| Statistica Sinica Preprint No: SS-2016-0151.R2 | |
|---|---|
| **Title** | Controlling Correlations in Sliced Latin Hypercube Designs |
| **Manuscript ID** | SS-2016-0151.R2 |
| **URL** | http://www.stat.sinica.edu.tw/statistica/ |
| **DOI** | 10.5705/ss.202016.0151 |
| **Complete List of Authors** | Peter Qian and<br>Jiajie Chen |
| **Corresponding Author** | Peter Qian |
| **E-mail** | peterq@stat.wisc.edu |

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



# Controlling Correlations in Sliced Latin Hypercube Designs

Jiajie Chen AND Peter Z. G. Qian

*Wells Fargo and The University of Wisconsin – Madison*

*Abstract:* A sliced Latin hypercube design is a special Latin hypercube design that can be partitioned into smaller Latin hypercube designs. We propose an algorithm to construct sliced Latin hypercube designs with controlled column-wise correlations for each slice and the entire design. The proposed algorithm can significantly decrease the column-wise correlations in each slice as the number of slices increases even if the number of runs in each slice is fixed. The algorithm is flexible in sample size and can be extended to control the quadratic canonical correlations of the larger design. The convergence behavior of the algorithm is studied and the effectiveness of the algorithm is illustrated by several examples.

*Key words and phrases:* Computer experiments, design of experiments, numerical integration, space-filling design, uncertainty quantification.

## 1. Introduction

The Latin hypercube design has been widely used in computer experiments (McKay et al. (1979)). A Latin hypercube design can be used to estimate the expected output $\mu = E[f(\mathbf{x})]$ of a computer model $f(\mathbf{x}) \in \mathbb{R}$ of a set of inputs $\mathbf{x} = (x_1, \ldots, x_p)$ with a specified distribution. Without

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



loss of generality, assume $\mathbf{x}$ is uniformly distributed on the unit cube $(0,1]^p$. For a sample of $n$ runs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from the uniform distribution on $(0,1]^p$, an estimate of $\mu$ is obtained as

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

A *Latin hypercube design* is obtained by generating $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ as

$$x_{ik} = (\pi_k(i) - \eta_{ik})/n, \qquad i = 1, \ldots, n, \quad k = 1, \ldots, p, \tag{1.2}$$

where the $\pi_k$ is an independent uniform permutation of $\{1, \ldots, n\}$ with all $n!$ possible permutations being equally probable, the $\eta_{ik}$ are independent $U[0,1)$ variables, and the $\eta_{ik}$ and the $\pi_\ell$ are mutually independent for $i = 1, \ldots, n$, $k, \ell = 1, \ldots, p$. By stratifying each dimension in a Latin hypercube design, the variance of $\hat{\mu}$ in (1.1) under this sampling scheme is smaller than its counterpart under Monte Carlo sampling (Stein (1987); Loh (1996)).

Qian (2012) introduced a new type of design called *sliced Latin hypercube design*. For positive integers $n$, $p$, $t$, and $N$ with $N = nt$, an $N \times p$ sliced Latin hypercube design is a special Latin hypercube design that can be partitioned into $t$ slices of $n \times p$ Latin hypercube designs. Qian (2012) established some results to demonstrate advantages of sliced Latin hyper-

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



cube designs over independent Latin hypercube designs for running multiple similar computer models or a computer model in batches.

Since the designs generated by Qian (2012) use independent random permutations for different columns, they are not guaranteed to achieve good two- or higher-dimensional space-filling structure. Yang et al. (2013) and Ba et al. (2015) use different methods to construct sliced orthogonal Latin hypercube designs of first-order or second-order that have orthogonal columns for each slice and the entire design. Other related work includes Huang et al. (2014), Yang et al. (2014) and Yin et al. (2014). Because these methods depend on the existence of orthogonal arrays (Hedayat et al. (1999)) or orthogonal Latin hypercubes (Ye (1998); Steinberg and Lin (2006); Sun et al. (2009)), they have restrictive sample sizes. Ba et al. (2015) and Chen et al. (2014b) constructed maximin based optimal sliced Latin hypercube designs that are useful for prediction purposes (Johnson et al. (1990)).

We propose a new algorithm to control the column-wise correlations in a sliced Latin hypercube design. The algorithm simultaneously controls the correlations in each slice and the entire design. The method extends the ranked Gram-Schmidt (RGS) algorithm of Owen (1994) in a new direction. The proposed method is flexible in sample size. Different from Owen (1994), the challenge here is the two-layer structure of a sliced Latin hyper-

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



cube design. The remainder of the paper is organized as follows. Section 2 details the proposed algorithm. Section 3 studies its performance. Section 4 extends the algorithm to further control quadratic canonical correlations. Section 5 illustrates the proposed algorithm in numerical integration. Section 6 concludes the article with some discussion.

## 2. Algorithms

Let $\mathbf{X}$ denote an $N \times p$ sliced Latin hypercube design with $t$ slices of $n \times p$ Latin hypercube designs $\mathbf{X}_{(1)}, \ldots, \mathbf{X}_{(t)}$. We consider the problem of controlling correlations in $\mathbf{X}$ with $p < n$. Obviously, if one can control the correlations in each slice, then the correlations in the entire design are controlled as well. Ba et al. (2015) showed that each $\mathbf{X}_{(r)}$ is based on an $n \times p$ Latin hypercube $\mathbf{A}_{(r)}$ of which each column is a permutation of $\{1, \ldots, n\}$. Let $\mathbf{a}_{(r)}^k$ denote the $k$th column of $\mathbf{A}_{(r)}$ for $k = 1, \ldots, p$. Let $\texttt{takeout}(\mathbf{z}, \mathbf{y})$ denote the residual vector of a linear regression with an intercept, $\mathbf{z}$ denote the values of the predictor, and $\mathbf{y}$ denote the values of the response. Let $\texttt{rank}(\mathbf{y})$ denote the vector of ranks of $\mathbf{y}$. Algorithm 1 controls the correlations in each slice by first generating $t$ independent correlation-controlled Latin hypercubes $\mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(t)}$ by the RGS algorithm (Owen (1994)).

