




# Flexible sliced Latin hypercube designs with slices of different sizes

Ru Yuan<sup>1,3</sup> · Bing Guo<sup>2,3</sup> · Min-Qian Liu<sup>3</sup> 

Received: 30 August 2018 / Revised: 8 July 2019 / Published online: 18 July 2019  
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

## Abstract

Sliced Latin hypercube designs (SLHDs) are widely used in various computer experiments. Literatures concerning the construction of SLHDs are all about constructing SLHDs with slices of the same size. However, in some cases, e.g. when an experiment with multiple computer models having different complexities or a sequential experiment with varying costs in different periods is considered, SLHDs with slices of different sizes are preferable. In this paper, we propose a new class of SLHD, named the flexible SLHD, in which the whole design is a Latin hypercube design (LHD), and each slice is also an LHD when its levels being properly collapsed, the difference lies in that its slices may have different run sizes. Several methods for constructing such designs are developed. Theoretical results on the constructed designs are derived, and discussion on the slice sizes of the constructed flexible SLHDs is provided. Furthermore, an optimization algorithm is developed to improve the space-filling property of the constructed SLHDs. The newly proposed flexible SLHD is also a special nested LHD (Qian in *Biometrika* 96:957–970, 2009), each of its slice can be viewed as a small LHD nested in the whole design.

**Keywords** Computer experiment · Level expansion · Nested Latin hypercube design · Sliced Latin hypercube design

---

✉ Min-Qian Liu  
mqliu@nankai.edu.cn

<sup>1</sup> School of Statistics and Mathematics, Zhongnan University of Economics and Law, Wuhan 430073, China

<sup>2</sup> College of Mathematics, Sichuan University, Chengdu 610065, China

<sup>3</sup> School of Statistics and Data Science, LPMC & KLMDASR, Nankai University, Tianjin 300071, China

## 1 Introduction

Sliced Latin hypercube designs (SLHDs), proposed by Qian (2012), are widely used in computer experiments with qualitative and quantitative factors, model calibration, cross validation, multiple experiments, stochastic optimization and data pooling. Such a design is a special Latin hypercube design (LHD, McKay et al. 1979) that can be partitioned into smaller slices, each of which is also an LHD when its levels being properly collapsed.

Extensive work has been done in the literature on constructing and finding “good” SLHDs. Qian (2012) provided a construction method for generating SLHDs column by column independently using sliced permutation matrix. Unlike the procedure introduced by Qian (2012), Ba et al. (2015) proposed a new approach for generating SLHDs that first generates small LHDs in each slice and then arranges them together to form the SLHD. However, these two methods cannot guarantee the low correlations among the columns of each slice of the design, and among the columns of the whole design. The rationales for using designs with exact or near orthogonality can be found in Ye (1998), Steinberg and Lin (2006), Bingham et al. (2009), Sun et al. (2009, 2010, 2011), and Yang and Liu (2012) among others. In consideration of the fact that SLHDs with good orthogonality are often required, much work on this topic has been done, such as Yang et al. (2013, 2016), Huang et al. (2014), Cao and Liu (2015), and Wang et al. (2017).

Recently, Xie et al. (2014) proposed general SLHDs for computer experiments. A general SLHD has multiple layers, at each of which there are multiple LHDs that can be sliced into smaller LHDs at the next layer. Furthermore, Kong et al. (2018) constructed sliced designs with flexible sizes of slices for computer experiments.

To the best of our knowledge, the proposed SLHDs and general SLHDs are all with slices of the same size, and the flexible sliced designs proposed by Kong et al. (2018) are not always SLHDs. In practice, in some cases, for example, when an experiment with multiple computer models having different complexities or a sequential experiment with varying costs in different periods is considered, SLHDs with slices of different sizes are preferable. So we propose a new class of SLHD, named the flexible SLHD, which is an SLHD with slices of different sizes. Such designs are useful for collective and batch evaluation of computer models. Several methods for constructing flexible SLHDs are developed. Meanwhile, discussion on the slice sizes of the constructed flexible SLHDs is also provided. Furthermore, an optimization algorithm for generating uniform flexible SLHDs is proposed. The newly proposed flexible SLHD is also a special nested LHD (Qian 2009), each of its slice can be viewed as a small LHD nested in the whole design.

The rest of this paper is organized as follows. Section 2 provides relevant definitions and notation. The construction methods and properties of the newly constructed designs are developed in Sect. 3, along with some discussion on special cases of the slice sizes of the constructed flexible SLHDs. Section 4 proposes an optimization algorithm for generating uniform flexible SLHDs. Concluding remarks are given in Sect. 5.

## 2 Definitions and notation

For a matrix  $A$ , let  $A(:, j)$  be its  $j$ th column and  $A(i, :)$  be its  $i$ th row. For any real number  $r$ ,  $\lceil r \rceil$  denotes the smallest integer not smaller than  $r$ , and for any real vector or matrix  $M$ ,  $\lceil M \rceil$  is similarly defined. For a positive integer  $b$ , let  $\mathbb{Z}_b$  denotes the set  $\{1, \dots, b\}$ . Throughout, sampling a uniform permutation on  $\mathbb{Z}_N$  means randomly taking a permutation on the set, with all  $N!$  possible permutations being equally probable.

Let  $A = (a_{ij})$  be an  $N \times p$  LHD, denoted by  $\text{LHD}(N, p)$ , in which each column is a uniform permutation on  $\{1, \dots, N\}$  and all the columns are obtained independently. For positive integers  $m$  and  $s$ , an SLHD with  $N = ms$  runs,  $p$  factors and  $s$  slices, denoted by  $\text{SLHD}(m, s, p)$ , is an  $\text{LHD}(N, p)$  that can be partitioned into  $s$  slices and each slice forms a smaller  $\text{LHD}(m, p)$  (after its levels being properly collapsed, this is also needed for any of the following LHDs).

Recently, Xie et al. (2014) defined a general SLHD as follows. Given positive integers  $s$  and  $p$ , for positive integers  $m, r_1, \dots, r_s$  and  $N$  with  $N = m \prod_{j=1}^s r_j$ , an  $s$ -layer general SLHD has the following form: in the first layer, the whole design consists of  $\prod_{j=1}^s r_j$  LHDs, each of which is an  $\text{LHD}(m, p)$ ; in the  $k$ th layer, for  $k = 2, \dots, s$ , there are  $\prod_{j=k}^s r_j$  LHDs, each of which is an  $\text{LHD}(m \prod_{j=1}^{k-1} r_j, p)$  and consists of  $r_{k-1}$  LHDs from the  $(k - 1)$ th layer. The  $r_s$  LHDs in the  $s$ th layer constitute the whole design, which is an  $\text{LHD}(N, p)$ . Such a design is denoted as  $\text{GSLHD}(r_1, \dots, r_s; m; p)$  by Xie et al. (2014).

