A flexible construction for sliced Latin hypercube designs

Jing Zhang 1,a, Jin Xu1 and Zhengming Wang2

1 College of Liberal Arts and Science, National University of Defense Technology, Changsha, Hunan, 410073, China
2 College of Systems Engineering, National University of Defense Technology, Changsha, Hunan, 410073, China
a Corresponding author: zhangjing¬_nudt@163.com

Abstract. Sliced Latin hypercube design is useful for computer experiments in batches or with quantitative factors. However, some constructions require the run size of each slice to be same. In this paper, we propose a method to construct a flexible sliced Latin hypercube design (FSLHD) with unequal run size for each slice. The whole design and each slice of FSLHD can achieve projective uniformity in one dimension. The FSLHD method is illustrated by some examples. The simulation indicates that, compared with some other sampling methods, the FSLHD method performs well.

1. Introduction
In engineering and science, computer experiments are becoming more and more important in recent years [1]. It is practical to simulate the real-world phenomenon by computer experiments [2]. In computer experiments, one of issues is to estimate the mean of output in a computer model \( f(x) \), where the input \( x = (x_1, ..., x_m) \) satisfies the uniform distribution on \( (0,1)^m \). A Latin hypercube design (LHD), proposed in [3] and it can be used to address the issues. An \( n \times m \) matrix, denoted by \( A=(a_{ij}) \), is called a Latin hypercube (LH) if each column of \( A \) is an independent permutation on \( Z_n \). Let \( D=(d_{ij}) \) of \( n \) runs in \( m \) factors be given through \( d_{ij} = (a_{ij} - \varepsilon_{ij}) / n \) for \( i = 1, ..., n \), \( j = 1, ..., m \) \( (1) \) defined a Latin hypercube design (LHD), where each \( \varepsilon_{ij} \sim U(0,1) \) is independent random variable. \( D \) has projective uniformity in one dimension, i.e., when \( D \) is projected onto each of the \( m \) factors, only one point accurately falls into each of \( n \) intervals denoted by \( [0,1/n), [1/n, 2/n), ..., [n-1/n, 1) \).

Consider each row of \( D \) as an input variable in a computer model, then the LHD is desirable to estimate the mean of output. The Latin hypercube sample can achieve a smaller variance than the independent identical distribution sample for estimating the mean of output in a computer model [4].

A method is proposed to construct a special Latin hypercube design named sliced Latin hypercube design (SLHD) in [5]. The whole design of the SLHD is partitioned into some slices and each slice is a smaller Latin hypercube design. One slice is a sub-design of the SLHD. The SLHD is desirable for running a computer model with batches or running multiple computer models based on similar mathematics [6], with one batch or one model using each slice of the SLHD as input values. However, the SLHD requires the run size of each slice to be identical. It may be limited in some practical situations. For example, because some alternative models have different precision and complexities on same problems, some computer experiments require different run sizes for different slices. Moreover,
we can find some variants of the sliced Latin hypercube designs in [7,8] with same restriction in run sizes. A flexible sliced design (FSD) with any run sizes for each slice is proposed in [9]. Each slice of the FSD is an LHD, while the whole design of the FSD is not an LHD. A sliced Latin hypercube design with different run size for each slice is introduced in [10], while the method only allows constructing the design with two different batches.

Inspired by the method of [11], we propose an algorithm to generate a flexible sliced Latin hypercube design (FSLHD) with flexible run size in each slice. The FSLHD has sliced structure described in [5]. Each slice is a sub-design of the FSLHD, where both the whole design and the sub-design are Latin hypercube designs (LHDs). The FSLHD can adapt to different batches of flexible run sizes in computer experiments. The simulation indicates that, compared with some other sampling methods, the FSLHD method performs well in estimating linear combination of sample mean values from similar computer models.

The paper is organized as follows. In Section 2, we introduce an algorithm to construct FSLHD. In Section 3, we provide the numerical illustration with some examples. Section 4 includes some concluding remarks.

2. Construction for FSLHD

In this section, we first provide some definitions and notations. For integers \(a \geq b \geq 1\), a discrete uniform distribution with support \(\{a, a + 1, \ldots, b\}\) is denoted by \(U_{ab}[a, b]\). For an integer \(r \geq 1\), let \(Z_r\) denote the set \(\{1, \ldots, r\}\). For a real number \(c\), \([c]\) is the smallest integer no less than \(c\). For a set \(A\), let \(|A|\) denote the number of elements of \(A\). For \(k\) integers \(n_1, \ldots, n_k\), the least common multiple of \(k\) integers is denoted by \(L = \text{lcm}(n_1, \ldots, n_k)\).