Qian's original construction of sliced Latin hypercube designs is equiv-

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



**Algorithm 1**

**Step 1.** Randomly obtain $t$ independent Latin hypercubes $\mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(t)}$.

**Step 2.** For $r = 1, \ldots, t$, generate a correlation-controlled Latin hypercube $\mathbf{A}_{(r)}$ using the RGS algorithm by alternating forward and backward steps:

**Forward:**

for $k = 2, ..., p$
  for $\ell = 1, ..., k-1$
    $\mathbf{a}^{\ell}_{(r)} \leftarrow \texttt{takeout}(\mathbf{a}^{k}_{(r)}, \mathbf{a}^{\ell}_{(r)})$
for $k = 1, ..., p$
  $\mathbf{a}^{k}_{(r)} \leftarrow \texttt{rank}(\mathbf{a}^{k}_{(r)})$,

**Backward:**

for $k = p-1, ..., 1$
  for $\ell = p, ..., k+1$
    $\mathbf{a}^{\ell}_{(r)} \leftarrow \texttt{takeout}(\mathbf{a}^{k}_{(r)}, \mathbf{a}^{\ell}_{(r)})$
for $k = 1, ..., p$
  $\mathbf{a}^{k}_{(r)} \leftarrow \texttt{rank}(\mathbf{a}^{k}_{(r)})$,

where $\leftarrow$ denotes assignment. Stack the $\mathbf{A}_{(r)}$ row by row to form an $N \times p$ matrix $\mathbf{A}$. Let $\mathbf{a}^k$ denote the $k$th column of $\mathbf{A}$. Let $a_{ik}$ denote the $i$th entry in $\mathbf{a}^k$.

**Step 3.** Let $\mathbf{\Theta} = (\theta_{ik})$ denote an $N \times p$ matrix with columns $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^p$. For each level $j$, there exists a unique set of indexes $1 \le i_{(j,1)} < \cdots < i_{(j,t)} \le N$ such that $a_{i_{(j,1)}k} = \cdots = a_{i_{(j,t)}k} = j$. Let $\boldsymbol{\theta}^k(j) = (\theta_{i_{(j,1)}k}, \ldots, \theta_{i_{(j,t)}k})^T$. Obtain $\boldsymbol{\theta}^k(j)$ as an independent uniform permutation of $\{1. \ldots, t\}$ for $k = 1, \ldots, p$ and $j = 1, \ldots, n$.

**Step 4.** Obtain the sliced Latin hypercube design $\mathbf{X}$ as

$$\mathbf{X} = N^{-1}[t(\mathbf{A} - 1) + \mathbf{\Theta} - \Gamma], \tag{2.1}$$

where $\Gamma$ is an $N \times p$ matrix whose entries are independent $U[0, 1)$ random variables that are also mutually independent with entries in $\mathbf{A}$ and $\mathbf{\Theta}$.

alent to Algorithm 1 without Step 2 (Ba et al. (2015); Chen et al. (2014a)). For simplicity, we fix the $\Gamma$ in (2.1) at .5. Let $\mathbf{\Theta}_{(r)}$ denote the $r$th slice in $\mathbf{\Theta}$ satisfying $\mathbf{X}_{(r)} = N^{-1}[t(\mathbf{A}_{(r)} - 1) + \mathbf{\Theta}_{(r)} - .5]$. While Step 2 in Algorithm 1 controls the column-wise correlations in the $\mathbf{A}_{(r)}$, entries in each $\mathbf{\Theta}_{(r)}$ are independently generated from a discrete uniform distribution on $\{1, \ldots, t\}$ in Step 3. If we can select and organize the $\mathbf{\Theta}_{(r)}$ in a better way, we might be able to improve Algorithm 1 to achieve lower correlations in

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



each slice. To this end, two issues need to be addressed: how to select the $\mathbf{\Theta}_{(r)}$ given the $\mathbf{A}_{(r)}$, and how to do so for all slices while keeping each $\boldsymbol{\theta}^k(j)$ as a permutation of $\{1, \ldots, t\}$.

Our algorithm tackles these two issues simultaneously. By allowing exchanges of elements in the $\mathbf{\Theta}_{(r)}$ across different slices. The algorithm requires construction of an $N \times (t+1)$ auxiliary matrix $\mathbf{B}$ with sliced $\mathbf{B}_{(1)}, \ldots, \mathbf{B}_{(t)}$. Let $\mathbf{b}^k$ denote the $k$th column of $\mathbf{B}$ and let $\mathbf{b}^k_{(r)}$ denote the $k$th column of $\mathbf{B}_{(r)}$. Let $\mathbf{x}^k(j)$ denote a sub-vector of $\mathbf{x}^k$ with corresponding values in $\mathbf{a}^k$ equal $j$ (similar as $\boldsymbol{\theta}^k(j)$ in Step 3 of Algorithm 1). Details are summarized in Algorithm 2.