Next, we define the flexible SLHD with slices of different sizes. Given positive integers  $s$  and  $p$ , for positive integers  $m_1, \dots, m_s$ , a flexible SLHD with  $N = m_1 + \dots + m_s$  runs,  $p$  factors and  $s$  slices of sizes  $m_1, \dots, m_s$ , respectively, denoted by  $\text{FSL}((m_1, \dots, m_s), s, p)$ , is an  $\text{LHD}(N, p)$  that can be partitioned into  $s$  subarrays, these subarrays can be collapsed into smaller LHDs of sizes  $m_i \times p$ ,  $1 \leq i \leq s$ , respectively. When  $m_1 = \dots = m_s = m$ , it reduces to an  $\text{SLHD}(m, s, p)$ . For  $s = 1$ , an  $\text{FSL}((m_1, \dots, m_s), s, p)$  reduces to an  $\text{LHD}(N, p)$ . Take the following design

$$D = (D_{(1)}^T, D_{(2)}^T, D_{(3)}^T)^T = \left( \begin{array}{cc|cc|cc|cccc} 4 & 7 & 12 & 2 & 9 & 6 & 10 & 3 & 11 & 1 & 5 & 8 \\ 12 & 3 & 1 & 8 & 6 & 10 & 4 & 7 & 9 & 5 & 2 & 11 \\ 7 & 4 & 6 & 12 & 2 & 9 & 8 & 3 & 10 & 5 & 11 & 1 \end{array} \right)^T \tag{1}$$

as an example. The whole design is an  $\text{LHD}(12, 3)$ , and after the levels being collapsed according to  $\lceil D_{(1)}/6 \rceil$ ,  $\lceil D_{(2)}/3 \rceil$  and  $\lceil D_{(3)}/2 \rceil$ , respectively, its three subarrays  $D_{(1)}, D_{(2)}, D_{(3)}$  become  $\text{LHD}(2, 3)$ ,  $\text{LHD}(4, 3)$  and  $\text{LHD}(6, 3)$ , respectively, thus  $D$  is an  $\text{FSL}((2, 4, 6), 3, 3)$ . Figure 1 shows the two-dimensional projections of  $D$ , where the symbols  $(\bullet, \circ, \Delta)$  represent the points from the three slices  $D_{(1)}, D_{(2)}$  and  $D_{(3)}$  of  $D$ , respectively (each level is shifted to its left by 0.5).

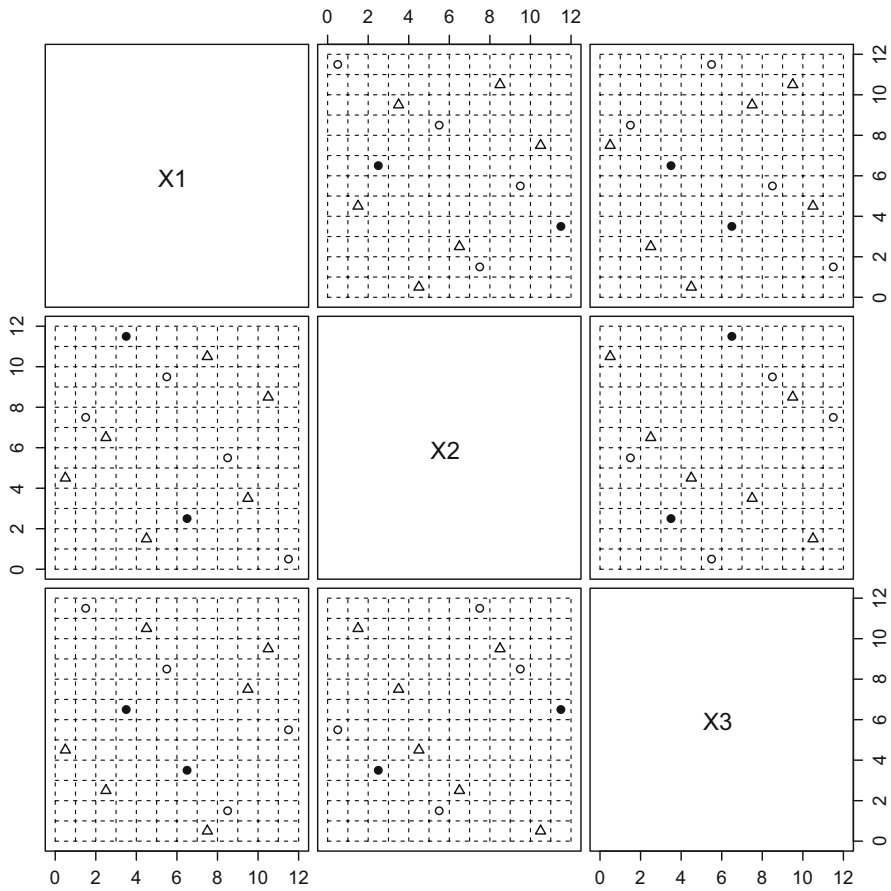


Fig. 1 Two-dimensional projections of the flexible SLHD in (1). The two “•” are from  $D_{(1)}$ , the four “○” are from  $D_{(2)}$ , and the six “△” are from  $D_{(3)}$

### 3 Construction of flexible SLHDs

In this section, we propose four methods for constructing  $FSL((m_1, \dots, m_s), s, p)$ 's with  $N = m_1 + \dots + m_s$ , starting with different types of initial designs. Without loss of generality, we assume  $m_1 \leq m_2 \leq \dots \leq m_s$ .

#### 3.1 Method I

In this subsection, we provide a construction method based on the method given in Xie et al. (2014). Assume  $m_{s+1} = N$  and  $m_i | m_{i+1}$  ( $1 \leq i \leq s$ ), where  $m_i | m_{i+1}$  means “ $m_i$  divides  $m_{i+1}$ ”, then there exist integers  $r_1, \dots, r_s$  such that  $m_{i+1} = m_i r_i = m_1 \prod_{j=1}^i r_j$ . The construction of an  $FSL((m_1, \dots, m_s), s, p)$  is as follows.

**Algorithm 1** *Step 1* Generate a GSLHD( $r_1, \dots, r_s; m_1; p$ ) using the construction method in Xie et al. (2014), denote the design as  $D_0$ .

*Step 2* Partition the rows of  $D_0$  as follows:  $D_{(1)}$  consists of the first  $m_1$  rows of  $D_0$ , for  $2 \leq i \leq s$ ,  $D_{(i)}$  consists of the  $(\sum_{j=1}^{i-1} m_j + 1)$ th to  $(\sum_{j=1}^{i-1} m_j + m_i)$ th rows of  $D_0$ .

*Step 3* Obtain a design  $D = (D_{(1)}^T, \dots, D_{(s)}^T)^T$ .

**Theorem 1** *The design  $D$  constructed in Algorithm 1 is an FSL( $(m_1, \dots, m_s), s, p$ ).*

The proof is obvious from the construction, so we omit it here. From the construction of general SLHDs in Xie et al. (2014), we have the following theorem directly.

**Theorem 2** *For each flexible SLHD constructed using Algorithm 1, its slice  $D_{(i)}$  is an SLHD( $m_1, m_i/m_1, p$ ), which holds for every  $i, 1 \leq i \leq s$ .*

**Example 1** Let us construct an FSL( $(2, 2, 4, 8), 4, 4$ ) using Algorithm 1. First, compute  $r_1 = 2/2 = 1, r_2 = 4/2 = 2, r_3 = 8/4 = 2, r_4 = 16/8 = 2$ . Then using a GSLHD(1, 2, 2, 2; 2; 4), we can construct an FSL( $(2, 2, 4, 8), 4, 4$ ) as

$$D = \begin{pmatrix} 1 & 9 & 7 & 15 & 14 & 6 & 11 & 3 & 2 & 10 & 13 & 5 & 12 & 4 & 8 & 16 \\ 4 & 12 & 5 & 13 & 2 & 10 & 8 & 16 & 15 & 7 & 1 & 9 & 11 & 3 & 6 & 14 \\ 6 & 14 & 10 & 2 & 3 & 11 & 7 & 15 & 12 & 4 & 8 & 16 & 13 & 5 & 1 & 9 \\ 10 & 2 & 7 & 15 & 6 & 14 & 3 & 11 & 13 & 5 & 9 & 1 & 8 & 16 & 12 & 4 \end{pmatrix}^T \tag{2}$$

Figure 2 shows the two-dimensional projections of the flexible SLHD in (2), where the symbols ( $\bullet, \blacktriangle, \circ, \triangle$ ) represent the points from its 1st, 2nd, 3rd and 4th slices, respectively (each level is shifted to its left by 0.5).

