IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 14, 2018, accepted September 13, 2018, date of publication September 19, 2018,
date of current version November 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2871166

Sliced Latin Hypercube Designs for Computer Experiments With Unequal Batch Sizes

JIN XU[1], XU HE[2], XIAOJUN DUAN[1], AND ZHENGMING WANG[3]
[1]College of Liberal Arts and Sciences, National University of Defense Technology, Changsha 410072, China
[2]Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China
[3]College of Systems Engineering, National University of Defense Technology, Changsha 410072, China

Corresponding author: Xu He (hexu@amss.ac.cn)

This work was supported by the Chinese Ministry of Science and Technology under Grant 2016YFF0203801 and in part by the National Science Foundation of China under Grant 11771450, Grant 61573367, Grant 11501550, Grant 11671386, Grant A011101, and Grant 11701566.

ABSTRACT Latin hypercube designs are frequently used in estimating the mean output value of computer simulations given random environmental factors. Sliced Latin hypercube designs are designs that can be partitioned into a number of batches so that both the whole design and the batches achieve optimal univariate uniformity. Such designs are useful for computer simulations that are carried out in batches, come from multiple resources, or have categorical variables. All existing sliced Latin hypercube designs have equal batch sizes. In this paper, we propose a new type of sliced Latin hypercube design that has unequal batch sizes and show their advantages theoretically and numerically.

INDEX TERMS Computer simulation, data integration, emulation, sampling methods, uncertainty.

I. INTRODUCTION

Computer simulations are frequently used in product and engineering design. In many scenarios, the output of computer experiments is influenced by environmental factors that can be assumed to follow random distributions [1]. To solve the problem of uncertainty, we need to compute the mean output value using sampling techniques. Let X denote an input value in $[0, 1]^d$ and $f(X)$ denote the output value, then the mean output value of $f(X)$ is $\mu = \int_{[0,1]^d} f(x)\mathrm{d}x$. Using a design $D \subset [0, 1]^d$ with n runs, we usually transform the variables to the uniform distribution on $[0, 1]$ and use $\hat{\mu} = \sum_{x\in D} f(x)/n$ to estimate μ.

Clearly, the accuracy of $\hat{\mu}$ is closely related to the design. We can also write a design with n runs as an $n \times d$ matrix, where each row gives one design point. Using this definition, a design with n points is called an Latin hypercube design (LHD) if each column of it has exactly one point in each of the n bins of $(0, 1/n], \cdots, ((n-1)/n, 1]$. LHDs achieve optimal univariate uniformity. As a result, $\hat{\mu}$ computed from an LHD is more accurate than that obtained from independent runs [4]. Also because LHDs can be constructed for any d and n, it is desirable to use an LHD to estimate μ, especially for large d problems. Reference [5] proposed the first type of LHD, referred to as ordinary LHD hereinafter.

A sliced Latin hypercube design (SLHD) is an LHD that can be partitioned into a number of small LHDs called slices or batches [6]. In some applications, for some uncontrollable reasons the computer runs in one or more batches may break down. In this case an SLHD is desirable because when no batch breaks down, the whole design achieves optimal univariate uniformity while when one or more batches break down, the remaining runs still have some uniformity properties. SLHDs are also useful for computer experiments with categorical variables, from multiple sources and model validation.

The original SLHD proposed in [6] and its variants (see [7]–[10]) have the restriction that sizes of the batches must be equal. However, in some applications we may want designs with unequal batch sizes. For instance, consider the situation that computer experiments are carried out by a number of computers, where the runs from each computer consist of a batch. If all computers have the same computing capability, then the batch sizes should be equal. On the other hand, if some computers have faster computation speed than others, in order to finish all computer runs in a fixed time, the batch sizes should be assigned according to the speed of computers and should be unequal. This calls for SLHDs with unequal batch sizes. One method that generates sliced designs with unequal batch sizes is flexible sliced designs (FSD) proposed in [11]. From an FSD, each batch is an LHD and the whole design has some uniformity properties. However, the whole design of an FSD does not achieve optimal univariate uniformity and are not LHD. Finally, the method given in [12] constructs designs with two batches for which both the

first batch and the whole design are Latin hypercube designs. However, from this method, the second batch is not a Latin hypercube design.

In this paper, we propose a new type of SLHD called sliced Latin hypercube designs with unequal batch sizes (referred to as USLHDs hereinafter). Both its whole design and its slices are LHDs. Furthermore, its number of slices is flexible and its batch sizes are unequal. We give the construction of USLHDs in Section II and provide some properties of USLHDs in Section III. In Section IV, we corroborate the usefulness of USLHDs in estimating the mean output value by numerical comparisons on two toy examples and two real examples in engineering and circuit analysis. Section V concludes this paper. All proofs are provided in the Appendix.

II. CONSTRUCTIONS

A. CONSTRUCTION FOR DESIGNS WITH TWO SLICES

In this section, we give two algorithms to construct USLHDs. In this subsection, we give our first algorithm that is applicable to USLHDs with two slices. This algorithm serves as the building block for our general algorithm.

We first give some definitions. For an integer $n > 0$, let Z_n denote the set $\{1, \cdots, n\}$. For two positive integers a, b, let $mod(a, b)$ denote the integer $c \in Z_b$ which is congruent to a modulo b. For sets A and B, let $A \setminus B$ denote the relative complement of B in A. Let n_1 and n_2 denote the sizes of the two batches, $n = n_1 + n_2$, g be the greatest common divisor of n_1 and n_2, $n_1' = n_1/g$, $n_2' = n_2/g$, and $n' = n/g$. Our first algorithm is given below:

Algorithm 1

Step 1: Generate an n_1-dimensional vector $\alpha = (\alpha_1, \ldots, \alpha_{n_1})$. The α_1 is generated from the discrete uniform distribution on $Z_{n'}$. For $i = 2, \cdots, n_1$, generate r_i from the discrete uniform distribution on $\{0, 1, \cdots, n_2'\}$ and compute $\alpha_i = mod(\alpha_{i-1} - r_i, n') + n'(i-1)$, where the α_1 and the r_i's are generated independently. Randomly permute α to get α^*.