**Proposition 1.** *For any* $1 \leq k < \ell \leq p$, $\mathbf{x}^\ell \leftarrow$ `takeout`$(\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}, \mathbf{x}^\ell)$ *in Algorithms 2 is equivalent to:* $\mathbf{x}^\ell_{(r)} \leftarrow$ `takeout`$(\mathbf{x}^k_{(r)}, \mathbf{x}^\ell_{(r)})$ *for* $r = 1, \ldots, t$.

Proposition 1 indicates that the output of the `takeout` step in Algorithm 2 can be duplicated by taking a takeout step for every slice. The `takeout` step in Algorithm 2 is applied to optimize the $\mathbf{X}_{(r)}$. Instead of ranking $\mathbf{x}^k$ directly, its ranking procedure proceeds in two steps. First, the ranks of $\mathbf{x}^k_{(r)}$ are assigned to $\mathbf{a}^k_{(r)}$ independently for $r = 1, \ldots, t$ to guarantee that each slice is based on a Latin hypercube, and second, the ranks of $\mathbf{x}^k(j)$ are given to $\boldsymbol{\theta}^k(j)$ to make sure that $\boldsymbol{\theta}^k(j)$ is a permutation of $\{1, \ldots, t\}$ for $j = 1 \ldots, n$. When $n = 1$, Algorithm 1 gives a randomized Latin hypercube

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



**Algorithm 2**

**Step 1.** Randomly generate a sliced Latin hypercube design $\mathbf{X}$.

**Step 2.** Alternate the following procedures:

**Forward:**

for $k = 2, \ldots, p$
  for $r = 1, \ldots, t$
    for $s = 1, \ldots, t$

$$\mathbf{b}_{(s)}^{r} \leftarrow \begin{cases} \mathbf{x}_{(r)}^{k}, & \text{if } s = r \\ \bar{\mathbf{x}}_{(r)}^{k}, & \text{otherwise.} \end{cases}$$

  for $\ell = 1, \ldots, k-1$
    for $r = 1, \ldots, t$
      $\mathbf{b}_{(r)}^{t+1} \leftarrow \bar{\mathbf{x}}_{(r)}^{\ell} - .5$
    $\mathbf{x}^{\ell} \leftarrow \texttt{takeout}(\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}, \mathbf{x}^{\ell})$
for $k = 1, \ldots, p$
  for $r = 1, \ldots, t$
    $\mathbf{a}_{(r)}^{k} \leftarrow \texttt{rank}(\mathbf{x}_{(r)}^{k})$,
  for $j = 1, \ldots, n$
    $\boldsymbol{\theta}^{k}(j) \leftarrow \texttt{rank}(\mathbf{x}^{k}(j))$,
  $\mathbf{x}^{k} \leftarrow [t\mathbf{a}^{k} + \boldsymbol{\theta}^{k} - (t + .5)]/N$,

**Backward:**

for $k = p-1, \ldots, 1$
  for $r = 1, \ldots, t$
    for $s = 1, \ldots, t$

$$\mathbf{b}_{(s)}^{r} \leftarrow \begin{cases} \mathbf{x}_{(r)}^{k}, & \text{if } s = r \\ \bar{\mathbf{x}}_{(r)}^{k}, & \text{otherwise.} \end{cases}$$

  for $\ell = p, \ldots, k+1$
    for $r = 1, \ldots, t$
      $\mathbf{b}_{(r)}^{t+1} \leftarrow \bar{\mathbf{x}}_{(r)}^{\ell} - .5$
    $\mathbf{x}^{\ell} \leftarrow \texttt{takeout}(\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}, \mathbf{x}^{\ell})$
for $k = 1, \ldots, p$
  for $r = 1, \ldots, t$
    $\mathbf{a}_{(r)}^{k} \leftarrow \texttt{rank}(\mathbf{x}_{(r)}^{k})$,
  for $j = 1, \ldots, n$
    $\boldsymbol{\theta}^{k}(j) \leftarrow \texttt{rank}(\mathbf{x}^{k}(j))$,
  $\mathbf{x}^{k} \leftarrow [t\mathbf{a}^{k} + \boldsymbol{\theta}^{k} - (t + .5)]/N$,

where $\texttt{takeout}(\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}, \mathbf{x}^{\ell})$ is the residual vector same as previously defined in Section 2.1 except with $\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}$ as predictors.

design $\mathbf{X}$ based on (1.2) while Algorithm 2 is reduced to the RGS algorithm that can control the correlations in $\mathbf{X}$. An example of using Algorithm 2 to generate a correlation-controlled sliced Latin hypercube design is provided in the supplementary document.

## 3. Performance

### 3.1. Convergence of algorithms

An algorithm converges if the design stops updating after several iterations. Owen (1994) observed that the RGS algorithm for a Latin hy-

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



percube design mostly converged within a few iterations when $p$ is small or much smaller than $n$. In cases where non-convergence occurred with relatively small $p$, the algorithm alternated two designs as the algorithm alternated between forward and backward steps. The same convergence behavior should hold for Algorithm 1 since it consists of $t$ independent RGS algorithms on the $\mathbf{A}_{(r)}$.

