Journal of Statistical Planning and Inference 149 (2014) 162–171

Contents lists available at ScienceDirect

Journal of Statistical Planning and Inference

journal homepage: www.elsevier.com/locate/jspi

# Sliced Latin hypercube designs via orthogonal arrays

Yuhui Yin [a], Dennis K.J. Lin [b], Min-Qian Liu [a,*]

[a] LPMC and Institute of Statistics, Nankai University, Tianjin 300071, China
[b] Department of Statistics, The Pennsylvania State University, University Park, PA 16802, USA

ARTICLE INFO

*Article history:*
Received 18 December 2013
Received in revised form
12 February 2014
Accepted 17 February 2014
Available online 25 February 2014

*Keywords:*
Computer experiment
Latin hypercube design
Orthogonal array
Space-filling design

ABSTRACT

Computer experiments are becoming increasingly popular in studying complex real world systems. A special class of sliced Latin hypercube design is proposed in this paper. Such designs are particularly suitable for computer experiments with both qualitative and quantitative factors, multi-fidelity computer experiments, cross-validation and data pooling. The resulting sliced Latin hypercube designs possess a desirable sliced structure and have an attractive low-dimensional uniformity. Meanwhile within each slice, it is also a Latin hypercube design with the same low-dimensional uniformity. The new sliced Latin hypercube designs can be constructed via both symmetric and asymmetric orthogonal arrays. The same desirable properties are possessed, although the uniformity may be differed. The construction methods are easy to implement, and unlike the existing methods, the resulting designs are very flexible in run sizes and numbers of factors. A detailed comparison with existing designs is made.

© 2014 Elsevier B.V. All rights reserved.

## 1. Introduction

Computer experiments are probably the most effective approach to probe the complex real world systems. This is especially true when the corresponding physical experiments are costly. One particularly important problem is the study of multi-fidelity computer experiments. A multi-fidelity computer experiment is a large, expensive computer experiment which can be executed at various degrees of fidelity. The main desirable property of a multi-fidelity computer experiment lies on whether it can effectively increase the accuracy of predictors with limited cost (Kennedy and O'Hagan, 2000; Qian et al., 2006; Qian and Wu, 2008). Another problem is the study of computer experiments with both qualitative and quantitative factors. For example, Schmidt et al. (2005) demonstrated a data center computer experiment which involves qualitative factors (such as diffuser location and hot-air return-vent location) and quantitative factors (such as rack power and diffuser flow rate). These computer experiments call for new experimental designs, which should possess a low-dimensional uniformity not only as a whole design but also within each slice.

There are many recent works in multi-fidelity computer experiments. Qian et al. (2009b) constructed nested space-filling designs for computer experiments with two levels of accuracy by making use of Galois fields and orthogonal arrays. Qian et al. (2009a) and Sun et al. (2013) constructed nested space-filling designs from nested orthogonal arrays and nested difference matrices. Qian (2009) proposed a novel approach to constructing nested Latin hypercube designs by defining a nested permutation. The method works for any number of factors, but the resulting designs do not guarantee (two or higher-dimensional) uniformity. Qian and Ai (2010) constructed a nested lattice sampling scheme by randomizing nested

* Corresponding author.
*E-mail address:* mqliu@nankai.edu.cn (M.-Q. Liu).

http://dx.doi.org/10.1016/j.jspi.2014.02.008
0378-3758 © 2014 Elsevier B.V. All rights reserved.

orthogonal arrays with nested permutations. Haaland and Qian (2010) proposed an approach to constructing nested space-filling designs based on existing $(t, s)$-sequences for multi-fidelity computer experiments. He and Qian (2011) proposed two methods for constructing nested Latin hypercube designs. Li and Qian (2013) proposed several methods for constructing nested (nearly) orthogonal designs with two layers by exploiting nesting in various discrete structures such as fractional factorial designs, Hadamard matrices, permutation matrices and rotation matrices. Yang et al. (2014) presented some methods for constructing nested orthogonal Latin hypercube designs using a special type of orthogonal design proposed by Yang and Liu (2012).

There are also recent work in computer experiments with both qualitative and quantitative factors. Qian and Wu (2009) proposed a sliced space-filling design—a Latin hypercube design for the quantitative factors, and then partition the design into groups corresponding to different level combinations of the qualitative factors. Qian (2012) proposed a Latin hypercube design that can be partitioned into slices each of which constitutes a smaller Latin hypercube design. Xu et al. (2011) constructed sliced space-filling designs based on Sudoku structure. Yang et al. (2013) defined a novel type of design—a sliced orthogonal Latin hypercube design, and proposed several construction methods. Recently, Huang et al. (in press) proposed a method to construct sliced orthogonal and nearly orthogonal Latin hypercube designs.

In this paper, a special class of sliced Latin hypercube designs is constructed. Such designs are suitable for computer experiments with both qualitative and quantitative factors, multi-fidelity computer experiments, cross-validation and data pooling. The resulting sliced Latin hypercube designs along with their slices all inherit the $r$-dimensional uniformity of the orthogonal arrays with strength $r$. Moreover, they have more flexible run sizes and the numbers of factors. The paper is organized as follows. In Section 2, a new class of sliced Latin hypercube designs is constructed based on symmetric orthogonal arrays. In Section 3, another class of sliced Latin hypercube designs is constructed based on asymmetric orthogonal arrays. Some comparisons and concluding remarks are given in Section 4. All the proofs are provided in the Appendix.

## 2. Sliced Latin hypercube designs via symmetric orthogonal arrays

Before proposing the method for constructing sliced Latin hypercube designs using symmetric orthogonal arrays, let us first present some necessary definitions and notations.

Let $A=(a_{ij})$ be an $n\times d$ matrix with each column consisting of a permutation of $\{1,\ldots,n\}$, and $u_{ij}$'s are independent U[0, 1) random variables, for $i=1,\ldots,n$ and $j=1,\ldots,d$. Then an $n\times d$ Latin hypercube design $D=(d_{ij})$ is defined through $d_{ij}=(a_{ij}-u_{ij})/n$. A Latin hypercube design $D=(D_1',\ldots,D_k')'$ is called a *sliced Latin hypercube design* with $k$ slices $D_1,\ldots,D_k$, if each slice is also a Latin hypercube design. While for $v$ Latin hypercube designs $D_1,\ldots,D_v$, if all the rows of $D_{i-1}$ are taken from $D_i$, i.e., $D_{i-1}$ is nested within $D_i$, for $i=2,\ldots,v$, then $(D_1,\ldots,D_v)$ is called a *nested Latin hypercube design* with $v$ layers. For a sliced Latin hypercube design $D=(D_1',\ldots,D_k')'$, it is obvious that $D_i$ is nested within $D$ for any $i$, so a sliced Latin hypercube design is also a nested design with at least two layers. Thus, designs with a sliced structure are also suitable for computer experiments with multiple levels of accuracy.