Step 2: Generate an n_2-dimensional vector $\beta = (\beta_1, \ldots, \beta_{n_2})$. Let l_i denote the ith smallest element of $Z_n \setminus \{\lceil \alpha_1/n_1' \rceil, \cdots, \lceil \alpha_{n_1}/n_1' \rceil\}$. For $i = 1, \cdots, n_2$, generate β_i independently from the discrete uniform distribution on $\{n_2'(l_i - 1) + 1, \cdots, l_i n_2'\} \cap \{n'(i-1) + 1, \cdots, in'\}$. Randomly permute β to get β^*.

Setp 3: Generate ϵ_α and ϵ_β independently from uniform distributions on $(0, 1]^{n_1}$ and $(0, 1]^{n_2}$, respectively, and let $\rho = (\alpha^* - \epsilon_\alpha)/(n_1' n)$ and $\tau = (\beta^* - \epsilon_\beta)/(n_2' n)$. Let $h^{(1)} = (\rho^T, \tau^T)^T$.

Setp 4: Repeat Steps 1-3 to obtain $h^{(2)}, \cdots, h^{(d)}$, and let $H = (h^{(1)}, \cdots, h^{(d)})$. The H, its first n_1 rows and its last n_2 rows give the USLHD and its two slices, respectively.

This algorithm guarantees that all of the first slice, the second slice, and the whole design are Latin hypercube designs.

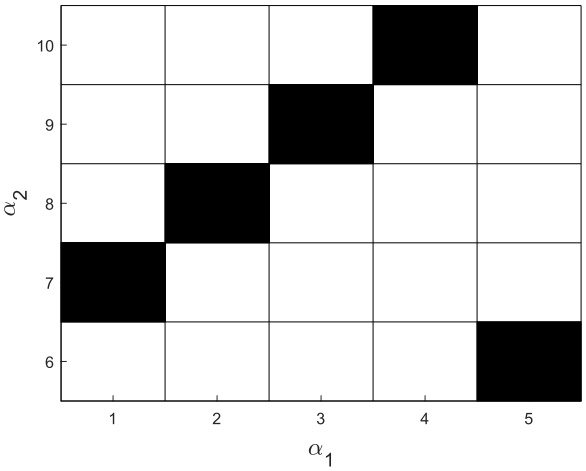

FIGURE 1. The probability mass function of α_1 and α_2. The probability in the black zones is zero and the probability in white zones is 1/20.

Meanwhile, each run of the design obeys uniform distribution on $(0, 1]^d$. These results are given in Theorem 1 and Theorem 2.

We remark that we cannot use the first column of an ordinary LHD with n_1 runs as ρ. This is because not every LHD can be extended to an LHD with more runs. For instance, consider the case with $n_1 = 2$, $n_2 = 3$ and $n = 5$. The $\rho = (0.42, 0.53)$ is a probable column of an ordinary LHD. However, based on this ρ there is no τ that makes $h^{(1)}$ a column of an LHD, because both 0.42 and 0.53 lie in $[2/5, 3/5)$ but a column of an LHD should contain exactly one point in $[2/5, 3/5)$. To solve this problem, we force the difference among entries of ρ to be larger than $1/n$. This is to guarantee that no two ρ_is fall within the same interval among $(0, 1/n], \cdots, ((n-1)/n, 1]$. In the meantime, from Proposition 1 that will be given in Section III, our Step 1 guarantees that the set $\{n_2'(l_i - 1) + 1, \cdots, l_i n_2'\} \cap \{n'(i-1) + 1, \cdots, in'\}$ is non-empty, insuring the validity of our Step 2. We provide a simple example to illustrate Algorithm 1.

Example 1: Consider $n_1 = 2$, $n_2 = 3$ and $d = 1$. Clearly, $g = 1$, $n' = n = 5$, $n_1' = 2$ and $n_2' = 3$. Figure 1 shows the probability mass function of α_1 and α_2. Suppose we sample $\alpha_1 = 2$. Then $\alpha_2 = mod(2 - r_2) + 5$ and r_2 follows the discrete uniform distribution on $\{0, 1, 2, 3\}$. When $r_2 = 0, 1, 2, 3$, the $mod(2 - r_2) = 2, 1, 5, 4$, respectively. Therefore, the range of α_2 is $\{6, 7, 9, 10\}$. Suppose we sample $\alpha_2 = 6$ and $\alpha^* = (6, 2)^T$. Then the set $Z_n \setminus \{\lceil \alpha_1/n_1' \rceil, \cdots, \lceil \alpha_{n_1}/n_1' \rceil\}$ in Step 2 is $\{2, 4, 5\}$. So β_1, β_2 and β_3 are sampled from $\{4, 5, 6\} \cap \{1, 2, 3, 4, 5\} = \{4, 5\}$, $\{10, 11, 12\} \cap \{6, 7, 8, 9, 10\} = \{10\}$ and $\{13, 14, 15\} \cap \{11, 12, 13, 14, 15\} = \{13, 14, 15\}$, respectively, with equal probability. Suppose we sample $\beta_1 = 5$, $\beta_2 = 10$, $\beta_3 = 13$, $\beta^* = (13, 5, 10)^T$, $\epsilon_\alpha = (0.7580, 0.7430)^T$ and $\epsilon_\beta = (0.392, 0.6560, 0.1705)^T$. Then $h^{(1)} = (\rho^T, \tau^T)^T = (0.5242, 0.1257, 0.8405, 0.2896, 0.6553)^T$. Clearly, exactly one entry of $h^{(1)}$ falls within each of the $(0, 1/5], \cdots, (4/5, 1]$, exactly one entry of ρ falls within

each of the $(0, 1/2]$ and $(1/2, 1]$ and exactly one entry of τ falls within each of the $(0, 1/3]$, $(1/3, 2/3]$ and $(2/3, 1]$.