For Algorithm 2 applied on a sliced Latin hypercube design, we talk of strong convergence, the entire design $\mathbf{X}$ stops updating, and weak convergence, $\mathbf{A}$ stops updating. Strong convergence of Algorithm 2 is expected to occur less frequently than the convergence of the RGS algorithm because the elements can still exchange between slices even after the algorithm converges in each slice. Such exchanges become inevitable especially when $t$ or $p$ are large. We generated twenty sliced Latin hypercube designs for each combination of $n \in \{10, 50\}$, $t \in \{5, 15\}$, and $p \in \{4, 9\}$. We tracked the sequence of designs generated in forty iterations for each case. We observed strong convergence in twenty cases where $t = 5$ and $p = 4$, four cases where $t = 15$ and $p = 4$, and zero cases where $p = 9$. The algorithm did not converge or alternate the $\mathbf{X}$ in twenty-nine cases including all twenty cases with $n = 10$, $t = 15$, and $p = 9$. Among the other one hundred and seven cases where the algorithm alternated $\mathbf{X}$, we observed eighty-eight cases with

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



weak convergence.

The algorithm does reduce the column-wise correlation within a few iterations even without weak convergence. Consider root mean square correlation (Owen, 1994) as the performance measure:

$$\rho_{\text{rms}}(\mathbf{D}) = \sqrt{\frac{\sum_{1\le k<\ell\le p}[\rho(\mathbf{d}^k, \mathbf{d}^\ell)]^2}{p(p-1)/2}}, \tag{3.1}$$

where $\mathbf{D}$ is a design matrix with $p$ columns and $\rho(\mathbf{d}^k, \mathbf{d}^\ell)$ is the sample correlation between the $k$th and $\ell$th columns. The left and right plots in Figure 1 show typical sequences of $\rho_{\text{rms}}$ after each iteration for $\mathbf{X}$ and the $\mathbf{X}_{(r)}$ when the algorithm alternates both $\mathbf{X}$ and $\mathbf{A}$ after some iterations, and when it does not converge or alternate in forty iterations. In following, we set the maximum number of alternations between the forward and backward procedures at ten for all the algorithms based on the RGS algorithm.

### 3.2. Controlling column-wise correlations

Let SL denote the sampling scheme in Qian (2012). Let CSL1 and CSL2 denote schemes that use Algorithms 1 and 2, respectively. To compare the performances in controlling column-wise correlations, we applied Algorithms 1 and 2 with $n \in \{10, 20, 50, 100\}$, $t \in \{5, 10, 20, 50\}$, and $p = 4$. For each combination of $n$ and $t$, we replicated twenty times. Let $\rho_{\text{rms}}^{\text{csl1}}$ and $\rho_{\text{rms}}^{\text{csl2}}$ denote the root mean square correlation under Algorithms 1 and 2,

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



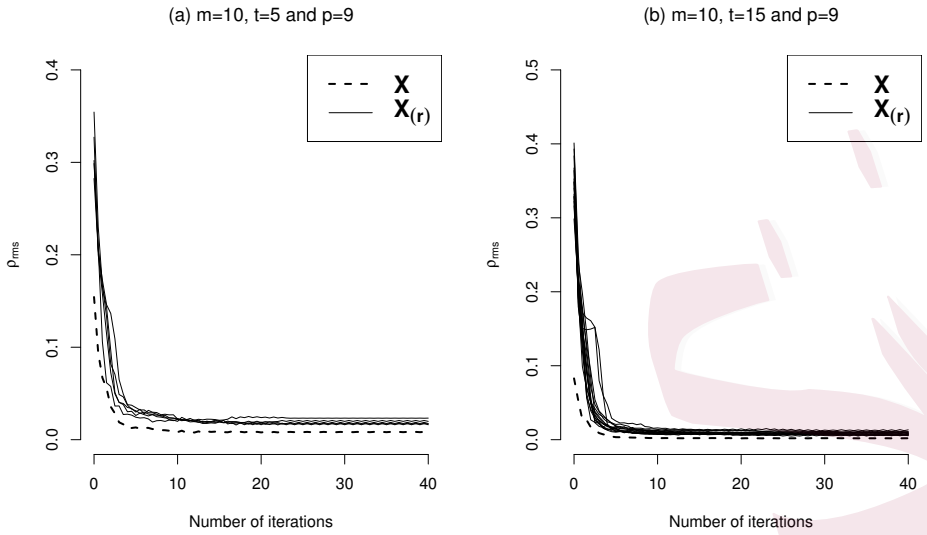


Figure 1: The $\rho_{\text{rms}}(\mathbf{X})$ (dashed) and the $\rho_{\text{rms}}(\mathbf{X}_{(r)})$ (solid) of the sequence of designs generated by Algorithm 2 for (a) $n = 10$, $t = 5$ and $p = 9$ (b) $n = 10$, $t = 15$ and $p = 9$.

respectively. We used the result for $\mathbf{X}_{(1)}$ to represent results for any single slice as the slices are exchangeable under both schemes. We randomly generated sliced Latin hypercube designs under SL to obtain $\rho^{\text{sl}}_{\text{rms}}(\mathbf{X}_{(1)})$ and $\rho^{\text{sl}}_{\text{rms}}(\mathbf{X})$ as the baseline. Figure 2 presents a plot of $\rho^{\text{csl1}}_{\text{rms}}$, $\rho^{\text{csl2}}_{\text{rms}}$, and $\rho^{\text{sl}}_{\text{rms}}$ versus $n$ on a log-log scale for $t = 20$. Here the root mean square correlations for both $\mathbf{X}_{(1)}$ and $\mathbf{X}$ converge faster under CSL1 and CSL2 at almost the same rate as $n$ grows, but CSL2 has the smaller magnitude.

Figure 3 presents a similar plot but with changing $t$ and fixed $n = 20$. We observe that CSL2 offers the smallest $\rho_{\text{rms}}$ not only in magnitude but in convergence rate as well.