An asymmetric orthogonal array $OA(n,s_1^{d_1}\ldots s_m^{d_m},r)$ is a matrix of size $n\times d$, in which the first $d_1$ columns have symbols from $\{0,\ldots,s_1-1\}$, the next $d_2$ columns have symbols from $\{0,\ldots,s_2-1\}$, and so on, with the property that in any $n\times r$ submatrix every possible $r$-tuple occurs an equal number of times as a row, where $d=d_1+\cdots+d_m$ is the total number of factors and $r$ is called the strength of this array. When all $s_i$'s are equal to $s$, the above asymmetric orthogonal array becomes a symmetric one, denoted by $OA(n,s^d,r)$, where each $n\times r$ submatrix contains all possible $1\times r$ row vectors with the same frequency, say $\lambda$, which is called the index of the array. Clearly in this case, $n=\lambda s^r$. For a matrix $A=(a_{ij})$ and a number $z$, $A\oplus z$ denotes their Kronecker sum, i.e., $A\oplus z=(a_{ij}+z)$. Let $N=kn$ with $n$ and $k$ being positive integers, and $Z_N$ be the set $\{1,\ldots,N\}$. A sliced permutation matrix SPM$(n,k)$ on $Z_N$ is defined to be an $n\times k$ matrix with each element of $Z_N$ appearing precisely once and each column of $\lceil \mathrm{SPM}(n,k)/k\rceil$ forming a permutation on $Z_n$ (cf. Qian, 2012).

For the construction of sliced Latin hypercube designs using orthogonal arrays, further assume $n=ts$ with $s$ and $t$ being positive integers, and define an $s$-level sliced permutation matrix $M(n,k,s)$ on $Z_N$ to be a sliced permutation matrix SPM$(n,k)$ formed by a row permutation of $s$ $t\times k$ matrices $G_1,\ldots,G_s$ (i.e., permuting the order of these $s$ matrices and combining them together row by row), where $G_i\oplus(1-i)tk$ is a sliced permutation matrix SPM$(t,k)$ defined on $Z_{tk}$, $i=1,\ldots,s$. An $M(n,k,s)$ can be generated by the following algorithm.

**Algorithm A.** Construction of an $M(n,k,s)$ matrix.

*Step* 1: Divide the elements of $Z_N$ into $n$ groups, $g_1,\ldots,g_n$, where

$$g_l=\{a\in Z_N\mid\lceil a/k\rceil=l\},\quad l=1,\ldots,n.$$

Note: for a real number $z$, $\lceil z\rceil$ denotes the smallest integer greater than or equal to $z$, and $\lfloor z\rfloor$ denotes the largest integer less than or equal to $z$.

*Step* 2: For $i=1,\ldots,t$, fill up the $i$th row of a $t\times k$ empty matrix $G_u$ with a uniform permutation on $g_{i+t(u-1)}$. A total of $s$ matrices $G_u$ for $u=1,\ldots,s$ will be resulted.

*Step* 3: For $j=1,\ldots,k$, randomly shuffle the entries in the $j$th column of $G_u$, $u=1,\ldots,s$.

*Step* 4: Randomly row juxtapose $G_1, \ldots, G_s$, a matrix $G = (G_{i1}', \ldots, G_{is}')'$ is obtained, where $G_{i1}, \ldots, G_{is}$ is a random permutation of $G_1, \ldots, G_s$. Call $G_{iu}$ the $u$th submatrix of $G$.

**Example 1.** Let $n=9$, $k=3$, $s=3$ and $N=27$, an $M(9,3,3)$ can be constructed as follows. First, divide $Z_{27}$ into 9 groups: $g_1=\{1,2,3\}, g_2=\{4,5,6\}, g_3=\{7,8,9\}, g_4=\{10,11,12\}, g_5=\{13,14,15\}, g_6=\{16,17,18\}, g_7=\{19,20,21\}, g_8=\{22,23,24\}$, and $g_9=\{25,26,27\}$. From Step 2, three matrices $G_1, G_2$ and $G_3$ may be obtained as

$$G_1=\begin{pmatrix}2&3&1\\4&6&5\\9&8&7\end{pmatrix},\quad G_2=\begin{pmatrix}10&12&11\\15&13&14\\18&17&16\end{pmatrix},\quad G_3=\begin{pmatrix}19&21&20\\24&23&22\\25&27&26\end{pmatrix}.$$

By independently randomly shuffling the entries in each column of $G_u$ (as in Step 3), $u=1,\ldots,3$, we have three new $G_1, G_2, G_3$ as

$$G_1=\begin{pmatrix}4&8&5\\2&3&7\\9&6&1\end{pmatrix},\quad G_2=\begin{pmatrix}15&12&16\\10&17&11\\18&13&14\end{pmatrix},\quad G_3=\begin{pmatrix}19&23&26\\25&27&20\\24&21&22\end{pmatrix}.$$

Finally, following Step 4, an $M(9,3,3)$ is obtained as

$$G=(G_{i1}',G_{i2}',G_{i3}')'=(G_3',G_2',G_1')'=\left(\begin{array}{ccc|ccc|ccc}19&25&24&15&10&18&4&2&9\\23&27&21&12&17&13&8&3&6\\26&20&22&16&11&14&5&7&1\end{array}\right)'.$$

**Remark 1.** In short, the key points in the construction of the permutation matrix $M(n,k,s)$ are that (i) the matrix $M(n,k,s)$ should consist of all the elements of $Z_N$ with $N=kn$ and each element appears once and only once; (ii) each submatrix of $M(n,k,s)$ should consist of $tk$ successive elements of $Z_N$ with $t=n/s$, which correspond to a sliced permutation matrix SPM $(t,k)$ on $Z_{tk}$ (up to a Kronecker sum); and (iii) the randomization in Algorithm A along with that in Algorithms B and C guarantees the statistical properties of the constructed designs in the following (cf. Lemma 1, Theorems 1 and 2).