We remark that the probability mass function given in Figure 1 is not the only choice to construct USLHDs. Any function that makes the (5, 6) cell black and that each column and row contains exactly one black cell works.

B. CONSTRUCTION OF DESIGNS WITH ARBITRARY NUMBER OF SLICES

In this subsection, we give our general algorithm to construct designs with arbitrary number of slices, assuming that there are t_1 slices of size m_1, t_2 slices of size m_2 and $m_1 \neq m_2$. Let $n_1 = t_1 m_1$, $n_2 = t_2 m_2$ and g, n, n_1', n_2' and n' be defined in the same way as in Section II-A. The algorithm is given below:

Algorithm 2

Step 1: Generate an n_1-dimensional vector α and an n_2-dimensional vector β by Steps 1 and 2 in Algorithm 1.

Step 2: Generate $\alpha^* = (\alpha_1^*, \ldots, \alpha_{n_1}^*)$. For $i = 1, \cdots, m_1$, randomly permute $\{\alpha_{(i-1)t_1+1}, \cdots, \alpha_{it_1}\}$ to obtain $\{\alpha_{(i-1)t_1+1}^{(1)}, \cdots, \alpha_{it_1}^{(1)}\}$. For $j = 1, \cdots, t_1$, randomly permute $\{\alpha_j^{(1)}, \alpha_{t_1+j}^{(1)}, \cdots, \alpha_{(m_1-1)t_1+j}^{(1)}\}$ to obtain $\{\alpha_{(j-1)m_1+1}^*, \cdots, \alpha_{jm_1}^*\}$.

Step 3: Generate $\beta^* = (\beta_1^*, \ldots, \beta_{n_2}^*)$. For $i = 1, \cdots, m_2$, randomly permute $\{\beta_{(i-1)t_2+1}, \cdots, \beta_{it_2}\}$ to obtain $\{\beta_{(i-1)t_2+1}^{(1)}, \cdots, \beta_{it_2}^{(1)}\}$. For $j = 1, \cdots, t_2$, randomly permute $\{\beta_j^{(1)}, \beta_{t_2+j}^{(1)}, \cdots, \beta_{(m_2-1)t_2+j}^{(1)}\}$ to obtain $\{\beta_{(j-1)m_2+1}^*, \cdots, \beta_{jm_2}^*\}$.

Setp 4: Generate ϵ_α and ϵ_β independently from uniform distributions on $(0, 1]^{n_1}$ and $(0, 1]^{n_2}$, respectively, and let $\rho = (\alpha^* - \epsilon_\alpha)/(n_1' n)$ and $\tau = (\beta^* - \epsilon_\beta)/(n_2' n)$. Let $h^{(1)} = (\rho^T, \tau^T)^T$.

Setp 5: Repeat Steps 1-4 to obtain $h^{(2)}, \cdots, h^{(d)}$, and let $H = (h^{(1)}, \cdots, h^{(d)})$. The H, rows $(i-1)m_1 + 1$ to im_1 and rows $(j-1)m_2 + 1 + n_1$ to $jm_2 + n_1$ give the USLHD, its ith m_1-run slice and its jth m_2-run slice, respectively.

We remark that Algorithm 1 is a special case of Algorithm 2 when $t_1 = t_2 = 1$.

III. PROPERTIES

In this section, we give some sampling properties of designs constructed from Algorithm 2. Firstly, Proposition 1 below validates our algorithms.

Proposition 1: We have $\lceil \alpha_i/n_1' \rceil \neq \lceil \alpha_j/n_1' \rceil$ for any $i, j = 1, \cdots, n_1$, $i \neq j$, and $\{n_2'(l_k - 1) + 1, \cdots, l_k n_2'\} \cap \{n'(k-1) + 1, \cdots, kn'\} \neq \emptyset$ for any $k = 1, \cdots, n_2$.

Next, we show that our generated designs are SLHDs.

Theorem 1: Let $h = (h_1, \ldots, h_n)$ denote an arbitrary column of H. We have

(i) Exactly one entry of h falls within each of the $(0, 1/n], \cdots, ((n-1)/n, 1]$.

(ii) Exactly one entry of $\{h_{(i-1)m_1+1}, \cdots, h_{im_1}\}$ falls within each of the $(0, 1/m_1], \cdots, ((m_1-1)/m_1, 1]$ for any $i = 1, \cdots, t_1$;

(iii) Exactly one entry of $\{h_{(j-1)m_2+1+n_1}, \cdots, h_{jm_2+n_1}\}$ falls with each of the $(0, 1/m_2], \cdots, ((m_2-1)/m_2, 1]$ for any $j = 1, \cdots, t_2$;

(iv) Exactly one entry of $\{h_1, \cdots, h_{n_1}\}$ falls within each of the $(0, 1/n_1], \cdots, ((n_1-1)/n_1, 1]$.

(v) Exactly one entry of $\{h_{n_1+1}, \cdots, h_n\}$ falls within each of the $(0, 1/n_2], \cdots, ((n_2-1)/n_2, 1]$.