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



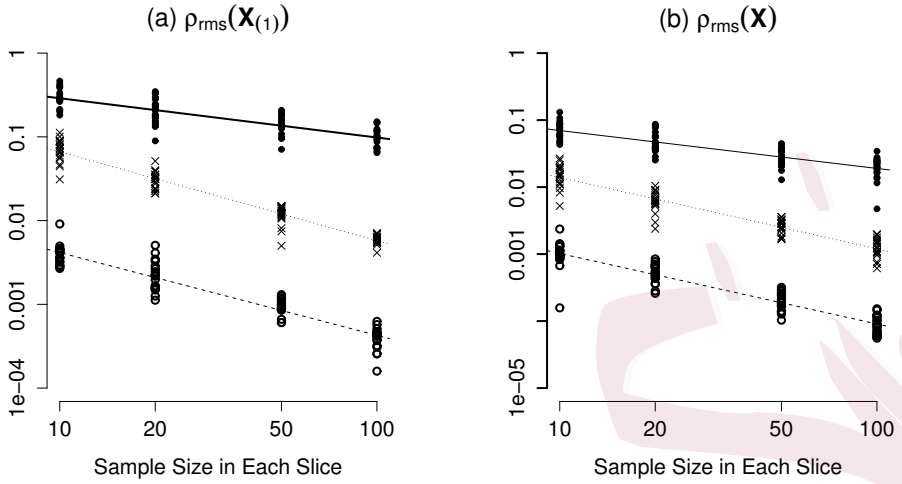


Figure 2: Root mean square correlation $\rho_{\text{rms}}$ of $\mathbf{X}_{(1)}$ and $\mathbf{X}$ versus sample size $n$ in each slice for $t = 20$ and $p = 4$. Dots denote $\rho_{\text{rms}}^{\text{sl}}$, crosses denote $\rho_{\text{rms}}^{\text{csl1}}$, and circles denote $\rho_{\text{rms}}^{\text{csl2}}$. The solid, dotted, dashed reference lines are the least squares regressions of $\log(\rho_{\text{rms}}^{\text{sl}})$, $\log(\rho_{\text{rms}}^{\text{csl1}})$ and $\log(\rho_{\text{rms}}^{\text{csl2}})$ on $\log(n)$, respectively.

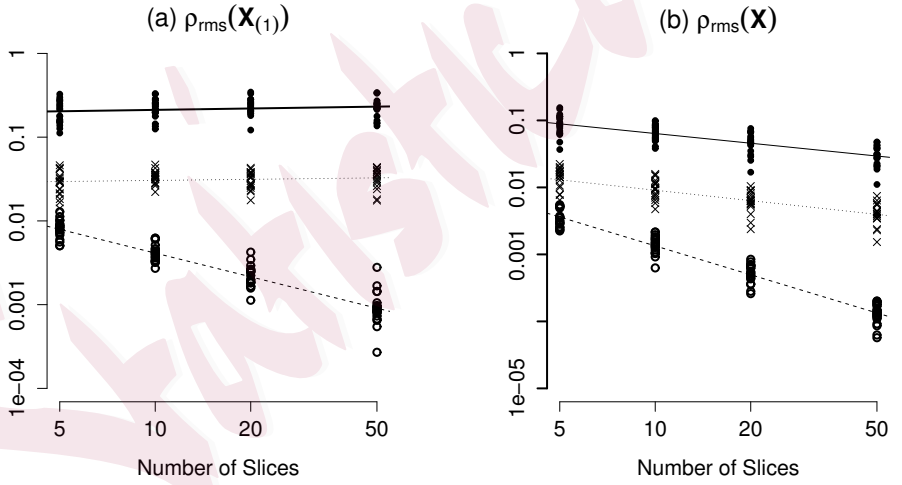


Figure 3: Root mean square correlation $\rho_{\text{rms}}$ of $\mathbf{X}_{(1)}$ and $\mathbf{X}$ versus number of slices $t$ in each slice for $n = 20$ and $p = 4$. Dots denote $\rho_{\text{rms}}^{\text{sl}}$, crosses denote $\rho_{\text{rms}}^{\text{csl1}}$, and circles denote $\rho_{\text{rms}}^{\text{csl2}}$. The solid, dotted, dashed reference lines are the least squares regressions of $\log(\rho_{\text{rms}}^{\text{sl}})$, $\log(\rho_{\text{rms}}^{\text{csl1}})$ and $\log(\rho_{\text{rms}}^{\text{csl2}})$ on $\log(n)$, respectively.

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



We fit the ordinary least square regression of $\log(\rho_{\text{rms}})$ versus $\log(n)$ and $\log(t)$ and found

$$\log(\rho^{\text{sl}}_{\text{rms}}(\mathbf{X}_{(1)})) \approxeq .01 - .53\log(n),$$

$$\log(\rho^{\text{csl1}}_{\text{rms}}(\mathbf{X}_{(1)})) \approxeq -.35 - 1.05\log(n),$$

$$\log(\rho^{\text{csl2}}_{\text{rms}}(\mathbf{X}_{(1)})) \approxeq -.29 - .99\log(n) - .98\log(t),$$

$$\log(\rho^{\text{sl}}_{\text{rms}}(\mathbf{X})) \approxeq .04 - .52\log(n) - .52\log(t),$$

$$\log(\rho^{\text{csl1}}_{\text{rms}}(\mathbf{X})) \approxeq -.27 - 1.09\log(n) - .48\log(t),$$

$$\log(\rho^{\text{csl2}}_{\text{rms}}(\mathbf{X})) \approxeq -.15 - 1.06\log(n) - 1.42\log(t),$$

where $\log(t)$ is insignificant and dropped for $\log(\rho^{\text{sl}}_{\text{rms}}(\mathbf{X}_{(1)}))$ and $\log(\rho^{\text{csl1}}_{\text{rms}}(\mathbf{X}_{(1)}))$.