Based on the permutation matrix $M(n,k,s)$ obtainable via Algorithm A, the following algorithm is proposed to construct sliced Latin hypercube designs from symmetric orthogonal arrays.

**Algorithm B.** Construction of sliced Latin hypercube designs via symmetric OA.

*Step* 1: Give an $OA(n,s^d,r)$, denoted by $OA_0$, randomize its rows, columns and symbols to generate a randomized orthogonal array. Independently repeat the above randomization $k$ times, a total of $k$ randomized orthogonal arrays, denoted by $OA_1, \ldots, OA_k$, can be obtained.
*Step* 2: Independently generate $M(n,k,s)$ (using Algorithm A) $d$ times, denote the outputs as $M_1, \ldots, M_d$.
*Step* 3: For $l=1,\ldots,k$ and $j=1,\ldots,d$, replace the $\alpha$'s in the $j$th column of $OA_l$ by a random permutation of the elements in the $l$th column of the $(\alpha+1)$th submatrix of $M_j$, $\alpha=0,\ldots,s-1$, $k$ new designs, denoted by $A_1, \ldots, A_k$, can be obtained.
*Step* 4: Based on each $A_l=(a_{ij})$, $l=1,\ldots,k$, an $n\times d$ design $D_l=(d_{ij})$ is generated by

$$d_{ij}=\frac{a_{ij}-u_{ij}}{N},\quad i=1,\ldots,n,\ j=1,\ldots,d,$$

where the $u_{ij}$'s are independent U[0, 1) random variables.
*Step* 5: Let $D=(D_1', \ldots, D_k')'$, which is the row juxtaposition of $D_l$'s.

**Remark 2.** In Algorithm B, the elements in the $(\alpha+1)$th submatrix of $M_j$ are used to replace the $\alpha$'s in all the $j$th columns of the $k$ randomize orthogonal arrays, and the elements in $M_j$ finally fill up the $j$th column of the resulting design $D$. As pointed out in Remark 1, the randomization here can ensure that the resulting designs have good statistical properties.

Lemma 1 and Theorem 1 below show the theoretical properties of the design $D$ constructed in Algorithm B.

**Lemma 1.** *Each $D_l$ constructed by Algorithm* B *is statistically equivalent to an $n\times d$ OA-based Latin hypercube design, for $l=1,\ldots,k$.*

**Theorem 1.** *For the design $D=(D_1', \ldots, D_k')'$ constructed by Algorithm* B*, we have*

(i) *$D$ is a Latin hypercube design with $k$ slices $D_l$ for $l=1,\ldots,k$.*
(ii) *$D$ and $D_l$'s achieve the same uniformity on $\underbrace{s\times\cdots\times s}_{r}$ grids, for $l=1,\ldots,k$.*
(iii) *For any $l=1,\ldots,k$, $(D_l, D)$ is a nested Latin hypercube design with two layers.*

An illustrative example is given below.

**Example 2.** Give an $OA(9,3^4,2)$, denoted by $OA_0$, where

$$OA_0=\begin{pmatrix}0&0&0&1&1&1&2&2&2\\0&1&2&0&1&2&0&1&2\\0&1&2&1&2&0&2&0&1\\0&2&1&1&0&2&2&1&0\end{pmatrix}'.$$

We now construct a 27-run sliced Latin hypercube design with 3 slices, i.e., $k=3$ and $N=27$.

*Step* 1: We randomize the rows, columns and symbols of $OA_0$, and may get three randomized orthogonal arrays as

$$OA_1=\begin{pmatrix}2&0&1&1&0&2&0&2&1\\1&0&2&0&1&2&2&0&1\\1&2&0&1&0&2&1&0&2\\0&0&0&2&2&2&1&1&1\end{pmatrix}',\quad OA_2=\begin{pmatrix}1&2&2&1&0&0&0&1&2\\1&1&0&0&1&2&0&2&2\\0&1&2&1&2&1&0&2&0\\0&2&0&1&1&0&2&2&1\end{pmatrix}',$$

$$OA_3=\begin{pmatrix}0&0&2&1&0&1&1&2&2\\1&0&2&2&2&1&0&0&1\\0&2&2&0&1&2&1&0&1\\1&0&1&0&2&2&1&2&0\end{pmatrix}'.$$

*Step* 2: We generate four $M(9,3,3)$'s as

$$M_1=\begin{pmatrix}14&10&16\\11&18&12\\17&13&15\\19&20&22\\26&24&21\\23&25&27\\1&2&6\\9&7&3\\4&5&8\end{pmatrix},\quad M_2=\begin{pmatrix}17&16&18\\10&12&15\\13&14&11\\26&23&24\\22&25&21\\20&19&27\\4&6&5\\3&1&8\\9&7&2\end{pmatrix},\quad M_3=\begin{pmatrix}13&10&17\\18&16&15\\12&14&11\\26&25&27\\22&24&21\\20&19&23\\1&4&3\\6&7&9\\8&2&5\end{pmatrix},\quad M_4=\begin{pmatrix}6&5&4\\7&3&9\\2&8&1\\24&21&19\\20&22&26\\27&25&23\\15&10&13\\11&14&16\\18&17&12\end{pmatrix}.$$

*Step* 3: Three matrices are obtained below:

$$A_1=\begin{pmatrix}4&22&20&2\\17&10&1&6\\23&4&13&7\\19&17&22&18\\14&26&12&11\\9&3&6&15\\11&9&26&20\\1&13&18&24\\26&20&8&27\end{pmatrix},\quad A_2=\begin{pmatrix}24&25&10&5\\2&19&19&17\\7&12&4&3\\25&16&25&25\\18&23&7&21\\10&7&24&8\\13&14&14&14\\20&6&2&10\\5&1&16&22\end{pmatrix},\quad A_3=\begin{pmatrix}12&24&15&19\\15&11&9&1\\8&8&5&23\\22&5&11&4\\16&2&23&16\\27&27&3&12\\21&18&27&26\\6&15&17&13\\3&21&21&9\end{pmatrix}.$$