Theorem 1 (i)-(iii) verify that the whole design and each batch are LHDs. Theorem 1 (iv) and (v) further show that the combination of the m_1-size batches and the combination of the m_2-size batches are also LHDs. Finally, $\hat{\mu}$ is an unbiased estimator of μ and $\hat{\mu} \to \mu$ as $n \to \infty$ if and only if the design points are uniformly distributed [13]. Theorem 2 shows that this is true for USLHD.

Theorem 2: The $(h_i^{(1)}, \cdots, h_i^{(d)})$ follows the uniform distribution on $[0, 1)^d$ for $i = 1, \cdots, n$.

From above results, similar to SLHD with equal batch sizes, USLHD achieves optimal univariate uniformity for both the whole design and its batches.

IV. NUMERICAL COMPARISON

In this section, we compare USLHD to some other schemes. Assume there are 3 low configuration computers and 3 high configuration computers for a numerical integration task, and from the time constraint we can arrange at most 6 and 8 computer trials for each low and high configuration computers, respectively. As discussed in Section 1, we propose to use a USLHD with $t_1 = 3$, $m_1 = 6$, $t_2 = 3$, $m_2 = 8$, $n_1 = 18$, $n_2 = 24$ and $n = 42$ to accomplish this task, where each 6-run batch is used for each low configuration computer and each 8-run batch is used for each high configuration computer. We consider four other strategies as follows.

OLHD: Use one ordinary LHD [5] with 42 runs. Randomly assign 6 runs to each low configuration computer and 8 runs to each high configuration computer.

OneSLHD: Use one SLHD [6] with 6 slices of 6 runs. Assign each slice to one computer.

TwoSLHD: Use two SLHDs, one with 3 slices of 6 runs and the other with 3 slices of 8 runs. Assign each 6-run slice to one low configuration computer and each 8-run slice to one high configuration computer.

FSD: Use a flexible sliced design with 3 slices of 6 runs and 3 slices of 8 runs [11]. Assign each 6-run slice to one low configuration computer and each 8-run slice to one high configuration computer.

We consider two circumstances. In the first situation, $\hat{\mu}$ is computed using all experiments. In the second situation, one low configuration computer fails and $\hat{\mu}$ is computed from runs of two low configuration computers and three high configuration computers. We compare the root mean square errors(RMSE) of $\hat{\mu}$ under the two circumstances from the five

schemes. The RMSE is defined as follows.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\mu}_i - \mu)^2}$$

Here, N is the times we repeat to compute the RMSE. Four examples are provided to show the advantage of the USLHD scheme.

A. EXAMPLE 1

The first model is:

$$\text{F1: } f(x) = \log(x_1x_2x_3x_4x_5),$$

where x is uniformly distributed on $[0, 1]^5$ [14]. This function is also used for numerical comparison in [6] and [11]. Table 1 gives the RMSEs of $\hat{\mu}$ under the two circumstances from the five schemes. The results are averaged among 10^4 independent trials. Clearly, OLHD and USLHD perform the best for Situation 1 when all runs are used and USLHD performs the best for Situation 2 when one slice is removed.

TABLE 1. Root mean square errors for F1.

TABLE 1: Root mean square errors for F1

	OLHD	OneSLHD	TwoSLHD	FSD	USLHD
Situation 1	0.0555	0.0641	0.0772	0.0760	0.0550
Situation 2	0.1542	0.0995	0.0974	0.0965	0.0837

B. EXAMPLE 2

We employ the function

$$\text{F2: } f(x) = \log(1/\sqrt{x_1} + 1/\sqrt{x_2})$$

as the computer model, where x is uniformly distributed on $[0, 1]^5$. The RMSEs are given in Table 2. Similarly, OLHD and USLHD perform the best for Situation 1 when all runs are used and USLHD performs the best for Situation 2 when one slice is removed.

TABLE 2. Root mean square errors for F2.

	OLHD	OneSLHD	TwoSLHD	FSD	USLHD
Situation 1	0.0068	0.0077	0.0090	0.0088	0.0068
Situation 2	0.0156	0.0110	0.0110	0.0109	0.0094

C. EXAMPLE 3: BOREHOLE FUNCTION

Our third example is the borehole function, which characterizes the flow of water through a borehole which is drilled from the ground surface through two aquifers [15]. The model formulation is based on assumptions of no groundwater gradient, steady-state flow from the upper aquifer into the borehole and from the borehole into the lower aquifer, and laminar, isothermal flow through the borehole. The output of this model is the flow rate through the borehole, which is computed by the equation

$$\text{F3: } f(x) = \frac{2\pi x_3 (x_4 - x_6)}{\log(x_2/x_1)\left[1 + \frac{2x_7x_3}{\log(x_2/x_1)x_1^2x_8} + x_3/x_5\right]}.$$

Table 3 gives the eight inputs and their ranges.

TABLE 3. Inputs and their ranges of the borehole model.

Factors	Ranges	Unit
x_1: the radius of the borehole	[0.05, 0.15]	m
x_2: the radius of influence	[100, 50000]	m
x_3: the transmissivity of the upper aquifer	[63070, 115600]	m^2/yr
x_4: the potentiometric head of upper aquifer	[990, 1110]	m
x_5: the transmissivity of the lower aquifer	[63.1, 116]	m^2/yr
x_6: the potentiometric head of lower aquifer	[700, 820]	m
x_7: the length of borehole	[1120, 1680]	m
x_8: the hydraulic conductivity of borehole	[1500, 15000]	m/yr

We compute the mean output value using the five schemes and report the RMSEs in Table 4. Seen from the results, OLHD performs the best and USLHD takes the second place for Situation 1, while USLHD is the best for Situation 2.