Empirical analysis then indicates that $\rho_{\text{rms}}(\mathbf{X}_{(1)})$ has a rate close to $n^{-1}$ under CSL1 and $N^{-1}$ under CSL2. Following Owen (1994), we consider why $n^{-1}$ and $N^{-1}$ might be the best that could be achieved under CSL1 and CSL2, respectively, when $p$ is fixed. By Proposition 1, each update in a single `takeout` step of $\mathbf{X}_{(r)}$ in Algorithm 2 is equivalent to

$$\mathbf{x}^{\ell}_{(r)} \leftarrow \mathbf{x}^{\ell}_{(r)} - .5 - (\mathbf{x}^{k}_{(r)} - .5)\rho(\mathbf{x}^{k}_{(r)}, \mathbf{x}^{\ell}_{(r)})\sigma(\mathbf{x}^{\ell}_{(r)})/\sigma(\mathbf{x}^{k}_{(r)}),$$

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



where $\sigma(\mathbf{x}^k_{(r)})$ and $\sigma(\mathbf{x}^\ell_{(r)})$ are the standard deviations of the two columns. Hence, the amount of change for each component in $\mathbf{x}^\ell_{(r)}$ after a single `takeout` step is $O_p[\rho(\mathbf{x}^k_{(r)}, \mathbf{x}^\ell_{(r)})]$. Changing $\mathbf{x}^\ell_{(r)}$ means a change in $\texttt{rank}(\mathbf{x}^\ell_{(r)})$, which requires modifying some components in $\mathbf{x}^\ell_{(r)}$ by $O(N^{-1})$. Suppose $p$ vectors are to be taken out of $\mathbf{x}^\ell_{(r)}$ before it is re-ranked. As $p$ is fixed, Algorithm 2 stops updating the $\mathbf{X}_{(r)}$ if $\rho(\mathbf{x}^k_{(r)}, \mathbf{x}^\ell_{(r)})$ is small compared to $N^{-1}$, which explains $\rho^{\text{csl2}}_{\text{rms}}(\mathbf{X}_{(1)}) = O_p(N^{-1})$. In Algorithm 1, the RGS procedure stops updating if $\rho(\mathbf{a}^k_{(r)}, \mathbf{a}^\ell_{(r)})$ is small compared to $n^{-1}$. Assuming $\rho(\mathbf{a}^k_{(r)}, \mathbf{a}^\ell_{(r)}) = O_p(n^{-1})$, we then have $\rho(\mathbf{x}^k_{(r)}, \mathbf{x}^\ell_{(r)}) = O_p[\rho(\mathbf{a}^k_{(r)}, \mathbf{a}^\ell_{(r)})] = O_p(n^{-1})$ according to (2.1) under CSL1.

## 4. Controlling canonical correlations

Tang (1998) proposed an algorithm that extends the RGS algorithm to control the polynomial canonical correlations in a Latin hypercube design. For vectors $\mathbf{y} = (y_1, \ldots, y_n)^T$ and $\mathbf{z} = (z_1, \ldots, z_n)^T$, let $\{\mathbf{y}, \mathbf{y}^2, \ldots, \mathbf{y}^w\}$ and $\{\mathbf{z}, \mathbf{z}^2, \ldots, \mathbf{z}^w\}$ denote two sets of vectors with $\mathbf{y}^j = (y^j_1, \ldots, y^j_n)^T$ and $\mathbf{z}^j = (z^j_1, \ldots, z^j_n)^T$. The maximal correlation between linear combinations of these two sets of vectors is the polynomial canonical correlation of order $w$ between $\mathbf{y}$ and $\mathbf{z}$, denoted $\rho_w(\mathbf{y}, \mathbf{z})$. Tang (1998) demonstrated that one can control the $\rho_2(\mathbf{x}^k, \mathbf{x}^\ell)$ (quadratic canonical correlation) in a Latin hypercube design by changing the `takeout` step in the RGS algorithm from $\mathbf{x}^\ell \leftarrow$

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



$\texttt{takeout}(\mathbf{x}^k, \mathbf{x}^\ell)$ to $\mathbf{x}^\ell \leftarrow \texttt{takeout}[\mathbf{x}^k, (\mathbf{x}^k)^2, \mathbf{x}^\ell]$, where $(\mathbf{x}^k)^2$ represents an extra predictor vector.

Using this idea, we can easily modify Algorithm 2 for reducing the quadratic canonical correlations of the entire design $\mathbf{X}$. After the modification, the `takeout` step in Algorithm 2 is

$$\mathbf{x}^\ell \leftarrow \texttt{takeout}[\mathbf{b}^1, \ldots, \mathbf{b}^{t+1}, (\mathbf{x}^k)^2, \mathbf{x}^\ell].$$

Let QCSL denote a scheme that uses Algorithm 2 with this modification. To examine the performance of QCSL, we used the root mean square quadratic canonical correlation of a design $\mathbf{X}$ given by

$$\rho_{\text{rmq}}(\mathbf{X}) = \sqrt{\frac{\sum_{1\le k<\ell\le p}[\rho_2(\mathbf{x}^k, \mathbf{x}^\ell)]^2}{p(p-1)/2}}. \tag{4.1}$$

We also examined the root mean square correlation $\rho_{\text{rms}}$ in (3.1) to see if the column-wise correlations were still well controlled.