Now, we introduce the construction of a flexible sliced vector with \(u\) slices. For an integer \(u \geq 2\), \(i = 1, \ldots, u\), let \(n = \sum_{i=1}^{u} n_i\), \(L = \text{lcm}(n_1, \ldots, n_u, n')\), \(t_i = L / n_i\), \(t' = L / n'\), where \(n_i\) is the number of design points for each slice. Firstly, let \(\tau\) be a flexible sliced vector of \(n'\) points. Let \(\tau(1), \ldots, \tau(u)\) be \(u\) slices of \(\tau\).

For \(i = 1, \ldots, u\), the flexible sliced vector is denoted by

\[
\tau = (\tau(1), \ldots, \tau(n'))^T = (\tau^{(1)}, \ldots, \tau^{(u)})^T,
\]

where \(\tau^{(i)} = (\tau^{(1)}(1), \ldots, \tau^{(i)}(n'))\). The flexible sliced vector satisfies:

\[
(2.1) \left( \tau^{(i)} / t_i \right) = \left( \left[ \tau^{(1)}(1)/t_1 \right], \ldots, \left[ \tau^{(i)}(n_i)/t_i \right] \right) \text{ is a permutation of } Z_{n_i}, \text{ for } i = 1, \ldots, u,
\]

\[
(2.2) \left( \tau / t' \right) = \left( \left[ \tau(1)/t' \right], \ldots, \left[ \tau(n')/t' \right] \right) \text{ is a permutation of } Z_{n'}.
\]

The conditions of the flexible sliced vector can guarantee to generate a flexible sliced Latin hypercube design.

We propose Algorithm 1 to generate a flexible sliced vector with \(u\) slices. In this algorithm, Step 3 guarantees that we can generate a flexible sliced vector. For example, let \(n_1 = 2, n_2 = 3, \text{ so } n' = 5, L = \text{lcm}(2, 3, 5) = 30, t_1 = 15, t_2 = 10, t' = 6\). Firstly, we generate a random vector \(a = (3, 1, 2, 4, 5)\) on \(Z_{n_1}\). In Step 2, we may obtain \(\tau(1) = 13 \in U_{\alpha_1}[13, 18], \tau(2) = 2 \in U_{\alpha_1}[1, 6], \tau(3) = 12 \in U_{\alpha_2}[7, 12], \tau(4) = 22 \in U_{\alpha_2}[19, 24] \text{ and } \tau(5) = 29 \in U_{\alpha_2}[25, 30]\). We obtain a vector \(\tau = (13, 2, 12, 22, 29)^T\), then \(\tau^{(1)} = (13, 2)\) and \(\tau^{(2)} = (12, 22, 9)\), while \(\left[ \tau^{(i)} / t_i \right] = (1, 1)\) is not a permutation of \(Z_2\) and \(\left[ \tau^{(2)} / t_2 \right] = (2, 2, 3)\) is not a permutation of \(Z_3\). So the \(\tau = (13, 2, 12, 22, 29)^T\) is not a flexible sliced vector. Then we go back to Step 2 and continue the same process. If \(\tau = (13, 22, 12, 29, 2)^T\), we obtain \(\tau^{(1)} = (13, 22)\) and \(\tau^{(2)} = (12, 29, 2)\). We can see as follows:
(2.1) \( \tau^{(1)}/t_1 = (13,22)/15 = (1,2) \) is a permutation of \( Z_2 \), \( \tau^{(2)}/t_2 = (12,29,2)/10 = (2,3,1) \) is a permutation of \( Z_2 \).

(2.2) \( \tau^{(1)}/t' = (13,22,12,29,2)/6 = (3,4,2,5,1) \) is a permutation of \( Z_5 \).

So \( \tau = (13,22,12,29,2) \) satisfies the conditions of a flexible sliced vector.

**Algorithm 1** A flexible sliced vector with \( u \) slices

Step 1: Draw a random vector \( \alpha = (\alpha(1), \ldots, \alpha(n')) \) on \( Z_n' \).