TABLE 4. Root mean square errors for F3.

	OLHD	OneSLHD	TwoSLHD	FSD	USLHD
Situation 1	1.4211	1.6990	1.5484	1.5641	1.5278
Situation 2	3.2184	1.9672	1.7432	1.7566	1.7223

D. EXAMPLE 4: OTL CIRCUIT FUNCTION

The Output Transformer Less (OTL) circuit function describes an OTL push-pull circuit [16]. OTL is a term used to describe amplifiers that do not have an output transformer. Usually, the output transformer is a major source of distortion. Amplifiers without the output transformer therefore produce cleaner and less distorted sound. The output V_m of the OTL circuit function is the midpoint voltage given the six input factors. The OTL circuit function is given as follows:

$$\text{F4: } V_m = \frac{(V_{b1} + 0.74)\beta(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\beta(R_{c2} + 9)}{(\beta(R_{c2} + 9) + R_f)R_{c1}},$$

where

$$V_{b1} = \frac{12R_{b2}}{R_{b1} + R_{b2}}.$$

The six input factors are shown in Table 5.

TABLE 5. Inputs and their ranges of the OTL circuit model.

Factors	Ranges	Unit
R_{b1}: Resistance b1	[50, 150]	$K - Ohms$
R_{b2}: Resistance b2	[25, 70]	$K - Ohms$
R_f: Resistance f	[0.5, 3]	$K - Ohms$
R_{c1}: Resistance c1	[1.2, 2.5]	$K - Ohms$
R_{c2}: Resistance c2	[0.25, 1.2]	$K - Ohms$
β: Current Gain	[50, 300]	$Amperes$

Tables 6 gives the results on RMSEs. Again, OLHD performs the best and USLHD takes the second place for Situation 1, while USLHD is the best for Situation 2.

TABLE 6. Root mean square errors for F4.

	OLHD	OneSLHD	TwoSLHD	FSD	USLHD
Situation 1	0.0162	0.0191	0.0188	0.0189	0.0169
Situation 2	0.0747	0.0264	0.0242	0.0242	0.0230

Results from the four examples are very consistent. OLHD and USLHD perform the best for Situation 1 when all runs are used. This is probably because OLHD, OneSLHD and USLHD are LHDs, TwoSLHD and FSD are not, and that OneSLHD has fewer total size than other schemes. USLHD performs the best for Situation 2 when one slice is removed. This is probably because USLHD achieves optimal univariate uniformity for both the whole design and the removed batch. Comparing to USLHD, OLHD has inferior uniformity for removed runs, OneSLHD has smaller size, and TwoSLHD and FSD have inferior uniformity for the whole design. To sum it, USLHD has the uniformly best performance.

V. CONCLUSIONS AND DISCUSSION

In this paper, we propose new SLHDs with unequal batch sizes, for which both the whole designs and their slices are LHDs. Numerical results show our designs are desirable in estimating the mean output value of computer experiments that are carried out in batches. Compared to SLHD proposed in [6], our method allows the batch sizes to be unequal. One shortcoming of our method is that it only allows two different batch sizes. Nevertheless, it is possible to apply our method to wider applications. For instance, for experiments with three distinct batch sizes, m_1, m_2 and $m_1 + m_2$, we can construct a USLHD with batch sizes m_1 and m_2 and then merge some batches to obtain $(m_1 + m_2)$-run batches. MATLAB codes for generating USLHDs are available in supplementary materials.

APPENDIX
PROOFS

Proof 1 (Proof of Proposition 1): For $i = 1, \cdots, n_1 - 1$, we have

$$\begin{aligned}\alpha_{i+1} - \alpha_i &= mod(\alpha_i - r, n') + in' - \alpha_i \\ &\geq n' + (n'_1 - n') = n'_1.\end{aligned}$$

Then, for any $i, j = 1, \cdots, n_1$ and $i \neq j$,$|\alpha_i - \alpha_j| \geq n'_1$. Therefore, $\lceil \alpha_i/n'_1 \rceil \neq \lceil \alpha_j/n'_1 \rceil$. It suffices to show $n'_2(l_k - 1) + 1 \leq kn'$ and $n'(k-1) + 1 \leq l_k n'_2$ with $k = 1, \cdots, n_2$. Because l_k is the k-th smallest element of $Z_n \setminus \{\lceil \alpha_1/n'_1 \rceil, \cdots, \lceil \alpha_{n_1}/n'_1 \rceil\}$, we have that $\lceil \alpha_{l_k-k}/n'_1 \rceil \leq l_k - 1$, $\lceil \alpha_{l_k-k+1}/n'_1 \rceil \geq l_k + 1$ and $l_k \geq k$. Firstly, we show the proof of $n'_2(l_k - 1) + 1 \leq kn'$ by two cases.

Case 1: $l_k = k$. It is easy to know it holds.

Case 2: $l_k - k \geq 1$. Clearly,

$$\begin{aligned}\lceil \frac{\alpha_{l_k-k}}{n'_1} \rceil \leq l_k - 1 \Rightarrow \alpha_{l_k-k} &\leq n'_1(l_k - 1) \\ \Rightarrow n'(l_k - k - 1) + 1 \leq \alpha_{l_k-k} &\leq n'_1(l_k - 1) \\ \Rightarrow n'_2(l_k - 1) + 1 &\leq kn'.\end{aligned}$$

Then, $n'_2(l_k - 1) + 1 \leq kn'$ holds. Secondly, from $\lceil \alpha_{l_k-k+1}/n'_1 \rceil \geq l_k + 1$, we have $\alpha_{l_k-k+1} \geq l_k n'_1 + 1$. Then, we prove $n'(k-1) + 1 \leq l_k n'_2$ in cases.