*Step* 4: Based on $A_1,A_2$ and $A_3$, we may get $D_1,D_2$ and $D_3$ as

$$D_1=\begin{pmatrix}0.1214&0.6056&0.8168&0.6742&0.5104&0.3179&0.3959&0.0004&0.9284\\0.7780&0.3451&0.1254&0.6059&0.9427&0.1042&0.2984&0.4559&0.7360\\0.7273&0.0368&0.4525&0.8082&0.4423&0.2199&0.9266&0.6577&0.2709\\0.0470&0.1980&0.2591&0.6471&0.3856&0.5528&0.7202&0.8589&0.9890\end{pmatrix}',$$

$$D_2=\begin{pmatrix}0.8649&0.0496&0.2331&0.9107&0.6408&0.3593&0.4707&0.7335&0.1846\\0.9087&0.6914&0.4356&0.5769&0.8405&0.2223&0.4929&0.1860&0.0133\\0.3362&0.6860&0.1216&0.8906&0.2285&0.8726&0.5003&0.0605&0.5653\\0.1762&0.6123&0.0791&0.8933&0.7418&0.2961&0.5154&0.3652&0.8138\end{pmatrix}',$$

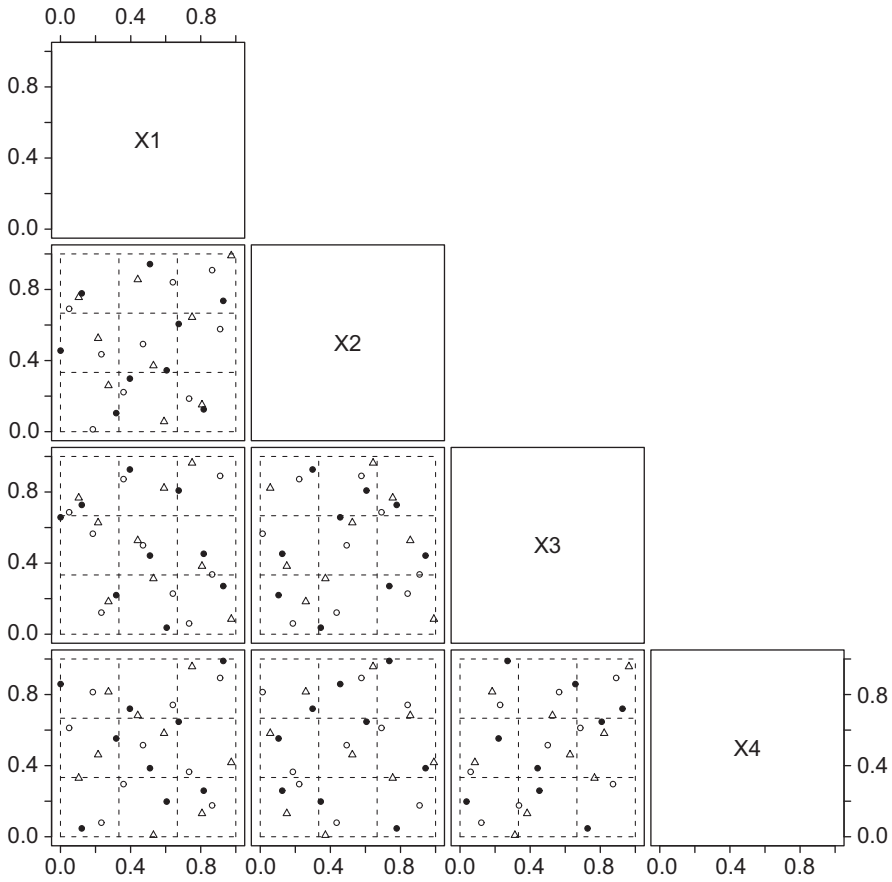


**Fig. 1.** Bivariate projections among columns of $D$.

$$D_3 = \begin{pmatrix} 0.4407 & 0.5302 & 0.2737 & 0.8072 & 0.5908 & 0.9743 & 0.7506 & 0.2147 & 0.1041 \\ 0.8558 & 0.3716 & 0.2597 & 0.1514 & 0.0570 & 0.9904 & 0.6436 & 0.5252 & 0.7556 \\ 0.5273 & 0.3130 & 0.1834 & 0.3828 & 0.8222 & 0.0847 & 0.9633 & 0.6275 & 0.7670 \\ 0.6821 & 0.0083 & 0.8154 & 0.1310 & 0.5819 & 0.4178 & 0.9573 & 0.4615 & 0.3300 \end{pmatrix}'.$$

*Step* 5: A sliced Latin hypercube design with three slices can be obtained by $D = (D_1', D_2', D_3')'$.

The bivariate projections of $D$ are shown in Fig. 1, where the markers "•", "○" and "△" denote the points from the slices $D_1$, $D_2$ and $D_3$, respectively. The symbol "X1" denotes the 1st column of $D_i$, "X2" denotes the 2nd column of $D_i$, and so on. From this figure, it can be seen that the points in the bivariate projections of each $D_i$ achieve uniformity on $3 \times 3$ grids and the points in the bivariate projections of $D$ also achieve the similar uniformity.

**Remark 3.** In Example 2, the parent orthogonal array has index $\lambda = 1$, so the slice $D_l$ can have maximum uniformity on $3 \times 3$ grids for $l = 1, 2, 3$. This implies that one and only one point lies in each grid. Thus the number of factors is limited. In fact, a variety of sliced Latin hypercube designs with more flexible run sizes and factor numbers can be obtained by taking the advantage of orthogonal arrays. For example, an $OA(18, 3^7, 2)$ can be used to generate sliced Latin hypercube designs with different run sizes and numbers of factors.

## 3. Sliced Latin hypercube designs via asymmetric orthogonal arrays

In this section, sliced Latin hypercube designs are constructed via asymmetric orthogonal arrays. Such designs have a sliced structure and attractive low-dimensional uniformity. Different from those sliced designs based on symmetric orthogonal arrays, such Latin hypercube designs can have different degrees of uniformity for different combinations of factors.

**Algorithm C.** Construction of sliced Latin hypercube designs via asymmetric OA.

*Step* 1: Give an $OA(n, s_1^{d_1} \ldots s_m^{d_m}, r)$, denoted by $OA_0$, randomize its rows, then randomize its columns in each group (columns which have the same number of levels), and its symbols in each column to generate a randomized orthogonal array. Obtain $k$ such arrays, $OA_1, \ldots, OA_k$, by independently repeating the above randomization $k$ times.
*Step* 2: Independently generate $M(n, k, s_i)$ $d_i$ times, denote the outputs as $M_1^{s_i}, \ldots, M_{d_i}^{s_i}$, $i = 1, \ldots, m$.
*Step* 3: For $l = 1,\ldots,k$, $i = 1, \ldots, m$ and $j = 1, \ldots, d_i$, replace the $\alpha$'s in the $j$th $s_i$-level column of $OA_l$ by a random permutation of the elements in the $l$th column of the $(\alpha+1)$th submatrix of $M_j^{s_i}$, $\alpha = 0, \ldots, s_i - 1$, call these $k$ designs as $A_1, \ldots, A_k$.
*Step* 4: Same as Step 4 of Algorithm B to generate design $D_l$, for $l = 1, \ldots, k$.
*Step* 5: Obtain design $D = (D_1', \ldots, D_k')'$, which is a row juxtaposition of $D_l$'s.