Figure 4 presents boxplots of $\rho_{\text{rms}}(\mathbf{X}_{(1)})$, $\rho_{\text{rms}}(\mathbf{X})$, and $\rho_{\text{rmq}}(\mathbf{X})$ under different sampling schemes. We observe that the $\rho_{\text{rms}}$ values are significantly lower under CSL1, CSL2, and QCSL than those under SL, that CSL2 and QCSL can further reduce the $\rho_{\text{rms}}$, and that QCSL can control $\rho_{\text{rmq}}(\mathbf{X})$ effectively.

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



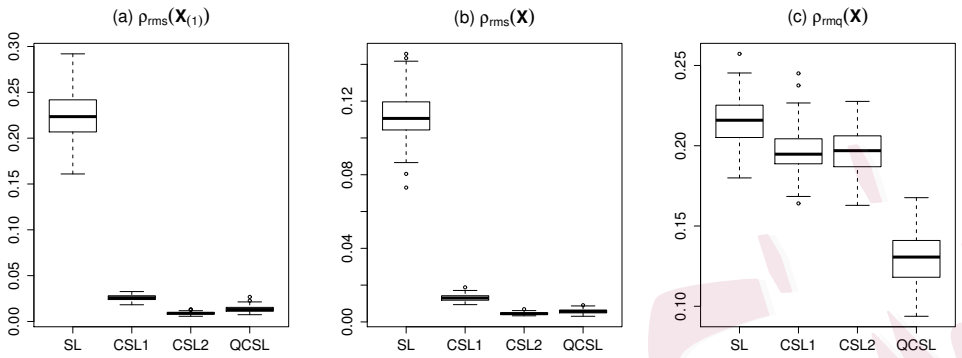


Figure 4: Boxplots of $\rho_{\text{rms}}(\mathbf{X}_{(1)})$, $\rho_{\text{rms}}(\mathbf{X})$ and $\rho_{\text{rmq}}(\mathbf{X})$ under SL, CSL1, CSL2 and QCSL with $n = 20$, $p = 8$ and $t = 4$. Each scheme has 100 replicates.

Our experiments with QSCL show that adding quadratic terms to Algorithm 2 can make controlling column-wise correlations less effective when $n$ is not much larger than $p$. We suggest using QCSL when $p < n/2$, but one can always examine the correlation criteria in Figure 4 first to determine whether CSL2 or QCSL should be used.

## 5. Numerical illustration

In this section, we provide numerical examples to demonstrate variance reduction under SL, CSL1, CSL2, and QCSL for the estimator of the expected output $\mu = E[f(\mathbf{x})]$ in Section 1. Owen (1994) and Tang (1998) provided theoretical justifications of variance reduction by controlling $\rho_{\text{rms}}$ in (3.1) and $\rho_{\text{rmq}}$ in (4.1).

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



Consider the Borehole function from (Morris et al. (1993)),

$$f(X) = \frac{2\pi T_u(H_u - H_l)}{\log(r/r_u)\left[1 + 2LT_u/\{\log(r/r_\omega)r_\omega^2 K_\omega\} + T_u/T_l\right]}, \qquad (5.1)$$

where the eight input variables, after appropriate scaling, lie in $(0, 1]^8$. Following (1.1), let $\hat{\mu}_{(r)}$ denote the estimator based on $\mathbf{X}_{(r)}$ and let $\hat{\mu} = t^{-1}\sum_{r=1}^{t}\hat{\mu}_{(r)}$ denote the estimator based on $\mathbf{X}$. We used $\hat{\mu}_{(1)}$ represent each slice as slices are exchangeable. For all sampling schemes, we computed $\hat{\mu}_{(1)}$ and $\hat{\mu}$ 1000 times with $n = 20$, $p = 8$, and $t = 4$. We approximated the true expected output of (5.1) at 77.652 based on a sample of 1,000,000 runs from a Latin hypercube design in (1.2).

We found that CSL1, CSL2, and QCSL provided more variance reduction than SL by controlling the column-wise correlations, while CSL2 outperformed CSL1 due to designs under CSL2, in general, have smaller root mean square correlations. Among all schemes, QCSL provided the most variance reduction in $\hat{\mu}$ with controlled quadratic canonical correlations. Schemes based on the RGS algorithm all produced estimators with little bias. Root mean square errors (RMSE) of $\hat{\mu}_{(1)}$ and $\hat{\mu}$ are given in Table 1.

## 6. Discussion

We have proposed an algorithm to control the correlations in a sliced

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



Table 1: Root mean square errors (RMSE) of $\hat{\mu}_{(1)}$ and $\hat{\mu}$ for the Borehole function with $n = 20$, $p = 8$, and $t = 4$ over the 1000 replicates

| | SL | CSL1 | CSL2 | QCSL |
|---|---|---|---|---|
| $\hat{\mu}_{(1)}$ | 2.159 | 0.644 | 0.431 | 0.441 |
| $\hat{\mu}$ | 1.013 | 0.213 | 0.185 | 0.121 |

Latin hypercube design such that the column-wise correlations in each slice and in the entire design are reduced simultaneously. The algorithm can be seen as first applying the ranked Gram-Schmidt algorithm in each slice and then exchanging elements across different slices to further improve the correlations.

When the number of runs in each slice is considerably larger than twice the number of columns, one can modify our Algorithm 2 with the method of Tang (1998) to control the quadratic canonical correlations in each slice. In cases where the number of runs in each slice is limited, it can be extended to control the quadratic canonical correlations for the entire design without sacrificing the performance in controlling the column-wise correlations in each slice. Such ideas can be generalized to control higher order polynomial canonical correlations.