Case 1: $l_k - k = n_1$.

We have $l_k n'_2 = (k + n_1)n'_2 \geq (k-1)n' + 1$.

Case 2: $l_k - k < n_1$.

Clearly,

$$\begin{aligned}\alpha_{l_k-k+1} &\geq l_k n'_1 + 1 \\ &\Rightarrow n'(l_k - k + 1) \geq l_k n'_1 + 1 \\ &\Rightarrow l_k n'_2 \geq (k-1)n' + 1,\end{aligned}$$

which completes this proposition.

Proof 2 (Proof of Theorem 1): From Proposition 1 and the process of generating β, we have $\lceil \gamma_i \rceil \neq \lceil \gamma_j \rceil$ with $i, j = 1, \cdots, n, i \neq j$, where $\gamma_i = \alpha_i/n'_1$ for $i \leq n_1$ and $\gamma_i = \beta_{i-n_1}/n'_2$ for $i > n_1$. So we have $\lceil h_i n \rceil \neq \lceil h_j n \rceil$. Therefore, exactly one entry of h falls within each of the $(0, 1/n], \cdots, ((n-1)/n, 1]$.

Clearly, $\{\lceil \alpha_1/n' \rceil, \cdots, \lceil \alpha_{n_1}/n' \rceil\} = \{1, \cdots, n_1\}$ holds. Then $\{\lceil \alpha_i^{(1)}/(n't_1) \rceil, \lceil \alpha_{t_1+i}^{(1)}/(n't_1) \rceil, \cdots, \lceil \alpha_{(m_1-1)t_1+i}^{(1)}/(n't_1) \rceil\}$ equals to $\{1, \cdots, m_1\}$ for any $i = 1, \cdots, t_1$. So the elements of

$$\{\lceil \alpha^*_{(i-1)m_1+1}/(n't_1) \rceil, \cdots, \lceil \alpha^*_{im_1+1}/(n't_1) \rceil\}$$

are in $\{1, \cdots, m_1\}$ and not equal to each other. Hence, exactly one entry of $\{h_{(i-1)m_1+1}, \cdots, h_{im_1}, \}$ falls within each of the $(0, 1/m_1], \cdots, ((m_1 - 1)/m_1, 1]$. Similarly, Exactly one entry of $\{h_{(j-1)m_2+1+n_1}, \cdots, h_{jm_2+n_1}\}$ falls with each of the $(0, 1/m_2], \cdots, ((m_2 - 1)/m_2, 1]$ for any $j = 1, \cdots, t_2$. Clearly, $\{\lceil \alpha_1/n' \rceil, \cdots, \lceil \alpha_{n_1}/n' \rceil\} = \{1, \cdots, n_1\}$ leads to that exactly one entry of $\{h_1, \cdots, h_{n_1}\}$ falls within each of the $(0, 1/n_1], \cdots, ((n_1 - 1)/n_1, 1]$. Similarly, exactly one entry of $\{h_{n_1+1}, \cdots, h_n\}$ falls within each of the $(0, 1/n_2], \cdots, ((n_2 - 1)/n_2, 1]$. This completes the proof.

Proof 3 (Proof of Theorem 2): We consider the probability mass function of α firstly. For $a \in Z'_n$, $P\{\alpha_1 = a\} = 1/n'$ is satisfied. Then, for $a = n' + 1, \cdots, 2n'$,

$$\begin{aligned}P\{\alpha_2 = a\} &= P\{mod(\alpha_1 - r, n') + n' = a\} \\ &= \sum_{k=0}^{n'_2} P\{r = k\} P\{\alpha_1 \equiv k + a (\text{mod } n')\} = 1/n'.\end{aligned}$$

Similarly, for $i = 3, \cdots, n_1$, $P\{\alpha_i = a\} = 1/n'$, for any $a = n'(i-1) + 1, \cdots, in'$. Then, we consider the probability mass function of β. For $i = 1, \cdots, n_2$ and any $a = n'(i-1) + 1, \cdots, in'$, let $k = \lceil a/n'_2 \rceil$ and L be the number of elements in $\{n'(i-1) + 1, \cdots, in'\} \cap \{n'_2(k-1) + 1, \cdots, kn'_2\}$. Simple analysis shows that $i \leq k \leq i + n'_1$. Then, we have

$$\begin{aligned}\text{P}\{\beta_i = a\} = 1/L * \text{P}\{&k \text{ is the } i\text{-th smallest element of} \\ &Z_n \setminus \{\lceil \alpha_1/n'_1 \rceil, \cdots, \lceil \alpha_{n_1}/n'_1 \rceil\}\}. \quad (1)\end{aligned}$$

When $k = i$, then $L = in'_2 - n'(i-1) = n' - in'_1$. So

$$\begin{aligned}\text{P}\{\beta_i = a\} &= 1/L * \text{P}\{\alpha_1 \geq in'_1 + 1\} \\ &= 1/(n' - in'_1) * (n' - in'_1)/n' \\ &= 1/n'.\end{aligned}$$

And when $k = i + n_1'$, then $L = in' - (k-1)n_2' = in_1' - n_2'(n_1 - 1)$. Thus

$$\begin{aligned}\mathrm{P}\{\beta_i = a\} &= 1/L * \mathrm{P}\{\alpha_{n_1} \le (k-1)n_1'\} \\ &= 1/[in_1' - n_2'(n_1-1)] * [in_1' - n_2'(n_1-1)]/n' \\ &= 1/n'\end{aligned}$$