**Theorem 2.** *For the design $D = (D_1', \ldots, D_k')'$ constructed by Algorithm C, we have*

(i) *$D$ is a Latin hypercube design with $k$ slices $D_l$ for $l = 1,\ldots,k$.*
(ii) *$D$ and $D_l$'s achieve $r$-dimensional uniformity; namely, if they are projected onto $r$ columns corresponding to factors with $s_{i1}, \ldots, s_{ir}$ levels in the parent orthogonal array, then they achieve uniformity on $s_{i1} \times \cdots \times s_{ir}$ grids.*
(iii) *For any $l = 1, \ldots, k$, $(D_l, D)$ is a nested Latin hypercube design with two layers.*

**Example 3.** Give an $OA(16, 2^6 4^3, 2)$, denoted by $OA_0$, where

$$OA_0 = \begin{pmatrix} 0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0 \\ 0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1 \\ 0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0 \\ 0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1 \\ 0&0&1&1&1&1&0&0&1&1&0&0&0&0&1&1 \\ 0&1&0&1&1&0&1&0&1&0&1&0&0&1&0&1 \\ 0&0&2&2&0&0&2&2&3&3&1&1&3&3&1&1 \\ 0&2&0&2&3&1&3&1&0&2&0&2&3&1&3&1 \\ 0&2&2&0&3&1&1&3&3&1&1&3&0&2&2&0 \end{pmatrix}'.$$

Let $k=2$ and $N=32$. A sliced Latin hypercube design can be constructed as follows:

*Step* 1: Generate two randomized orthogonal arrays $OA_1$ and $OA_2$ as

$$OA_1 = \begin{pmatrix} 0&1&0&0&0&1&1&0&1&1&1&0&1&0&1&0 \\ 1&1&0&1&1&1&1&0&1&0&0&1&0&0&0&0 \\ 0&1&0&0&1&0&1&1&0&0&1&1&1&0&0&1 \\ 1&0&1&0&0&0&1&0&1&1&0&1&1&0&0&1 \\ 1&1&1&0&0&1&0&0&0&0&1&1&0&0&1&1 \\ 1&0&0&1&0&1&0&1&1&0&1&0&1&0&0&1 \\ 1&0&3&2&1&3&3&3&0&2&2&2&1&0&1&0 \\ 1&1&2&3&2&0&3&1&2&1&2&0&0&0&3&3 \\ 0&1&1&1&3&3&0&2&2&3&0&2&1&0&2&3 \end{pmatrix}',$$

$$OA_2 = \begin{pmatrix} 0&1&1&1&0&0&1&0&0&1&1&0&0&0&1&1 \\ 0&1&0&0&1&1&0&0&0&1&1&1&0&1&1&0 \\ 1&1&1&1&1&1&0&0&0&0&0&0&1&0&1&0 \\ 0&1&0&1&0&1&1&0&1&1&0&1&1&0&0&0 \\ 0&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1 \\ 1&0&1&1&0&0&0&0&0&1&1&1&1&1&0&0 \\ 3&3&2&1&1&2&2&0&3&0&3&1&0&2&0&1 \\ 1&3&2&0&2&0&1&0&2&2&0&1&3&3&1&3 \\ 2&0&0&1&3&2&3&0&1&2&3&0&3&1&1&2 \end{pmatrix}'.$$

*Step* 2: Generate six $M(16,2,2)$'s and three $M(16,2,4)$'s as

$$\begin{pmatrix} M_1^{2'} \\ M_2^{2'} \\ M_3^{3'} \\ M_4^{2'} \\ M_5^{2'} \\ M_6^{2'} \end{pmatrix}' = \left(\begin{array}{cc|cc|cc|cc|cc|cc} 4 & 13 & 21 & 29 & 28 & 19 & 19 & 32 & 15 & 3 & 3 & 13 \\ 8 & 16 & 25 & 27 & 22 & 23 & 25 & 21 & 7 & 1 & 9 & 4 \\ 2 & 12 & 32 & 26 & 18 & 21 & 30 & 17 & 4 & 16 & 5 & 6 \\ 11 & 1 & 28 & 24 & 24 & 26 & 27 & 28 & 10 & 13 & 15 & 2 \\ 14 & 6 & 20 & 18 & 32 & 29 & 22 & 20 & 5 & 8 & 14 & 12 \\ 15 & 9 & 23 & 31 & 25 & 31 & 31 & 29 & 11 & 12 & 1 & 16 \\ 10 & 7 & 30 & 22 & 20 & 17 & 23 & 24 & 2 & 9 & 7 & 8 \\ 5 & 3 & 17 & 19 & 30 & 27 & 18 & 26 & 14 & 6 & 11 & 10 \\ 21 & 18 & 16 & 15 & 12 & 16 & 14 & 7 & 18 & 28 & 23 & 31 \\ 23 & 29 & 13 & 4 & 3 & 4 & 11 & 13 & 22 & 24 & 17 & 29 \\ 26 & 25 & 9 & 7 & 15 & 8 & 8 & 10 & 20 & 17 & 30 & 20 \\ 17 & 20 & 12 & 10 & 10 & 6 & 6 & 5 & 23 & 25 & 32 & 25 \\ 30 & 27 & 2 & 1 & 7 & 13 & 9 & 16 & 32 & 21 & 27 & 24 \\ 28 & 24 & 5 & 6 & 1 & 9 & 3 & 12 & 30 & 29 & 22 & 28 \\ 19 & 32 & 3 & 11 & 14 & 11 & 2 & 4 & 27 & 31 & 26 & 18 \\ 31 & 22 & 8 & 14 & 5 & 2 & 15 & 1 & 26 & 19 & 19 & 21 \end{array}\right), \quad \begin{pmatrix} M_1^{4'} \\ M_2^{4'} \\ M_3^{4'} \end{pmatrix}' = \left(\begin{array}{cc|cc|cc} 22 & 24 & 12 & 16 & 25 & 26 \\ 19 & 20 & 10 & 9 & 31 & 30 \\ 23 & 18 & 15 & 13 & 29 & 32 \\ 17 & 21 & 14 & 11 & 27 & 28 \\ 2 & 7 & 5 & 6 & 9 & 11 \\ 4 & 6 & 8 & 7 & 14 & 13 \\ 8 & 1 & 2 & 1 & 16 & 10 \\ 5 & 3 & 4 & 3 & 12 & 15 \\ 30 & 29 & 29 & 31 & 19 & 22 \\ 27 & 28 & 28 & 30 & 17 & 20 \\ 26 & 31 & 32 & 26 & 21 & 18 \\ 32 & 25 & 25 & 27 & 23 & 24 \\ 16 & 13 & 23 & 21 & 5 & 2 \\ 11 & 15 & 19 & 20 & 1 & 7 \\ 14 & 12 & 18 & 24 & 3 & 6 \\ 10 & 9 & 22 & 17 & 8 & 4 \end{array}\right).$$