Step 2:

\[
\text{for } i = 1, \ldots, n' \text{ do} \\
\quad \text{draw a } \tau(i) \text{ from } U_m[\{(\alpha(i) - 1)t' + 1, \alpha(i)t'\}]. \\
\text{end for}
\]

Step 3:

\[
\text{for } i = 1, \ldots, u \text{ do} \\
\quad \text{if } \left( [\tau^{(1)}(1)/t_1], \ldots, [\tau^{(n)}(n)/t_n] \right) \neq n_i \text{ then} \\
\quad \quad \text{go back to Step 2.} \\
\text{end if} \\
\text{end for}
\]

Step 4: Draw a flexible sliced vector with \( u \) slices \( \tau = (\tau^{(1)}, \ldots, \tau^{(n)})^T \).

Let \( M = (m_j) \) denote an \( n' \times m \) flexible sliced Latin hypercube (FSLH) in which all columns are flexible sliced vectors, and each column of FSLH is independent. A flexible sliced Latin hypercube design \( D = (d_j) \) of \( n' \) runs with \( m \) factors is constructed by

\[
d_{jk} = (m_j - u_j)/L, \quad \text{for } j = 1, \ldots, n', \ k = 1, \ldots, m (2)\]

where \( u_j \sim U(0,1) \) are independent random variables. \( D \) is composed of sub-designs \( D_i, \ldots, D_u \), row by row. For \( i = 1, \ldots, u \), the sub-design \( D_i \) contains \( n_i \) runs called \( i \)th sliced design of \( D \) and the \( D_i \) is projected onto each of \( m \) factors, only one point falls into each of \( n_i \) intervals denoted by \([0,1/n_i],[1/n_i,2/n_i], \ldots, [n_i-1/n_i,1] \), so the whole design \( D \) and the sliced design \( D_i \) are LHDs with projective uniformity in one dimension. For \( i = 1, \ldots, u \), let FSLHD \( (n_1, \ldots, n_u, m) \) denote a flexible sliced Latin hypercube design with \( u \) slices, where the \( i \)th sliced design of the FSLHD has \( n_i \) runs and \( m \) denotes the design dimension. We illustrate the construction of FSLHD through one example.

**Example 2.1**

Consider FSLHD \( (2,3,5,2) \) with \( n_1 = 2, n_2 = 3, n_3 = 5, n' = 10, L = \text{lcm}(2,3,5,10) = 30, t_1 = 15, t_2 = 10, t_3 = 6 \) and \( t' = 3 \). We draw a two-dimensional FSLH via Algorithm 1 and each column of the FSLH satisfies with the conditions (2.1),(2.2). The two-dimensional FSLH is displayed in Figure 1. An FSLHD \( (2,3,5,2) \) is constructed by equation (2). The design points of the FSLHD \( (2,3,5,2) \) with three slices are depicted in Figure 2. \( D \) denotes the whole design with three slices. Three sliced designs are \( D_i \).
\(D^2\) and \(D^3\), respectively. From Figure 2, we can see that \(D^1, D^2, D^3\) and \(D\) are LHDs with projective uniformity in one dimension.

\[
\begin{array}{cccc}
16 & 5 & 2 & 1 & 6 & 2 \\
10 & 18 & 1 & 2 & 4 & 6 \\
26 & 20 & 3 & 2 & 9 & 7 \\
2 & 26 & 1 & 3 & 1 & 9 \\
20 & 7 & 2 & 1 & 7 & 3 \\
15 & 24 & 3 & 4 & 5 & 8 \\
24 & 12 & 4 & 2 & 8 & 4 \\
4 & 29 & 1 & 5 & 2 & 10 \\
8 & 14 & 2 & 3 & 3 & 5 \\
29 & 3 & 5 & 1 & 10 & 1 \\
\end{array}
\]

Figure 1. In Example 2.1, \(n_1=2, n_2=3, n_3=5\), the FSLH with three slices (left). Each column of FSLH satisfies with the conditions (2.1),(2.2) (right).

![Figure 1](image1.png)

(a) (b)

(c) (d)

Figure 2. In Example 2.1, the former three figures are the points of the three sliced designs (\(D^1, D^2, D^3\)). The last figure is the points of the whole design (\(D\)).

3. Numerical Simulation

In this section, two examples are given to illustrate some advantages of the FSLHHD method. In Sections 1, two motivating problems are to run a computer model in batches and run multiple computer models based on similar mathematics, where each bath or model uses one slice of the sliced design as inputs.