For the first two slices of design  $D$  in (2), it is easy to check that they satisfy the property in Theorem 2, i.e. they are two SLHD(2, 1, 4)'s (in fact two LHD(2, 4)'s). As to the last two slices, Fig. 3 plots the two-dimensional projections of these two slices (with their levels being properly collapsed, and then each level is shifted to its left by 0.5), which show that the property in Theorem 2 also holds for these two slices, i.e. they are SLHD(2, 2, 4) and SLHD(2, 4, 4), respectively.

### 3.2 Method II

In this subsection, we propose an approach to constructing an FSL( $(m_1, \dots, m_s), s, p$ ), based on the method in Qian (2012), using the level expansion procedure. Assume  $m_1 | m_i$  and  $m_i | N$ , i.e.,  $N = m_i t_i$  ( $1 \leq i \leq s$ ). The details are as follows.

**Algorithm 2** *Step 1* Generate an SLHD( $m_1, t_1, p$ ) using the method in Qian (2012), denote the design as  $D_0$ .

*Step 2* Collapse the levels of each column in  $D_0$  according to  $i \rightarrow \lceil i/t_1 \rceil, i = 1, \dots, N$ ; denote the obtained design as  $D_1$ , such a design contains  $t_1$  LHD( $m_1, p$ )'s.

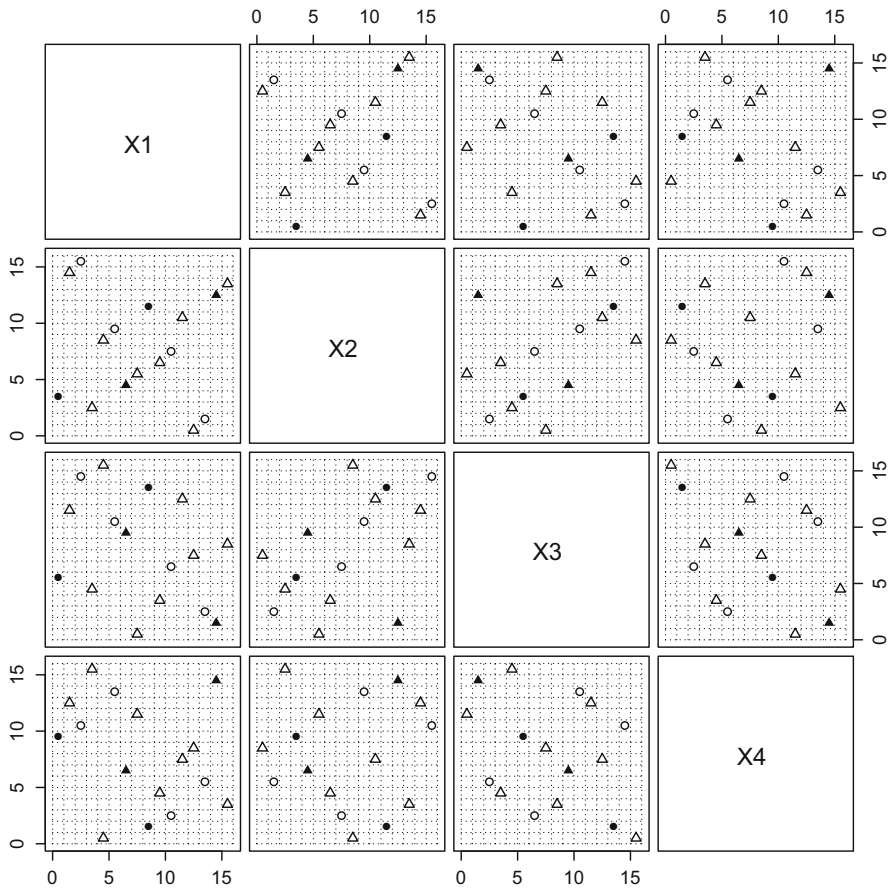


Fig. 2 Two-dimensional projections of the flexible SLHD in (2). The two “●” are from its 1st slice, the two “▲” are from its 2nd slice, the four “○” are from its 3rd slice, the eight “△” are from its 4th slice

Step 3 Partition the rows of  $D_1$  as follows:  $D_{1(1)}$  consists of the first  $m_1$  rows of  $D_1$ , for  $2 \leq i \leq s$ ,  $D_{1(i)}$  consists of the  $(\sum_{j=1}^{i-1} m_j + 1)$ th to  $(\sum_{j=1}^{i-1} m_j + m_i)$ th rows of  $D_1$ , i.e.,  $D_{1(i)}$  consists of  $m_i/m_1$   $LHD(m_1, p)$ 's.

Step 4 Take the  $j$ th ( $j = 1, \dots, p$ ) column of  $D_1$ , for  $w = s, \dots, 1$ , repeat:

Divide the elements of  $\mathbb{Z}_N$  into  $m_w$  blocks,  $b_1, \dots, b_{m_w}$ , where

$$b_i = \{a \in \mathbb{Z}_N \mid \lceil a/t_w \rceil = i\}.$$

Replace the  $m_w/m_1$  entries  $k$  ( $k = 1, \dots, m_1$ ) in  $D_{1(w)}(:, j)$  by randomly choose an element from the  $((k-1)m_w/m_1 + 1)$ th to  $((k-1)m_w/m_1 + m_w/m_1)$ th blocks, respectively, except for the elements that have been chosen once.

During the repeat process, once there is a block that all its elements have been chosen once, stop the loop and restart from the beginning of Step 4.

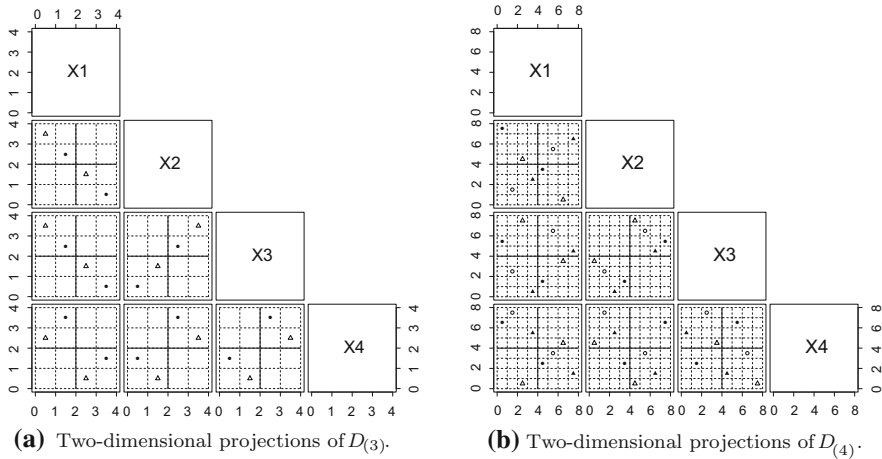


Fig. 3 Two-dimensional projections of the last two slices of  $D$  in (2)

Step 5 Denote the corresponding obtained designs by  $D_{(s)}, \dots, D_{(1)}$ , respectively. Stack them row by row to form design  $D = (D_{(1)}^T, \dots, D_{(s)}^T)^T$ .

**Remark 1** In Step 4, maybe more than one repeat loop should be implemented. For example, take  $m_1 = 2, m_2 = 6, m_3 = 8, m_4 = 8, N = 24$ , for  $w = 4$  and  $w = 3$ , if integers 5, 6, 7, 8 are chosen from the 2nd and 3rd blocks, then when  $w = 2$ , there is a block, i.e., {5, 6, 7, 8}, with all its elements having been chosen once, so restarting from the beginning of Step 4 is necessary. During the process of selecting elements for level expansion in Step 4, we should choose the elements such that they are uniformly scattered in each block and ensure that all the elements having been chosen are spread as evenly as possible on  $\mathbb{Z}_N$  (do not take two successive integers on  $\mathbb{Z}_N$ , if possible). In this way, a flexible SLHD will be finally found for the parameters satisfying the conditions given at the beginning of this subsection.