*Step* 3: Designs $A_1$ and $A_2$ can be obtained as

$$A_1 = \begin{pmatrix} 4 & 23 & 2 & 14 & 11 & 26 & 30 & 15 & 21 & 19 & 17 & 8 & 28 & 5 & 31 & 10 \\ 3 & 16 & 23 & 8 & 5 & 13 & 12 & 30 & 9 & 28 & 21 & 2 & 25 & 17 & 20 & 32 \\ 25 & 14 & 28 & 18 & 15 & 20 & 5 & 10 & 22 & 32 & 3 & 7 & 1 & 24 & 30 & 12 \\ 9 & 23 & 6 & 25 & 19 & 27 & 3 & 31 & 14 & 2 & 18 & 15 & 11 & 22 & 30 & 8 \\ 32 & 18 & 23 & 15 & 4 & 27 & 11 & 2 & 10 & 5 & 30 & 22 & 14 & 7 & 20 & 26 \\ 27 & 11 & 14 & 17 & 7 & 32 & 3 & 26 & 23 & 1 & 22 & 15 & 30 & 5 & 9 & 19 \\ 8 & 17 & 10 & 27 & 4 & 16 & 14 & 11 & 19 & 32 & 26 & 30 & 5 & 23 & 2 & 22 \\ 8 & 2 & 29 & 23 & 28 & 14 & 19 & 5 & 25 & 4 & 32 & 12 & 10 & 15 & 22 & 18 \\ 31 & 9 & 12 & 16 & 5 & 8 & 25 & 21 & 17 & 3 & 29 & 23 & 14 & 27 & 19 & 1 \end{pmatrix}',$$

$$A_2 = \begin{pmatrix} 3 & 29 & 32 & 20 & 16 & 7 & 24 & 1 & 6 & 18 & 27 & 13 & 9 & 12 & 22 & 25 \\ 31 & 15 & 24 & 22 & 11 & 6 & 27 & 26 & 19 & 10 & 4 & 7 & 18 & 1 & 14 & 29 \\ 11 & 4 & 13 & 6 & 2 & 16 & 19 & 31 & 27 & 23 & 29 & 17 & 9 & 21 & 8 & 26 \\ 20 & 12 & 26 & 16 & 28 & 10 & 13 & 17 & 5 & 4 & 24 & 7 & 1 & 21 & 32 & 29 \\ 16 & 9 & 21 & 6 & 13 & 25 & 3 & 8 & 28 & 1 & 29 & 31 & 24 & 12 & 17 & 19 \\ 31 & 16 & 21 & 20 & 4 & 13 & 12 & 8 & 2 & 18 & 28 & 24 & 29 & 25 & 10 & 6 \\ 9 & 12 & 31 & 1 & 7 & 29 & 28 & 20 & 15 & 18 & 13 & 6 & 21 & 25 & 24 & 3 \\ 3 & 24 & 31 & 16 & 26 & 13 & 1 & 9 & 30 & 27 & 11 & 7 & 20 & 17 & 6 & 21 \\ 22 & 32 & 26 & 11 & 7 & 24 & 4 & 28 & 13 & 20 & 6 & 30 & 2 & 10 & 15 & 18 \end{pmatrix}'.$$

Steps 4 and 5 of Algorithm C, a sliced Latin hypercube design with two slices, denoted by $D = (D_1', D_2')'$, can then be obtained.

The bivariate projections of $D_1$ are shown in Fig. 2. Only those among the 1st, 2nd, 8th and 9th columns are presented, where the symbols "X1", "X2", "X8" and "X9" correspond to these four columns. From this figure, we can see that different combinations of factors have different degrees of uniformity. For example, X1 and X2 have uniformity on $2 \times 2$ grids, X8 and X9 achieve uniformity on $4 \times 4$ grids, and X2 and X8 achieve uniformity on $2 \times 4$ grids.

**Remark 4.** The uniformity of $D_l$ and $D$ is inherited from their parent asymmetric orthogonal arrays, so when a suitable parent asymmetric orthogonal array is chosen, the resulting sliced Latin hypercube designs will have favorable uniformity property. In this example, since the first four columns of the $OA(16, 2^6 4^3, 2)$ constitute a symmetric $OA(16, 2^4, 4)$, the $D_l$'s and $D$ will carry the nice uniformity property till 4-dimension when projected onto the first four columns of the original $OA(16, 2^6 4^3, 2)$.

Compared with Algorithm B, Algorithm C extends the construction method from symmetric orthogonal arrays to asymmetric orthogonal arrays. Such an extension may improve the low-dimensional uniformity of $D_l$ and $D$, depending on the combinations of factors as in the above-mentioned $OA(16, 2^6 4^3, 2)$, and can result in sliced Latin hypercube designs with more flexible run sizes and numbers of factors.