When $i < k < i + n_1$, the number of elements of the set $\{x : x \le k-1, x \in \{\lceil \alpha_1/n_1' \rceil, \cdots, \lceil \alpha_{n_1}/n_1' \rceil\}\}$ is $k - i$. Namely, $\lceil \alpha_{k-i}/n_1' \rceil \le k-1$. Meanwhile, k is the i-th smallest element, so, $\lceil \alpha_{k-i+1}/n_1' \rceil \ge k+1$. Simple analysis shows that the necessary and sufficient condition of the i-th smallest element $k \in (i, i+n_1)$ is $\lceil \alpha_{k-i} 1/n_1' \rceil \le k-1$ and $\lceil \alpha_{k-i+1}/n_1' \rceil \ge k+1$. Therefore, (1) turns to

$$\begin{aligned}&\mathrm{P}\{\beta_i = a\} \\ &\quad = 1/L * \mathrm{P}\{\lceil \alpha_{k-i}/n_1' \rceil \le k-1, \lceil \alpha_{k-i+1}/n_1' \rceil \ge k+1\} \\ &\quad = 1/L * \mathrm{P}\{\alpha_{k-i} \le n_1'(k-1), \alpha_{k-i+1} \ge kn_1' + 1\}.\end{aligned}$$

We discuss it in three cases.

Case 1: $n_2'(k-1) + 1 \le n'(i-1)$.

In this case, $L = kn_2' - n'(i-1)$. Besides, it is clear that $n_1'(k-1) = n'(k-i) + n'(i-1) - n_2'(k-1) \ge n'(k-i) + 1$, so $\mathrm{P}\{\alpha_{k-i} \le n_1'(k-1)\} = 1$. Besides, $kn' + 1 = n'(k-i) + in' - kn_2' + 1$, then $\mathrm{P}\{\alpha_{k-i+1} \ge kn_1' + 1\} = [kn_2' - n'(i-1)]/n'$. So $\mathrm{P}\{\beta_i = a\} = 1/n'$.

Case 2: $n_2'(k-1) \ge n_1'(i-1)$ and $kn_2' \le in'$.

For this situation, we have $L = n_2'$. Besides,

$$\begin{aligned}&\mathrm{P}\{\beta_i = a\} \\ &= 1/L * \mathrm{P}\{\alpha_{k-i} \le n_1'(k-1), \alpha_{k-i+1} \ge kn_1' + 1\} \\ &= 1/n_2' \sum_{p=1}^{in'-kn_2'+n_2'} \sum_{q=in'-kn_2'+1}^{n'} \mathrm{P}\{\alpha_{k-i} = n'(k-i-1) + p, \\ &\qquad \alpha_{k-i+1} = n'(k-i-1) + q\} \quad (2)\end{aligned}$$

When $in' - kn_2' = 0$, then (2) is expressed as

$$\begin{aligned}\mathrm{P}\{\beta_i = a\} &= 1/n_2' \sum_{p=1}^{n_2'} \sum_{q=1}^{n'} \mathrm{P}\{\alpha_{k-i} = n'(k-i-1) + p, \\ &\qquad \alpha_{k-i+1} = n'(k-i-1) + q\} \\ &= 1/n'.\end{aligned}$$

And when $in' - kn_2' > 0$ and $n_2' + 1 \ge in' - kn_2'$,

$$\begin{aligned}\mathrm{P}\{\beta_i = a\} &= 1/[n'n_2'(n_2'+1)][2 \sum_{p=1}^{in'-kn_2'} (n_2' + 1 - p) \\ &\qquad + ((k+1)n_2' - in' + 1)(n_2'(k+1) - in')] \\ &= 1/n'.\end{aligned}$$

Moreover, when $in' - kn_2' > 0$ and $n_2' + 1 \le in' - kn_2'$. Then,

$$\mathrm{P}\{\beta_i = a\} = 1/[n'n_2'(n_2'+1)][2 \sum_{p=1}^{n_2'} (n_2' + 1 - p)] = 1/n'$$

Case 3: $kn_2' \ge in' + 1$. We have that $L = in' - n_2'(k-1)$ and $kn' + 1 = n'(k-i) + in' - kn_2' + 1 \le n'(k-i)$, so $\mathrm{P}\{\alpha_{k-i+1} \ge kn_1' + 1\} = 1$. Moreover, $n_1'(k-1) = n'(k-i) + n'(i-1) - n_2'(k-1) \ge n'(k-i) + 1$, so $\mathrm{P}\{\alpha_{k-i} \le n_1'(k-1)\} = [in' - n_2'(k-1)]/n'$. Then, $\mathrm{P}\{\beta_i = a\} = 1/n'$.

Clearly, we have $P\{\alpha_i^* = a\} = 1/(n_1 n')$ for any $i = 1, \cdots, n_1, a = 1, \cdots, n_1 n'$ and $P\{\beta_j^* = b\} = 1/(n_2 n')$ for any $j = 1, \cdots, n_2, n = 1, \cdots, n_2 n'$. For any $x \in [0, 1)$ and $i = 1, \cdots, n_2$, let $x_0 = \lceil x n_1 n' \rceil$, so

$$\begin{aligned}\mathrm{P}\{h_i < x\} &= \sum_{a=1}^{a=x_0} \{\mathrm{P}\{\alpha_i^* = a\} \mathrm{P}\{(a-\epsilon)/(n_1 n') < x\}\} \\ &= 1/(n_1 n')[\sum_{a=1}^{a=x_0-1} \mathrm{P}\{(a-\epsilon)/(n_1 n') < x\} \\ &\qquad + \mathrm{P}\{(x_0 - \epsilon)/(n_1 n') < x\}] \\ &= 1/(n_1 n')[x_0 - 1 + 1 - x_0 + x n_1 n'] \\ &= x\end{aligned}$$