The following example is an illustration for Remark 1.

**Example 2** Take  $m_1 = 2, m_2 = 6, m_3 = 8, m_4 = 8, N = 24$  as in Remark 1, consider constructing one column of an FSL((2, 6, 8, 8), 4,  $p$ ). Generate an SLHD(2, 12, 1),

$$D_0 = (17 \ 1 \ ; 21 \ 9 \ 16 \ 10 \ 19 \ 11 \ ; 14 \ 12 \ 23 \ 3 \ 5 \ 24 \ 4 \ 18 \ ; 20 \ 2 \ 7 \ 13 \ 8 \ 22 \ 6 \ 15)^T,$$

then we have

$$D_1 = (D_{1(1)}^T, D_{1(2)}^T, D_{1(3)}^T, D_{1(4)}^T)^T = (2 \ 1 \ ; 2 \ 1 \ 2 \ 1 \ 2 \ 1 \ ; 2 \ 1 \ 2 \ 1 \ 1 \ 2 \ 1 \ 2 \ ; 2 \ 1 \ 1 \ 2 \ 1 \ 2 \ 1 \ 2)^T.$$

For  $w = 4$  and 3, replace the eight entries 1 in  $D_{1(4)}$  and  $D_{1(3)}$  by elements 1, 4, 7, 10 and 3, 6, 9, 12, respectively; replace the eight entries 2 in  $D_{1(4)}$  and  $D_{1(3)}$  by elements 13, 16, 19, 22 and 15, 18, 21, 24, respectively. For  $w = 2$ , replace the three entries 1

in  $D_{1(2)}$  by elements 2, 5, 11; replace the three entries 2 by elements 14, 17, 23. For  $w = 1$ , replace its two entries 1 and 2 by 8 and 20, respectively. Then, we have

$$D = (D_{(1)}^T, D_{(2)}^T, D_{(3)}^T, D_{(4)}^T)^T = (20 \ 8 \mid 14 \ 2 \ 17 \ 5 \ 23 \ 11 \mid 15 \ 3 \ 18 \ 6 \ 9 \ 21 \ 12 \ 24 \mid 13 \ 1 \ 4 \ 16 \ 7 \ 19 \ 10 \ 22)^T.$$

It is easy to verify that  $D$  is one column of an FSL((2, 6, 8, 8), 4,  $p$ ).

**Theorem 3** *The design  $D$  constructed in Algorithm 2 is an FSL( $(m_1, \dots, m_s)$ ,  $s$ ,  $p$ ).*

**Remark 2** For the cases in Method I, designs with the same parameters can also be constructed by Algorithm 2. Moreover, randomly shuffle the entries in each column of each slice in design  $D$ , the obtained design is still a flexible SLHD with the same parameters. In this way, in order to improve the space-filling properties of the constructed designs, we can adopt some optimality criteria for evaluating designs, such as the various measures of uniformity, among which, for example, the centered  $L_2$ -discrepancy (Hickernell 1998a) and the wrap-around  $L_2$ -discrepancy (Hickernell 1998b) are two popular choices.

**Theorem 4** *For a design constructed using Algorithm 2, its slice  $D_{(i)}$  is an SLHD( $m_1, m_i/m_1, p$ ), which holds for every  $i, 1 \leq i \leq s$ .*

Let us see an example to illustrate Algorithm 2.

**Example 3** Suppose  $N = 12, m_1 = 2, m_2 = 4, m_3 = 6, p = 3$ . First, generate an SLHD(2, 6, 3) using the method in Qian (2012), denote it as  $D_0$ ,

$$D_0 = \begin{pmatrix} 7 & 6 & 3 & 8 & 1 & 10 & 12 & 4 & 9 & 2 & 11 & 5 \\ 9 & 4 & 2 & 10 & 1 & 11 & 6 & 12 & 8 & 5 & 3 & 7 \\ 5 & 10 & 9 & 1 & 7 & 6 & 8 & 2 & 12 & 4 & 3 & 11 \end{pmatrix}^T. \tag{3}$$

Second, for each column of  $D_0$ , after its levels being collapsed according to Step 2 in Algorithm 2, we get design  $D_1$ ,

$$D_1 = \begin{pmatrix} 2 & 1 & 1 & 2 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 1 & 2 \end{pmatrix}^T.$$

Third, partition  $D_1$  as follows,

$$D_1 = (D_{1(1)}^T, D_{1(2)}^T, D_{1(3)}^T)^T = \left( \begin{array}{c|c|c} 2 & 1 & 1 & 2 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 1 & 2 \end{array} \right)^T.$$



Fourth, divide  $\mathbb{Z}_{12}$  into six blocks:  $\{1, 2\}$ ,  $\{3, 4\}$ ,  $\{5, 6\}$ ,  $\{7, 8\}$ ,  $\{9, 10\}$ ,  $\{11, 12\}$ , then we can obtain

$$D_{(3)} = \begin{pmatrix} 10 & 3 & 11 & 1 & 5 & 8 \\ 4 & 7 & 9 & 5 & 2 & 11 \\ 8 & 3 & 10 & 5 & 11 & 1 \end{pmatrix}^T ;$$

divide  $\mathbb{Z}_{12}$  into 4 blocks:  $\{1, 2, 3\}$ ,  $\{4, 5, 6\}$ ,  $\{7, 8, 9\}$ ,  $\{10, 11, 12\}$ , then we can have

$$D_{(2)} = \begin{pmatrix} 12 & 2 & 9 & 6 \\ 1 & 8 & 6 & 10 \\ 6 & 12 & 2 & 9 \end{pmatrix}^T ;$$

similarly, we can get  $D_{(1)}$ ,

$$D_{(1)} = \begin{pmatrix} 4 & 7 \\ 12 & 3 \\ 7 & 4 \end{pmatrix}^T .$$

Finally, stack the above three slices row by row, we have the design  $D$  given in (1), which is an  $\text{FSL}((2, 4, 6), 3, 3)$  with its two-dimensional projections shown in Fig. 1.

In Algorithm 2, if we take an SLHD with each of its slice being an orthogonal array based LHD as the initial SLHD in *Step 1*, a flexible SLHD with good stratification properties can then be obtained. That is to say, Method II can also be employed to generate flexible SLHDs with good stratification properties.

### 3.3 Method III

In this subsection, a more general method is proposed. As long as  $m_i|N$ , i.e.,  $N = m_i t_i$ ,  $1 \leq i \leq s$ , an  $\text{FSL}((m_1, \dots, m_s), s, p)$  can be constructed as follows.

**Algorithm 3** *Step 1* Give positive integers  $s, m_1, \dots, m_s$ , and  $N = m_1 + \dots + m_s$ .

*Step 2* For  $w = s, \dots, 1$ , repeat:

Divide the elements of  $\mathbb{Z}_N$  into  $m_w$  blocks,  $b_1, \dots, b_{m_w}$ , where:

$$b_i = \{a \in \mathbb{Z}_N | \lceil a/t_w \rceil = i\}. \tag{4}$$

Randomly choose an element from the 1st to  $m_w$ th blocks, respectively, except for the elements that have been chosen once. These  $m_w$  chosen elements constitute the entries of a column of the  $w$ th slice, denote it as  $D_{(w)}$ .

During the repeat process, once there is a block in which all its elements have been chosen once, stop the loop and restart from the beginning of *Step 2*.