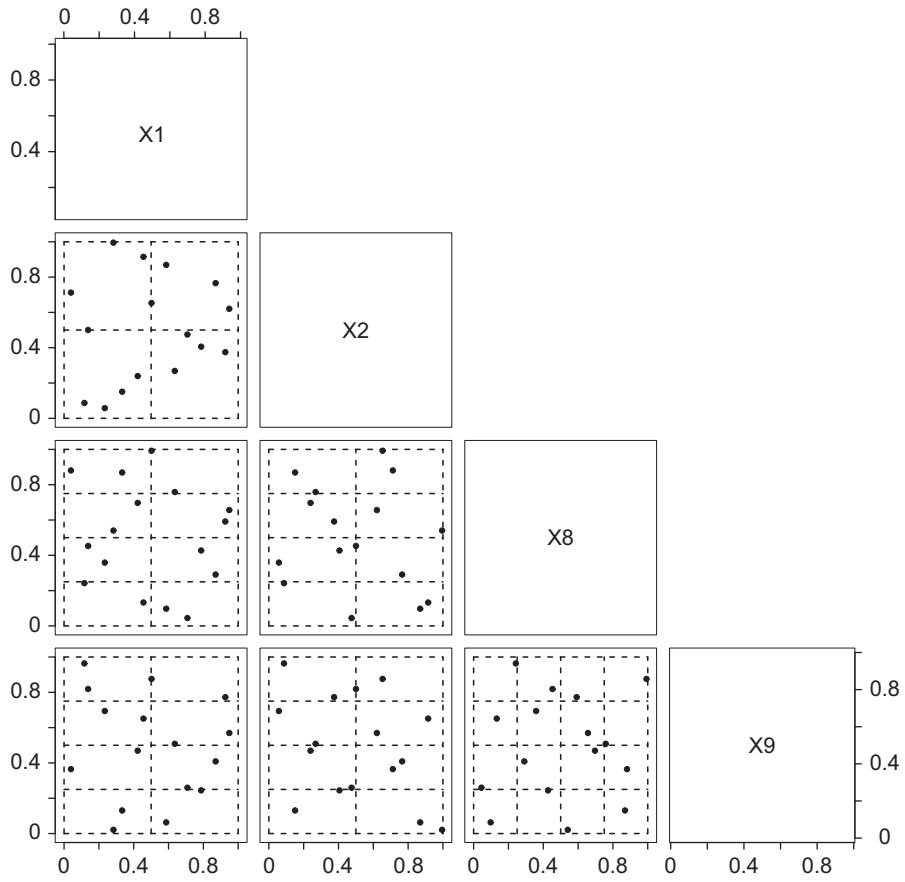


**Fig. 2.** Bivariate projections among columns of $D_1$.

**Table 1**
Comparisons of the proposed designs with the sliced designs due to Qian and Wu (2009) and Qian (2012).[a]

| Construction method | Run size | | Maximum number of factors | Number of slices | Maximum dimension of uniformity | Constraints |
|---|---|---|---|---|---|---|
| | $D$ | $D_l$ | | | | |
| Qian and Wu (2009) | $(p^{u_1})^t$ | $(p^{u_2})^t$ | $p^{u_2}+1$ | $\frac{(p^{u_1})^t}{(p^{u_2})^t}$ | $t$ for $D$, $(t-1)$ for $D_l$ | $u_2 \mid u_1$, $p^{u_2} \geq t-1 \geq 0$ |
| | $(p^{u_1})^t$ | $(p^{u_2})^t$ | $p^{u_2}+1$ | $\frac{(p^{u_1})^t}{(p^{u_2})^t}$ | $t$ for $D$, $(t-1)$ for $D_l$ | $t(u_2-1) \leq u_1-1$, $p^{u_2} \geq t-1 \geq 0$ |
| | $(p^{u_1})^k$ | $(p^{u_2})^k$ | $\frac{(p^{u_2})^k-1}{p^{u_2}-1}$ | $\frac{(p^{u_1})^k}{(p^{u_2})^k}$ | 2 | $2u_2 \leq u_1-1$ |
| | $(p^{u_1})^2$ | $(p^{u_2})^2$ | $p^{u_2}$ | $\frac{(p^{u_1})^2}{(p^{u_2})^2}$ | 2 | $u_2 \mid u_1$ |
| Qian (2012) | $N$ | $n$ | $m$ | $b$ | 1 | $N=nb$ |
| The proposed methods | $N$ | $n$ | $f$ | $k$ | $r$ | $N=nk$, the parent orthogonal array with $n$ runs, $f$ factors and strength $r$ |

[a] $p$ is a prime, $u_1 > u_2 \geq 1$, $b \geq 2$, $k \geq 2$, $t$ and $m$ are positive integers.

## 4. Comparisons and concluding remarks

Algorithm B (in Section 2) provides a method for construction of sliced Latin hypercube designs using symmetric orthogonal arrays. The resulting sliced Latin hypercube designs and their slices inherit the $r$-dimensional uniformity of the used orthogonal array of strength $r$. Algorithm C (in Section 3) provides another method for construction of sliced Latin hypercube designs using asymmetric orthogonal arrays. The resulting sliced Latin hypercube designs and their slices also

achieve the $r$-dimensional uniformity of the parent orthogonal arrays with strength $r$. Most existing orthogonal arrays can be found from the websites maintained by Dr. N.J.A. Sloane (http://neilsloane.com/oadir/index.html) and Dr. W.F. Kuhfeld (http://support.sas.com/techsup/technote/ts723.html).

Some comparisons of the proposed designs with existing sliced designs, mainly due to Qian and Wu (2009), and Qian (2012), are displayed in Table 1. In Table 1, we compare the newly constructed sliced Latin hypercube designs with the existing ones in five favorable properties: the run size, the maximum number of factors under a certain run size, the number of slices, the maximum dimension of uniformity and the constraints. It can be seen that the designs constructed by Qian and Wu (2009) have limited run sizes $N=(p^{u_1})^t$—it has to be a power of $p^{u_1}$; while the proposed sliced Latin hypercube design can be constructed for any $N$ which is a multiple of the run size of an orthogonal array ($N=nk$). The newly constructed design has a much more flexible run size and number of factors. In computer experiments, it is desirable that the whole design and its slices have one or higher-dimensional uniformity. The newly constructed sliced Latin hypercube designs have a better uniformity than the designs constructed by Qian (2012), which have only one-dimensional uniformity though have the advantage of accommodating any number of factors, $m$.

Note that the newly constructed designs are also nested Latin hypercube designs. These designs are easy to construct, and are very appropriate for computer experiments, including collective evaluations of computer models, ensembles of multiple computer models, computer experiments with qualitative and quantitative variables, cross-validation and data pooling.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 11271205), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20130031110002), and the “131” Talents Program of Tianjin. The authors thank Dr. Zhiguang Qian and a referee for their valuable comments and suggestions.