We compared the designs generated under our sampling schemes with the second-order orthogonal sliced Latin hypercube design (OSLHD) of Ba et al. (2015) to estimate the expected output of the Borehole function. If OSLH denotes a sampling scheme that generates an OSLHD with random-

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



ized columns and slices, it requires $n = 2^{c+1}$, $p \leq 2^c$ for any $c \geq 1$ such that $t$ can be any positive integer. Table 2 reports RMSEs under SL, CSL2, QCSL, and OSLH with $n = 16$, $p = 8$, and $t = 4$. The OSLH gives the smallest RMSE of $\hat{\mu}$ because $\rho(\mathbf{x}^k, \mathbf{x}^\ell) = \rho((\mathbf{x}^k)^2, \mathbf{x}^\ell) = 0$ for $1 \leq k < \ell \leq p$, but it performs worse than CSL2 and QCSL within a single slice. We find that OSLHDs on average have the highest root mean square quadratic canonical correlations among all schemes. Our designs also have more flexible sample sizes than second-order orthogonal sliced Latin hypercube designs.

Table 2: Root mean square errors (RMSE) of $\hat{\mu}_{(1)}$ and $\hat{\mu}$ for the Borehole function with $n = 16$, $p = 8$, and $t = 4$ over the 1000 replicates

| | SL | CSL2 | QCSL | OSLH |
|---|---|---|---|---|
| $\hat{\mu}_{(1)}$ | 2.320 | 0.492 | 0.563 | 0.719 |
| $\hat{\mu}$ | 1.205 | 0.207 | 0.146 | 0.033 |

Sliced Latin hypercube designs are useful in reducing the error in problems involving multiple integration problems (Qian (2012); Zhang and Qian (2013); Chen et al. (2014a)). The algorithms proposed here can further reduce the error in those applications. For fitting second-order regression models, orthogonal sliced Latin hypercube designs (Yang et al. (2013); Ba et al. (2015)) are more desirable.

## Supplementary Materials

The proof of Proposition 1 and an example of using Algorithm 2 are

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



included in the online supplemental materials.

## Acknowledgements

The authors would like to thank the Editor, an associate editor, and referees for their helpful comments which led to the improvement of this paper. This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF1510156 and NSF Grants DMS 1055214 and DMS 1564376.

## References

Ba, S., Myers, W. R., and Breneman, W. A. (2015). Optimal sliced Latin hypercube designs. *Technometrics* **57**, 479-487.

Chen, J., Lim, C. H., Linderoth, J. T., Qian, P. Z. G., and Wright, S. J. (2014). Validating sample average approximation solutions with negatively dependent batches. *arXiv:1404.7208*.

Chen, Y., Steinberg, D. M., and Qian, P. Z. G. (2014). Maximin sliced Latin hypercube designs, with application to cross validating prediction errors. *Springer Handbook on Uncertainty Quantification, in press*.

Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications*. New York: Springer.

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



Huang, H. Z., Yang, J. F., and Liu, M. Q. (2014). Construction of sliced (nearly) orthogonal Latin hypercube designs. *Journal of Complexity* **30**, 355-365.

Iman, R. L., and Conover, W. J. (1982). A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics, Part B–Simulation and Computation* **11**, 311-334.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26**, 131-148.

Loh, W. L. (1996). On Latin hypercube sampling. *Annals of Statistics* **24**, 2058-2080.

McKay, M. D., Conover, W. J., and Beckman, R. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**, 239-245.

Morris, M. D., Mitchell, T. J., and Ylvisaker, D. (1993). Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics* **35**, 243-255.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration, and visualization. *Statistica Sinica* **2**, 439-452.

Owen, A. B. (1994). Controlling correlations in Latin hypercube samples. *Journal of the American Statistical Association* **89**, 1517-1522.

Qian, P. Z. G. (2012). Sliced Latin hypercube designs. *Journal of the American Statistical Association* **107**, 393-399.

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151

## REFERENCES

Rosenblatt, M. (1953). Remark on a multivariate transformation. *Annals of Mathematical Statistics* **23**, 470-472.

Stein, M. (1987). Large-sample properties of simulations using Latin hypercube sampling. *Technometrics* **29**, 143-151.

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

Sun, F. S., Liu, M. Q., and Lin, D. K. J. (2009). Construction of orthogonal Latin hypercube designs. *Biometrika* **96**, 971-974.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association* **88**, 1392-1397.

Tang, B. (1998). Selecting Latin hypercubes using correlation criteria. *Statistica Sinica* **8**, 965-977.

Yang, X., Chen, H., and Liu, M. Q. (2014). Resolvable orthogonal array-based uniform sliced Latin hypercube designs. *Statistics&Probability Letters* **93** 109-115.

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

Yin, Y. H., Lin, D. K. J., and Liu, M. Q. (2014). Sliced Latin hypercube designs via orthogonal arrays. *Journal of Statistical Planning and Inference* **149**, 162-171.

Ye, K. Q. (1998). Orthogonal column Latin hypercubes and their application in computer

Statistica Sinica: Preprint
doi:10.5705/ss.202016.0151



experiments. *Journal of the American Statistical Association* **93**, 1430-1439.

Zhang, Q., and Qian, P. Z. G. (2013). Designs for crossvalidating approximation models. *Biometrika* **100**, 997-1004.

Wells Fargo, Charlotte, NC 28202

E-mail: jiajie.chen@wellsfago.com

Department of Statistics, The University of Wisconsin – Madison, Madison, WI 53706

E-mail: peter.qian@wisc.edu