Thus, h_i is a U[0, 1) random variable with $i = 1, \cdots, n_1$. Similarly, for $i = n_1 + 1, \cdots, n, x \in [0, 1)$ and $x_0 = \lceil x n_2 n' \rceil$,

$$\begin{aligned}\mathrm{P}\{h_i < x\} &= \sum_{b=1}^{b=x_0} \{\mathrm{P}\{\beta_{i-n_1}^* = b\} \mathrm{P}\{(b-\epsilon)/(n_2 n')\} < x\}\} \\ &= 1/(n_2 n')[\sum_{b=1}^{b=x_0-1} \mathrm{P}\{\epsilon > b - x n_2 n'\} \\ &\qquad + \mathrm{P}\{\epsilon > x_0 - x n_2 n'\}] \\ &= 1/(n_1 n')[x_0 - 1 + 1 - x_0 + x n_2 n'] \\ &= x\end{aligned}$$

Therefore, for $i = 1, \cdots, n$, h_i is a random variable on U[0, 1). Because of the independence, we have $(h_i^{(1)}, \cdots, h_i^{(d)})$ follows the uniform distribution on $[0, 1)^d$ for $i = 1, \cdots, n$. This theorem holds.

REFERENCES

[1] Q. Xu, Y. Yang, Y. Liu, and X. Wang, "An improved latin hypercube sampling method to enhance numerical stability considering the correlation of input variables," *IEEE Access*, vol. 5, pp. 15197–15205, 2017.

[2] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Stat. Sci.*, vol. 4, no. 4, pp. 409–423, 1989.

[3] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. New York, NY, USA: Springer, 2013.

[4] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[5] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[6] P. Z. G. Qian, "Sliced latin hypercube designs," *J. Amer. Stat. Assoc.*, vol. 107, no. 497, pp. 393–399, 2012.

[7] M. Ai, B. Jiang, and K. Li, "Construction of sliced space-filling designs based on balanced sliced orthogonal arrays," *Statistica Sinica*, vol. 24, no. 4, pp. 1685–1702, 2014.

[8] S. Ba, W. R. Myers, and W. A. Brenneman, "Optimal sliced latin hypercube designs," *Technometrics*, vol. 57, no. 4, pp. 479–487, 2015.

[9] H. Xie, S. Xiong, P. Z. G. Qian, and C. F. J. Wu, "General sliced Latin hypercube designs," *Statistica Sinica*, vol. 24, no. 3, pp. 1239–1256, 2014.

[10] J. Yang, H. Chen, D. K. J. Lin, and M.-Q. Liu, "Construction of sliced maximin-orthogonal Latin hypercube designs," *Statistica Sinica*, vol. 26, no. 2, pp. 589–603, 2016.

[11] X. Kong, M. Ai, and K. L. Tsui, "Flexible sliced designs for computer experiments," *Ann. Inst. Stat. Math.*, vol. 70, no. 3, pp. 631–646, 2018.
[12] J. Xu, X. Duan, Z. Wang, and L. Yan, "A general construction for nested Latin hypercube designs," *Statist. Probab. Lett.*, vol. 134, pp. 134–140, Mar. 2018.
[13] R. Durrett, *Probability: Theory and Examples*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[14] S. S. Drew and T. Homem-de-Mello, "Some large deviations results for Latin hypercube sampling," in *Proc. Winter Simul. Conf.*, Dec. 2005, pp. 673–681.
[15] M. D. Morris, T. J. Mitchell, and D. Ylvisaker, "Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction," *Technometrics*, vol. 35, no. 3, pp. 243–255, 1993.
[16] E. N. Ben-Ari and D. M. Steinberg, "Modeling data from computer experiments: An empirical comparison of Kriging with MARS and projection pursuit regression," *Qual. Eng.*, vol. 19, no. 4, pp. 327–338, 2007.

JIN XU received the B.S. and M.S. degree in applied mathematics from the National University of Defense Technology, Changsha, China, in 2012 and 2014, respectively, where he is currently pursuing the Ph.D. degree in applied mathematics. His research interest includes the computer experiment, system evaluation, and data processing.

XU HE received the B.S. degree in mathematics from Peking University in 2007 and the Ph.D. degree in statistics from the University of Wisconsin–Madison in 2012. He is currently an Associate Professor in statistics with the Academy of Mathematics and System Sciences, Chinese Academy of Sciences. He has published papers in the *Journal of the American Statistical Association*, *Annals of Statistics*, *Biometrika*, *Technometrics*, and other journals. His research interests cover computer simulation, experimental design, and uncertainty quantification.

XIAOJUN DUAN received the B.S. and M.S. degrees in applied mathematics and the Ph.D. degree in control science and engineering in 1997, 2000, and 2003, respectively. She is currently a Professor in applied mathematics with the National University of Defense Technology. She has completed three projects funded by the National Natural Science Foundation of China. She has co-published two monographs and over 40 papers. Her research interests cover areas, such as experimental design, system evaluation, and data processing.

ZHENGMING WANG was born in 1962. He received the B.S. and M.S. degrees in applied mathematics and the Ph.D. degree in system engineering in 1982, 1986, and 1998, respectively. He is currently a Professor in applied mathematics with the National University of Defense Technology. He has completed five projects funded by the National Natural Science Foundation of China. He has co-published five monographs (all ranked first) and well over 90 papers. His research interests cover areas, such as experiment evaluation and data fusion, mathematical modeling in tracking data, and image processing.