## Appendix A. Proofs

**Proof of Lemma 1.** Let $d_{ij}$ be the entry of $D_l$, for $i=1,\ldots,n$ and $j=1,\ldots,d$. From the construction of $M(n,k,s)$ in Algorithm A, $d_{ij}$ can be expressed as

$$d_{ij}=\frac{w_{ij}k}{N}-\frac{e_{ij}+u_{ij}}{N}=\frac{w_{ij}}{n}-\frac{e_{ij}+u_{ij}}{N},$$

where $w_{1j},\ldots,w_{nj}$ constitute a uniform random permutation on $Z_n$, each $e_{ij}$ is a discrete random variable with the probability mass function $\Pr(e_{ij}=c)=1/k$, for $c=0,1,\ldots,k-1$, and the $w_{ij}$, $e_{ij}$ and $u_{ij}$ are mutually independent.

Since $w_{1j},\ldots,w_{nj}$ is a uniform random permutation on $Z_n$, to prove the one-dimensional uniformity of $D_l$, it remains to verify that each $(e_{ij}+u_{ij})/N$ is a U$[0,1/n)$ random variable. For $x\in[0,1/n)$, let $\lfloor Nx\rfloor=c_0$, we have

$$\begin{aligned}&\Pr\left(\frac{e_{ij}+u_{ij}}{N}\le x\right)\\&=\frac{1}{k}\left[\sum_{c=0}^{c_0-1}\Pr(c+u_{ij}\le Nx)+\Pr(c_0+u_{ij}\le Nx)+\sum_{c=c_0+1}^{k-1}\Pr(a+u_{ij}\le Nx)\right]\\&=\frac{1}{k}[c_0+(Nx-c_0)+0]\\&=nx,\end{aligned}$$

which indicates that $(e_{ij}+u_{ij})/N\sim$ U$[0,1/n)$. Thus $D_l$ has one-dimensional uniformity.

Furthermore, from Step 3 of Algorithm B, the $t$ $\alpha$'s in the $j$th column of the parent randomized orthogonal array $OA_l$ are replaced by a permutation of $t$ successive items in the corresponding column of $M_j$. So according to Tang (1993), the design $D_l$ is statistically equivalent to an $n\times d$ OA-based Latin hypercube design. □

**Proof of Theorem 1.** (i) Since the elements in each $G_l$ are successive items, and for each column of $(OA_1',\ldots,OA_k')'$, all the $tk$ $\alpha$'s are replaced by the elements of some submatrix $G_l$, then according to Tang (1993), $D$ is an OA-based Latin hypercube design and achieves $r$-dimensional uniformity.

From Lemma 1, $D_l$ is an OA-based Latin hypercube design, $l=1,\ldots,k$. Thus $D$ is a sliced Latin hypercube design with $k$ slices.

(ii) From the construction, we know that each $OA_l$ achieves uniformity on $\underbrace{s\times\cdots\times s}_{r}$ grids, it is obvious that $(OA_1',\ldots,OA_k')'$ also achieves uniformity on $\underbrace{s\times\cdots\times s}_{r}$ grids. So $D$ and $D_l$'s achieve uniformity on $\underbrace{s\times\cdots\times s}_{r}$ grids.

(iii) The conclusion is obvious. □

**Proof of Theorem 2.** The proof is similar to that of Theorem 1 and is thus omitted. □

## References

Haaland, B., Qian, P.Z.G., 2010. An approach to constructing nested space-filling designs for multi-fidelity computer experiments. Statist. Sinica 20, 1063–1075.
He, X., Qian, P.Z.G., 2011. Nested orthogonal array-based Latin hypercube designs. Biometrika 98, 721–731.
Huang, H.Z., Yang, J.F., Liu, M.Q. Construction of sliced (nearly) orthogonal Latin hypercube designs. J. Complex., http://dx.doi.org/10.1016/j.jco.2013.10.004, in press.
Kennedy, M.C., O'Hagan, A., 2000. Predicting the output from a complex computer code when fast approximations are available. Biometrika 87, 1–13.
Li, J., Qian, P.Z.G., 2013. Construction of nested (nearly) orthogonal designs for computer experiments. Statist. Sinica 23, 451–466.
Qian, P.Z.G., 2009. Nested Latin hypercube designs. Biometrika 96, 957–970.
Qian, P.Z.G., 2012. Sliced Latin hypercube designs. J. Amer. Statist. Assoc. 107, 393–399.
Qian, P.Z.G., Ai, M.Y., 2010. Nested Lattice sampling: a new sampling scheme derived by randomizing nested orthogonal arrays. J. Amer. Statist. Assoc. 105, 1147–1155.
Qian, P.Z.G., Ai, M.Y., Wu, C.F.J., 2009a. Construction of nested space-filling designs. Ann. Statist. 37, 3616–3643.
Qian, P.Z.G., Tang, B., Wu, C.F.J., 2009b. Nested space-filling designs for computer experiments with two levels of accuracy. Statist. Sinica 19, 287–300.
Qian, Z., Seepersad, C., Joseph, R., Allen, J., Wu, C.F.J., 2006. Building surrogate models based on detailed and approximate simulations. ASME Trans.: J. Mech. Des. 128, 668–677.
Qian, P.Z.G., Wu, C.F.J., 2008. Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. Technometrics 50, 192–204.
Qian, P.Z.G., Wu, C.F.J., 2009. Sliced space-filling designs. Biometrika 96, 945–956.
Schmidt, R.R., Cruz, E.E., Iyengar, M.K., 2005. Challenges of data center thermal management. IBM J. Res. Develop. 49, 709–723.
Sun, F.S., Yin, Y.H., Liu, M.Q., 2013. Construction of nested space-filling designs using difference matrices. J. Statist. Plann. Inference 143, 160–166.
Tang, B., 1993. Orthogonal array-based Latin hypercubes. J. Amer. Statist. Assoc. 88, 1392–1397.
Xu, X., Haaland, B., Qian, P.Z.G., 2011. Sudoku-based space-filling designs. Biometrika 98, 711–720.
Yang, J.F., Lin, C.D., Qian, P.Z.G., Lin, D.K.J., 2013. Construction of sliced orthogonal Latin hypercube designs. Statist. Sinica 23, 1117–1130.
Yang, J.Y., Liu, M.Q., 2012. Construction of orthogonal and nearly orthogonal Latin hypercube designs from orthogonal designs. Statist. Sinica 22, 433–442.
Yang, J.Y., Liu, M.Q., Lin, D.K.J., 2014. Construction of nested orthogonal Latin hypercube designs. Statist. Sinica 24, 211–219.