**Table 1** Constraints on the parameter values for each proposed method

Method	Constraints on parameters
Algorithm 1	$m_{s+1} = N, m_i   m_{i+1}, 1 \leq i \leq s$
Algorithm 2	$m_1   m_i, m_i   N, 1 \leq i \leq s$
Algorithm 3	$m_i   N, 1 \leq i \leq s$

*Step 3* Stack the slices  $D_{(s)}, \dots, D_{(1)}$  obtained in *Step 2* row by row to form a column  $D = (D_{(1)}^T, \dots, D_{(s)}^T)^T$ .

The above algorithm provides the construction for one column of a flexible SLHD, independently implement the above algorithm  $p$  times, a flexible SLHD with  $p$  columns is thus obtained.

**Theorem 5** An  $FSL((m_1, \dots, m_s), s, p)$  will be obtained by independently implementing Algorithm 3  $p$  times.

Algorithm 3 is more general than Algorithms 1 and 2, and it can be used to construct new flexible SLHDs that not included in the aforementioned two methods.

**Example 4** For  $s = 4, m_1 = 2, m_2 = 3, m_3 = 3, m_4 = 4, N = 12, p = 2$ , a flexible SLHD generated using Algorithm 3 is

$$D = (D_{(1)}^T, D_{(2)}^T, D_{(3)}^T, D_{(4)}^T)^T = \left( \begin{array}{cc|cc|cc|ccc} 7 & 4 & 8 & 3 & 12 & 5 & 2 & 10 & 6 & 11 & 1 & 9 \\ 1 & 9 & 4 & 12 & 8 & 6 & 2 & 10 & 3 & 5 & 11 & 7 \end{array} \right)^T.$$

It is easy to verify that the above design is an  $FSL((2, 3, 3, 4), 4, 2)$ .

The conditions under which an  $FSL((m_1, \dots, m_s), s, p)$  exists for each of the aforementioned three proposed methods, i.e. Algorithms 1, 2 and 3, are summarized in Table 1. From this table, it is easy to see that Algorithm 2 is more flexible than Algorithm 1, and Algorithm 3 is the most flexible of all. In addition, flexible SLHDs generated by these three methods are limited to those constraints on parameters.

**Theorem 6** For an  $FSL((m_1, \dots, m_s), s, p)$  constructed by any of the three methods proposed above,

- (i) when  $s = 2$ , the design is an ordinary SLHD with  $N$  runs and two slices each with  $N/2$  runs;
- (ii) when  $s = 3$ , the positive integers  $m_1, m_2$  and  $m_3$  with  $m_1 \leq m_2 \leq m_3, N = m_1 + m_2 + m_3$  and  $m_i | N$  for  $i = 1, 2, 3$  must satisfy one of the following conditions:
  - (a)  $m_1 = m_2 = m_3$ ;
  - (b)  $m_3 = 2m_1$  and  $m_1 = m_2 < m_3$ ;
  - (c)  $m_2 = 2m_1, m_3 = 3m_1$ , and  $m_1 < m_2 < m_3$ .

The proof of Theorem 6 is deferred to the Appendix. For  $s > 3$ , when a flexible SLHD with parameters  $m_1, \dots, m_s$  can be constructed via one of the three methods proposed above, the conditions that all these parameters should satisfy can be deduced by the number theory method, the details are omitted here since they are too complicated.

### 3.4 Method IV

In this part, we will present a general method which can be used to generate an  $FSL((m_1, \dots, m_s), s, p)$  for any given positive integers  $m_1, \dots, m_s, s, p$  and  $N = m_1 + \dots + m_s$ . The following is the construction for one column of such a design, repeat the procedure  $p$  times independently, an  $FSL((m_1, \dots, m_s), s, p)$  can then be constructed.

**Algorithm 4** *Step 1* Same as *Step 1* of Algorithm 3.

*Step 2* Same as *Step 2* of Algorithm 3, except for replacing (4) with

$$b_i = \{a \in \mathbb{Z}_N \mid \lceil m_w a / N \rceil = i\}. \tag{5}$$

*Step 3* Same as *Step 3* of Algorithm 3.

**Theorem 7** *An  $FSL((m_1, \dots, m_s), s, p)$  will be obtained by independently implementing Algorithm 4  $p$  times.*

Algorithm 4 provides a way to generate flexible SLHDs for any given parameters. Obviously, it covers Algorithm 3 as a special case.

**Example 5** For  $s = 2, m_1 = 3, m_2 = 4, p = 1, N = 7$ , a flexible SLHD generated using Algorithm 4 is

$$D = (D_{(1)}^T, D_{(2)}^T)^T = ( 6 \ 4 \ 2 \ \vdots \ 7 \ 5 \ 3 \ 1 )^T.$$

Collapsing its slices by  $D_{(1)} \rightarrow \lceil D_{(1)} * 3/7 \rceil$  and  $D_{(2)} \rightarrow \lceil D_{(2)} * 4/7 \rceil$ , we get  $D_{(1)} \rightarrow (3, 2, 1)^T$  and  $D_{(2)} \rightarrow (4, 3, 2, 1)^T$ , it can be seen that the above design is an  $FSL((3, 4), 2, 1)$ .

For  $s = 3, m_1 = 3, m_2 = 5, m_3 = 6, p = 1, N = 14$ , a flexible SLHD generated using Algorithm 4 is

$$D = (D_{(1)}^T, D_{(2)}^T, D_{(3)}^T)^T = ( 12 \ 6 \ 4 \ \vdots \ 14 \ 11 \ 8 \ 5 \ 2 \ \vdots \ 13 \ 10 \ 9 \ 7 \ 3 \ 1 )^T.$$

Similarly, it can be verified that such a design is an  $FSL((3, 5, 6), 3, 1)$ .

A flexible sliced design with the same parameters (i.e.,  $s = 3, m_1 = 3, m_2 = 5, m_3 = 6, p = 1, N = 14$ ) generated by Kong et al. (2018)'s method is

$$\begin{aligned} D &= (D_{(1)}^T, D_{(2)}^T, D_{(3)}^T)^T \\ &= (0.066 \ 0.441 \ 0.680 \ \vdots \ 0.114 \ 0.287 \ 0.528 \ 0.627 \ 0.828 \ \vdots \ 0.085 \\ &\quad 0.254 \ 0.339 \ 0.555 \ 0.689 \ 0.864 )^T. \end{aligned} \tag{6}$$

The whole design  $D$  in (6) is not an LHD, which can be easily seen from Fig. 4.

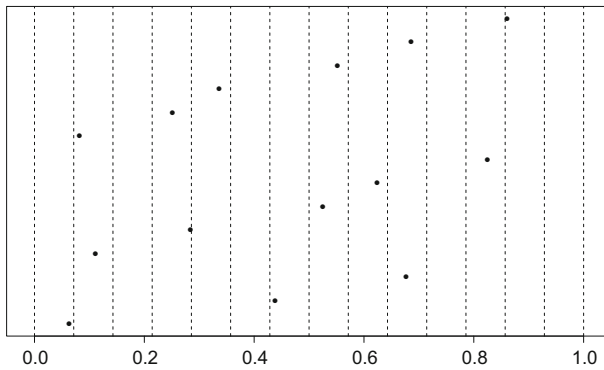


Fig. 4 Scatter plot of the flexible sliced design in (6)

**Remark 3** More generally, when the number of runs  $N$  is far less than the least common multiple of  $m_1, \dots, m_s$ , designs generated by our methods tend to have better one-dimensional uniformity than the designs proposed by Kong et al. (2018). Without loss of generality, consider the design constructed in Kong et al. (2018) with its parameters  $n_1 = \dots = n_k = 1$  and  $N_1, \dots, N_k$ , denoted by  $\text{FSD}(N_1 \times 1 + \dots + N_k \times 1)$ , such a design is a flexible sliced design with  $k$  slices of sizes  $N_1, \dots, N_k$  respectively. Let  $l$  be the least common multiple of  $N_1, \dots, N_k$ , the main procedure to generate one column of an  $\text{FSD}(N_1 \times 1 + \dots + N_k \times 1)$  is as follows [the notations used here are the same as those defined in Kong et al. (2018)]:

- (i) let  $M = (m_{i,h})_{k \times l}$  be a  $k \times l$  matrix whose  $h$ th column is a random permutation of  $\{(h - 1)k + 1, \dots, hk\}$ ;
- (ii) for  $i = 1, \dots, k, q = 1, \dots, N_i$ , draw  $a_q^{(i)}$  from  $\{m_{i,(q-1)\lambda_i+1}, \dots, m_{i,q\lambda_i}\}$ , where  $\lambda_i = l/N_i$ . Let  $M^{(i)} = (m_1^{(i)}, \dots, m_{N_i}^{(i)})^T$ , with its  $N_i$  elements being a random permutation of  $a_1^{(i)}, \dots, a_{N_i}^{(i)}$ ;
- (iii) for  $i = 1, \dots, k, q = 1, \dots, N_i$ , let

$$d_q^{(i)} = \frac{1}{kl}(m_q^{(i)} - \epsilon_{i,q}),$$

where  $\epsilon_{i,q}$ 's are independent random variables following the uniform distribution on  $(0, 1)$ .  $D^{(i)} = (d_1^{(i)}, \dots, d_{N_i}^{(i)})^T$  constitutes the  $i$ th slice of an  $\text{FSD}(N_1 \times 1 + \dots + N_k \times 1)$ .

From the above procedure to generate a flexible sliced design, it can be easily seen that each slice of a flexible sliced design can be collapsed into an LHD, while for the whole design, when the number of runs  $N$  ( $N = N_1 + \dots + N_k$ ) is far less than  $l$ , the one-dimensional uniformity of the whole design of an  $\text{FSD}(N_1 \times 1 + \dots + N_k \times 1)$  may be worse than an LHD. This also holds for flexible sliced designs with general parameters satisfying the condition that the number of runs  $N$  is far less than  $l$ .

### 4 Optimal flexible SLHDs

The whole design and each slice of a flexible SLHD generated by methods in Sect. 3 have one-dimensional maximum stratification property. However, a randomly generated design by the proposed methods can be poor in terms of the space-filling property in higher dimensions, see e.g., the two-dimensional projections shown in Fig. 3. In this section, we provide an optimization algorithm for generating optimal flexible SLHDs with desirable space-filling properties.

#### 4.1 Optimality criteria for flexible SLHDs

The uniform design is a major kind of space-filling design, various measures of uniformity have been introduced, see Fang et al. (2018) for more details. Hickernell (1998a) proposed the centered  $L_2$ -discrepancy ( $CL_2$ ) criterion to measure the uniformity of a design. Let  $X = \{x_1, \dots, x_N\}$  be a set of  $N$  points in the  $p$ -dimensional unit cube  $C^p = [0, 1]^p$ , where  $x_i = (x_{i1}, \dots, x_{ip})$ . The  $CL_2$  of  $X$  can be calculated as

$$(CL_2(X))^2 = \left(\frac{13}{12}\right)^p - \frac{2}{N} \sum_{i=1}^N \prod_{k=1}^p \left[1 + \frac{1}{2}|x_{ik} - 0.5| - \frac{1}{2}|x_{ik} - 0.5|^2\right] + \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \prod_{k=1}^p \left[1 + \frac{1}{2}|x_{ik} - 0.5| + \frac{1}{2}|x_{jk} - 0.5| - \frac{1}{2}|x_{ik} - x_{jk}|\right].$$

Here, the  $CL_2$  criterion is considered due to the appealing property that it becomes invariant under reordering the runs, relabeling factors, or reflecting the points about any plane passing through the center of the unit cube and parallel to its faces. For comparing designs, designs with lower  $CL_2$ -values are more uniform, and hence are more desirable.

When extending the  $CL_2$  criterion for evaluating the goodness of a flexible SLHD  $D$  with  $s$  slices of sizes  $m_1, \dots, m_s$  respectively, both the uniformity of the whole design and that of the sub-design in each slice should be considered. As a result, the uniform flexible SLHD should not only minimize  $CL_2(D)$  for the whole design, but also minimize  $CL_2(D_{(i)})$  for each slice ( $i = 1, \dots, s$ ). To solve this multi-objective optimization problem, we propose a single objective function that takes a weighted average of  $CL_2(D)$  and  $CL_2(D_{(i)})$  for each slice ( $i = 1, \dots, s$ ), denoted by  $\phi_{CL_2}(D)$ , where

$$\phi_{CL_2}(D) = \frac{1}{2} \left\{ CL_2(D) + \sum_{i=1}^s \frac{m_i}{N} CL_2(D_{(i)}) \right\},$$

with  $N = m_1 + \dots + m_s$ , more weights are given to slices that contain more design points. Based on this combined measure, we can define a uniform flexible SLHD as the one that minimizes  $\phi_{CL_2}(D)$ .

Other popular criteria for designing computer experiments can be extended for selecting the flexible SLHD in a similar way. For example, we can define a maximin distance flexible SLHD as one that maximizes the minimum inter-site distance based on the maximin distance criterion proposed by Johnson et al. (1990).

## 4.2 An optimization algorithm

In this section, the threshold accepting algorithm, proposed by Dueck and Scheuer (1990), is used to search uniform flexible SLHDs.

**Algorithm 5** (Optimization algorithm for uniform flexible SLHDs)

*Step 1* Randomly generate an initial flexible SLHD using the methods proposed in Sect. 3, denoted by  $D_0$ , and calculate  $\phi_{CL_2}(D_0)$ , denoted by  $d_0$ . Give a sequence of threshold parameters  $Th = (T_1, \dots, T_L)$ , where  $T_1 > \dots > T_L = 0$ . Denote the iteration number under each  $T_l$  by  $itmax$  for  $l = 1, \dots, L$ . Set indices  $l = 1$  and  $i = 1$ .

*Step 2* Randomly select a slice in  $D_0$ , and interchange two randomly chosen elements within a randomly selected column in this slice. Denote the new design by  $D_{try}$ . Calculate  $\phi_{CL_2}(D_{try})$ , denoted by  $d_{try}$ .

*Step 3* If  $d_{try} - d_0 < T_l$ , replace  $D_0$  by  $D_{try}$  and set  $d_0 = d_{try}$ ; else leave  $D_0$  unchanged.

*Step 4* Update  $i = i + 1$ ; if  $i \leq itmax$ , go to Step 2.

*Step 5* Update  $l = l + 1$ , if  $l \leq L$ , reset  $i = 1$  and go to Step 2; else deliver  $D_{best} = D_0$ .

Note that Algorithm 5 may find a locally optimal design, thus repeating Algorithm 5 with different initial designs can increase the possibility of finding a global optimal design.

**Remark 4** (i) The proposed optimization algorithm is an application of the threshold accepting algorithm in searching for optimal flexible sliced Latin hypercube designs under the  $\phi_{CL_2}(\cdot)$  criterion, and many other greedy optimization algorithms can be used. (ii) The choice of the length of the threshold sequence, i.e.  $L$  in Algorithm 5, depends on the size of the design, usually, the number of out circles  $L$  should be much less than the number of inner circles  $itmax$ . In the literature, it was suggested that  $L \in [10, 100]$  and  $itmax \in [10^4, 10^5]$  when the number of runs  $N \leq 1000$ , and  $L$  and  $itmax$  increase as  $N$  increases. One can do some empirical study in advance for obtaining suitable  $L$  and  $itmax$ , please refer to Fang et al. (2018, Sect. 4.2) for more details. In Examples 6 and 7, we take  $L = 11$ ,  $(T_1, T_2, \dots, T_{11}) = (10^{-5}, 9 \times 10^{-6}, \dots, 10^{-6}, 0)$ , and  $itmax = 10^4$ .

