthogonal arrays with mixed levels and small runs. *Technometrics* 34:409–422.

Xiu, D. 2010. *Numerical methods for stochastic computations: A spectral method approach*. Princeton University Press.

Yang, J. F., C. D. Lin, P. Z. G. Qian, and D. J. K. Lin. 2013. Construction of sliced orthogonal Latin hypercube designs. *Statistica Sinica* 23:1117–1130.

Ye, K. Q. 1998. Orthogonal column Latin hypercubes and their application in computer experiments. *J. Am. Statist. Assoc.* 93(444):1430–9.