We consider a computer experiment using \(u\) similar computer model \(f_1(x), ..., f_u(x)\). The input values \((x = (x_1,...,x_n))\) satisfy the uniform distribution on \((0,1]^n\). For \(i = 1,...,u, \; n_i > 0, \; 0 \leq \lambda_i \leq 1, \; \sum_{i=1}^u \lambda_i = 1\), define

\[
\mu_i = E[f_i(x)], \quad \eta = \sum_{i=1}^u \lambda_i \mu_i, \quad (3)
\]
each $f_i$ has $n_i$ design points as input values. We run computer models $f_1,...,f_u$ to get the estimation of $\mu_1,...,\mu_u$. Here we aim to estimate $\eta$ defined by (3). We introduce four sampling methods to realize the aim:

Let $n' = \sum_{i=1}^{u} n_i$, for $i = 1,...,u$,

i. SPLHD: generate an $n' \times m$ LHD of $n'$ runs and split it into $u$ slices, where run size of each slice is $n_i$ and each of slices corresponds to one $f_i$;

ii. COLHD: combine $u$ independent Latin hypercube designs of $n_i,...,n_u$ runs, respectively, each of which corresponds to each $f_i$;

iii. FSD: generate an $n' \times m$ a flexible sliced design with $u$ slices in [8]. Each slice is a smaller LHD of $n_i$ run, which is assigned to each $f_i$;

iv. FSLHD: generate an $n' \times m$ FSLHD with $u$ slices. Each slice is a smaller LHD of $n_i$ runs, which is assigned to each $f_i$.

For above methods, let $X_i$ of $n_i$ design points denote one slice as inputs in a computer model. The estimations of $\mu_i$ and $\eta$ are defined by

$$\hat{\mu}_i = \sum_{k=1}^{n} f_i(x^k) / n_i, \quad \hat{\eta} = \sum_{i=1}^{u} \lambda_i \hat{\mu}_i, \quad \text{for } i = 1,...,u, \tag{4}$$

where $x^k$ is the $k$th row of the $X_i$.

For four methods, we compare the root mean square errors (RMSEs) of $\hat{\eta}$ and $3\hat{\mu}$ through two examples.

**Example 3.1**

E1: $f(x) = \log(x_1x_2x_3x_4x_5)$,

E2: $f_1(x) = \log(1/\sqrt{x_1} + 1/\sqrt{x_2})$,

$$f_2(x) = \log(0.98/\sqrt{x_1} + 0.95/\sqrt{x_2})$$

$$f_3(x) = \log(1.02/\sqrt{x_1} + 1.02/\sqrt{x_2})$$

The function of E1 is from [12] and the $x$ satisfies uniform distribution on $(0,1)^5$. The functions of E2 are from [5] and the $x$ satisfies uniform distribution on $(0,1)^2$. In E1, assume $f_1(x) = f_2(x) = f_3(x) = f(x)$, then $f(x)$ acts as a computer model and we provide three machines to estimate $\eta$, where the designs $X_1',X_2'$ and $X_3'$ are the sets of the input values for the three machines. In E2, we use the three similar computer models to estimate $\eta$, where the designs $X_1',X_2'$ and $X_3'$ are the sets of the input values for the three similar computer models.

For the four schemes, we estimate $\eta$ by $\hat{\eta}$ based on the designs $X_1',X_2'$ and $X_3'$, and we estimate $\hat{\mu}_i$ by $\hat{\mu}_i$ based on the design $X_i'$ via the equation (4). For $n_1 = 3$, $n_2 = 4$, $n_3 = 5,6,7$, Table 1 shows the RMSEs of $\hat{\eta}$ over the $10^4$ replicates with $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$ for each scheme. Table 2 shows the RMSEs of $\hat{\mu}_i$ over the $10^4$ replicates for each scheme.