**Example 6** (Example 1 continued) For the FSL((2, 2, 4, 8), 4, 4) constructed in Example 1, it can be calculated that  $\phi_{CL_2}(D) = 0.2330$ . A uniform FSL((2, 2, 4, 8), 4, 4) generated using Algorithm 5, denoted by  $D_*$ , with  $\phi_{CL_2}(D_*) = 0.1933$  is as follows:

$$D_* = \begin{pmatrix} 13 & 4 & 6 & 11 & 7 & 15 & 2 & 10 & 3 & 1 & 12 & 5 & 9 & 16 & 8 & 14 \\ 6 & 13 & 10 & 4 & 2 & 12 & 7 & 15 & 1 & 9 & 14 & 16 & 5 & 8 & 11 & 3 \\ 9 & 4 & 6 & 13 & 1 & 11 & 15 & 7 & 8 & 10 & 2 & 12 & 3 & 5 & 16 & 14 \\ 4 & 9 & 15 & 8 & 11 & 14 & 6 & 1 & 3 & 13 & 7 & 10 & 16 & 5 & 2 & 12 \end{pmatrix}^T.$$

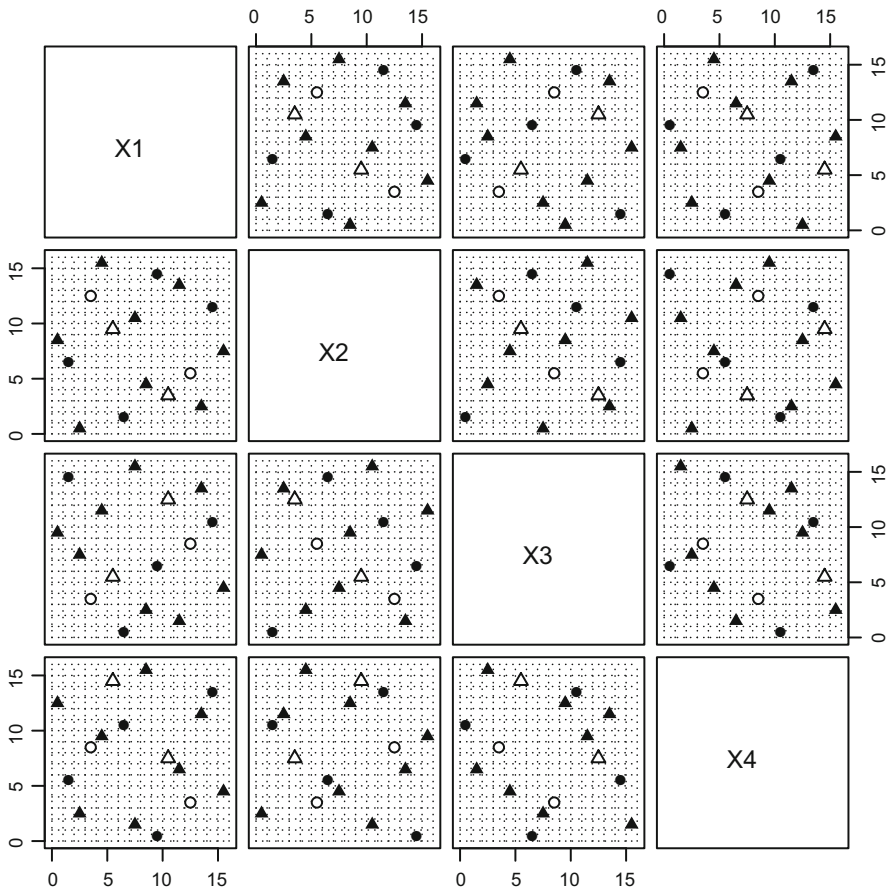
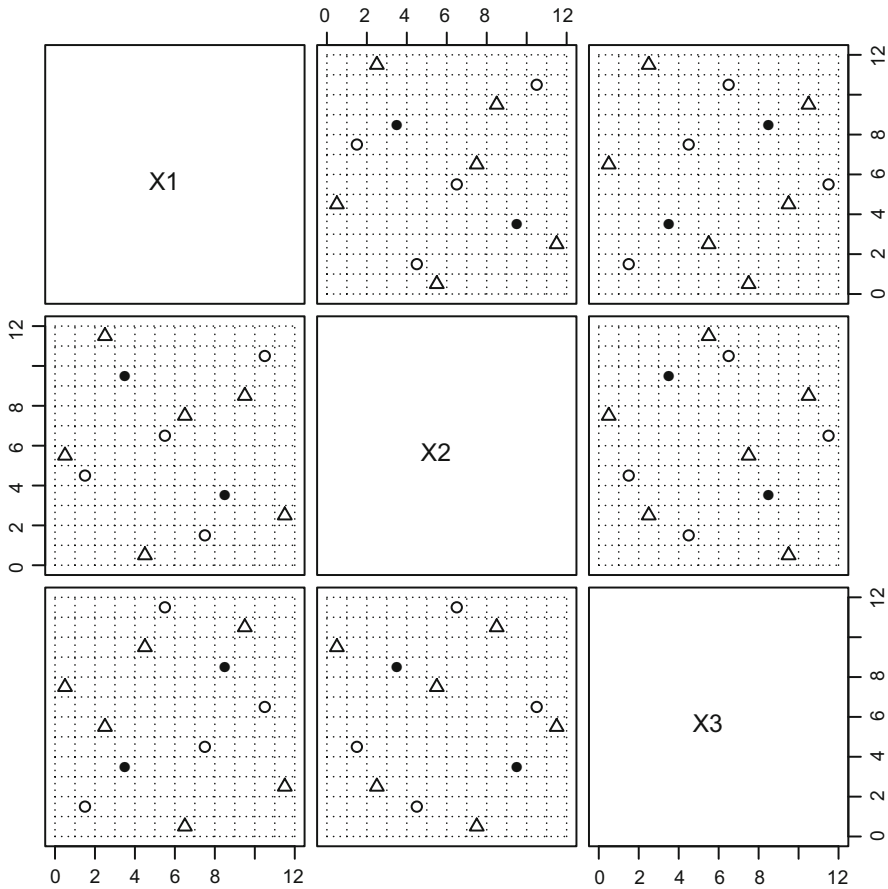


Fig. 5 Two-dimensional projections of the uniform FSL((2, 2, 4, 8), 4, 4)  $D_*$  in Example 6

The two-dimensional projections of  $D_*$  are shown in Fig. 5, where the symbols ( $\circ$ ,  $\Delta$ ,  $\bullet$ ,  $\blacktriangle$ ) represent the points from the 1st, 2nd, 3rd and 4th slices of  $D_*$ , respectively. It can be seen that both the whole design and each slice achieve a better uniformity than the design  $D$  in Example 1.

**Example 7** (Example 3 continued) For the FSL((2, 4, 6), 3, 3)  $D$  obtained in Example 3, we have  $\phi_{CL_2}(D) = 0.1643$ . A uniform FSL((2, 4, 6), 3, 3)  $D_*$  generated using Algorithm 5 with  $\phi_{CL_2}(D_*) = 0.1474$  is

$$D_* = \left( \begin{array}{cc|cc|cc|cc|cc|cc} 4 & 9 & 11 & 2 & 8 & 6 & 1 & 3 & 7 & 10 & 12 & 5 \\ 10 & 4 & 11 & 5 & 2 & 7 & 6 & 12 & 8 & 9 & 3 & 1 \\ 4 & 9 & 7 & 2 & 5 & 12 & 8 & 6 & 1 & 11 & 3 & 10 \end{array} \right)^T.$$



**Fig. 6** Two-dimensional projections of the uniform FSL((2, 4, 6), 3, 3)  $D_*$  in Example 7

The two-dimensional projections of  $D_*$  are shown in Fig. 6, where the symbols ( $\bullet$ ,  $\circ$ ,  $\triangle$ ) represent the points from the 1st, 2nd and 3rd slices of  $D_*$ , respectively. It also shows a better uniformity than the design  $D$  in Example 3.

## 5 Concluding remarks

In this paper, we propose a new class of SLHD, named the flexible SLHD, and provide several methods to construct such designs. For Methods I and II, different types of initial designs are applied, moreover, if the initial designs have “good” properties, the obtained flexible SLHDs also tend to have the same properties. Theoretical results on the constructed designs are developed. In addition, for two special cases when  $s = 2$  and 3, we provide some discussion on the sizes of the whole design and each slice for Methods I-III. Furthermore, an optimization algorithm is developed for selecting uniform flexible SLHDs under the commonly used centered  $L_2$ -discrepancy criterion.



Nested LHDs are widely used in computer experiments with multiple levels of accuracy (Qian 2009; Sun et al. 2014; Yang et al. 2014, 2016). Without loss of generality, consider a computer experiment with two-level of accuracy, the low-accuracy experiment (LE) and the high-accuracy experiment (HE). Denote the designs for LE and HE as  $D_l$  and  $D_h$ , respectively. If  $D_h \subset D_l$  and they are both LHDs (after levels being properly collapsed if necessary), then the pair of designs  $(D_h, D_l)$  is called a nested LHD. A flexible SLHD constructed in this paper is also a special nested LHD, each of its slice can be viewed as a small LHD that nested in the whole design. Besides, for a design constructed using Algorithms 1 or 2, each of its slices is also a special nested LHD after its levels being properly collapsed. In general, a flexible SLHD can be used for computer experiments with multiple levels of accuracy.

**Acknowledgements** The authors thank Editor Professor Werner G. Müller and two referees for their valuable comments. This work was supported by the National Natural Science Foundation of China (Grant Nos. 11771220, 11431006, and 11501305), National Ten Thousand Talents Program, Tianjin Development Program for Innovation and Entrepreneurship, Tianjin “131” Talents Program, Project 61331903, and Sichuan University Post-Doctor Research Project. The first two authors contribute equally to this work.

## Appendix: Proof of Theorem 6

- (i) It is known that  $N = m_1 + m_2$ , and  $m_1|N$ ,  $m_2|N$ . So we have  $m_1|m_2$  and  $m_2|m_1$ , as all the numbers are positive integers, therefore,  $m_1 = m_2 = N/2$ .
- (ii) For the positive integers  $m_1, m_2$ , and  $m_3$ , since  $m_1 \leq m_2 \leq m_3$ ,  $N = m_1 + m_2 + m_3$  and  $m_i|N$  for  $i = 1, 2, 3$ , then we have

$$m_1 + m_2 \leq 2m_3 \text{ and } m_3|(m_1 + m_2),$$

which imply

$$m_1 + m_2 = 2m_3, \text{ or} \quad (7)$$

$$m_1 + m_2 = m_3. \quad (8)$$

From Eq. (7), it is easy to deduce that  $m_1 = m_2 = m_3$ . From Eq. (8) and  $m_2|N$ , we have  $m_2|2(m_1 + m_2)$ , which implies  $m_2|2m_1$ ; since  $m_1 \leq m_2$ ,  $m_1$  and  $m_2$  have the following relation,

$$2m_1 = m_2, \text{ or} \quad (9)$$

$$2m_1 = 2m_2. \quad (10)$$

If Eq. (9) holds, it is easy to deduce that  $m_2 = 2m_1$ ,  $m_3 = 3m_1$ , and  $m_1 < m_2 < m_3$ . Alternatively, if Eq. (10) holds, it is easy to deduce that  $m_3 = 2m_1$  and  $m_1 = m_2 < m_3$ . This completes the proof.  $\square$

## References

- Ba S, Myers WR, Brennenman WA (2015) Optimal sliced Latin hypercube designs. *Technometrics* 57:479–487
- Bingham D, Sitter RR, Tang B (2009) Orthogonal and nearly orthogonal designs for computer experiments. *Biometrika* 96:51–65
- Cao RY, Liu MQ (2015) Construction of second-order orthogonal sliced Latin hypercube designs. *J Complex* 31:762–772
- Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm superior to simulated annealing. *J Comput Phys* 90:161–175
- Fang KT, Liu MQ, Qin H, Zhou YD (2018) Theory and application of uniform experimental designs. Springer and Science Press, Singapore
- Hickernell FJ (1998a) A generalized discrepancy and quadrature error bound. *Math Comput* 67:299–322
- Hickernell FJ (1998b) Lattice rules: how well do they measure up? In: Hellekalek P, Larcher G (eds) Random and quasi-random point sets. Springer, New York, pp 106–166
- Huang HZ, Yang JF, Liu MQ (2014) Construction of sliced (nearly) orthogonal Latin hypercube designs. *J Complex* 30:355–365
- Johnson ME, Moore LM, Ylvisaker D (1990) Minimax and maximin distance designs. *J Stat Plan Inference* 26:131–148
- Kong X, Ai M, Tsui KL (2018) Flexible sliced designs for computer experiments. *Ann Inst Stat Math* 70:631–646
- McKay MD, Beckman RJ, Conover WJ (1979) Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21:239–245
- Qian PZG (2009) Nested Latin hypercube designs. *Biometrika* 96:957–970
- Qian PZG (2012) Sliced Latin hypercube designs. *J Am Stat Assoc* 107:393–399
- Steinberg DM, Lin DKJ (2006) A construction method for orthogonal Latin hypercube designs. *Biometrika* 93:279–288
- Sun FS, Liu MQ, Lin DKJ (2009) Construction of orthogonal Latin hypercube designs. *Biometrika* 96:971–974
- Sun FS, Liu MQ, Lin DKJ (2010) Construction of orthogonal Latin hypercube designs with flexible run sizes. *J Stat Plan Inference* 140:3236–3242
- Sun FS, Pang F, Liu MQ (2011) Construction of column-orthogonal designs for computer experiments. *Sci China Math* 54:2683–2692
- Sun FS, Liu MQ, Qian PZG (2014) On the construction of nested space-filling designs. *Ann Stat* 42:1394–1425
- Wang XL, Zhao YN, Yang JF, Liu MQ (2017) Construction of (nearly) orthogonal sliced Latin hypercube designs. *Stat Probab Lett* 125:174–180
- Xie HZ, Xiong SF, Qian PZG, Wu CFJ (2014) General sliced Latin hypercube designs. *Stat Sin* 24:1239–1256
- Yang JY, Liu MQ (2012) Construction of orthogonal and nearly orthogonal Latin hypercube designs from orthogonal designs. *Stat Sin* 22:433–442
- Yang JF, Lin CD, Qian PZG, Lin DKJ (2013) Construction of sliced orthogonal Latin hypercube designs. *Stat Sin* 23:1117–1130
- Yang JY, Liu MQ, Lin DKJ (2014) Construction of nested orthogonal Latin hypercube designs. *Stat Sin* 24:211–219
- Yang JY, Chen H, Lin DKJ, Liu MQ (2016) Construction of sliced maximin-orthogonal Latin hypercube designs. *Stat Sin* 26:589–603
- Yang X, Yang JF, Lin DKJ, Liu MQ (2016) A new class of nested orthogonal Latin hypercube designs. *Stat Sin* 26:1249–1267
- Ye KQ (1998) Orthogonal column Latin hypercubes and their application in computer experiments. *J Am Stat Assoc* 93:1430–1439

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.