**Table 1.** The RMSEs of $\hat{\eta}$ with $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$, in Example 3.1.

| $n_1 = 3, n_2 = 4$ | SPLHD | COLHD | FSD | FSLHD |
|--------------------|--------|-------|-----|-------|
| **E1**             |        |       |     |       |
| $n_i = 5$          | 0.1927 | 0.3292| 0.3269| 0.1930|
| $n_i = 6$          | 0.1785 | 0.3026| 0.2895| 0.1772|
| $n_i = 7$          | 0.1677 | 0.2858| 0.2862| 0.1664|
| **E2**             |        |       |     |       |
| $n_i = 5$          | 0.0259 | 0.0368| 0.0373| 0.0230|
Table 2. The RMSEs of $\hat{\mu}_i$ in Example 3.1.

| $n_1=6$ | $n_2=7$ |
|---------|---------|
| 0.0269  | 0.0294  |
| 0.0356  | 0.0350  |
| 0.0345  | 0.0355  |
| 0.0229  | 0.0222  |

Table 3. The RMSEs of $\hat{\eta}$ with $\lambda_1=0.3, \lambda_2=0.4$, in Example 3.1.

| $n_1=3, n_2=4$ | SPLH D | COLHD | FSD D | FSLH D |
|----------------|--------|-------|-------|--------|
| E1 $n_1=5$    | 0.8165 | 0.4567 | 0.4588 | 0.4667 |
| $n_2=6$       | 0.7201 | 0.3842 | 0.3805 | 0.3837 |
| $n_2=7$       | 0.6411 | 0.3312 | 0.3257 | 0.3308 |
| E2 $n_1=5$    | 0.0826 | 0.0496 | 0.0499 | 0.0499 |
| $n_2=6$       | 0.0734 | 0.0424 | 0.0431 | 0.0427 |
| $n_2=7$       | 0.0655 | 0.0369 | 0.0374 | 0.0373 |

4. Further discussion

In this paper, we construct a flexible sliced Latin hypercube design (FSLHD) with different run size in different slices. The FSLHD method avoids the restriction in run size in [5,7,8]. Both the whole design and each slice of FSLHD are LHDs with projective uniformity in one dimension. Compared with some other sliced design methods, FSLHD method is desirable in estimating linear combination of sample mean values from multiple computer models. In our future work, some optimization algorithms based on space-filling criteria [13,14] will be considered to achieve better space-filling properties of FSLHD.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (No.61573367, No. 11771450).
References
[1] K. T. Fang, R. Z. Li, A. Sudjianto, Design and Modeling for Computer Experiments (New York: Chapman and Hall/CRC, 2006).
[2] C. F. J. Wu, Post-Fisherian Experimentation: From Physical to Virtual. J Amer Statistical Assoc, 110, 612-620 (2015).
[3] M. D. McKay, R. J. Beckman, W. J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 21, 381-402 (1979).
[4] M. Stein, Large sample properties of simulations using Latin hypercube sampling. Technometrics, 29, 143-151 (1987).
[5] P. Z. G. Qian, Sliced Latin hypercube designs. J Amer Statistical Assoc, 107, 393-399 (2012).
[6] B. Williams, M. Morris, T. Santner, Using multiple computer models/Multiple data sources simultaneously to infer calibration parameters. 2009 INFORMS Annual Conference, October CA: San Diego, 11–14 (2009).
[7] Y. Yin, D. K. J. Lin, M. Q. Liu, Sliced Latin hypercube designs via orthogonal arrays. J. Statist. Plann. Inference, 149, 162-171 (2014).
[8] Y. Hwang, X. He, P. Z. G. Qian, Sliced orthogonal array-based Latin hypercube designs. Technometrics, 58, 50-61 (2016).
[9] X. Kong, M. Ai, K. L. Tsui, Flexible sliced designs for computer experiments. AnISM, 70, 1-16 (2018).
[10] J. Xu, X. He, X. J. Duan and Z. M. Wang, Sliced Latin hypercube designs for computer experiments with unequal batch sizes," in IEEE Access. doi:10.1109/ACCESS. 2018. 2871166 (2018).
[11] D. J. Chen, S. F. Xiong, Flexible nested Latin hypercube designs for computer experiments. JQT, 49, 337-353 (2017).
[12] S. S. Drew, T. Homemdemello, Some large deviations results for Latin hypercube sampling. vbgWinter Simulation Conference, 673-681(2005).
[13] K. Q. Ye, W. Li, A. Sudjianto, Algorithmic construction of optimal symmetric Latin hypercube designs. J. Statist. Plann. Inference, 90, 145-159 (2000).
[14] A. Grosso, A. R. M. J. U. Jamali, M. Locatelli, Finding maximin Latin hypercube designs by iterated local search heuristics. Eur. J. Oper. Res, 197, 541-547 (2009).