13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference
13 - 15 September 2010, Fort Worth, Texas

AIAA 2010-9085

Fast Generation of Space-filling Latin Hypercube Sample Designs

Keith R. Dalbey*
Sandia National Labs, Albuquerque, NM, 87123, USA

George N. Karystinos†
Technical University of Crete, Chania, 73100, Greece

Latin Hypercube Sampling (LHS) and Jittered Sampling (JS) both achieve better convergence than standard Monte Carlo Sampling (MCS) by using stratification to obtain a more uniform selection of samples, although LHS and JS use different stratification strategies. The "Koksma-Hlawka-like inequality" bounds the error in a computed mean in terms of the sample design's discrepancy, which is a common metric of uniformity. However, even the "fast" formulas available for certain useful L_2 norm discrepancies require $\mathcal{O}\left(N^2 M\right)$ operations, where M is the number of dimensions and N is the number of points in the design. It is intuitive that "space-filling" designs will have a high degree of uniformity. In this paper we propose a new metric of the space-filling property, called "Binning Optimality," which can be evaluated in $\mathcal{O}\left(N \log(N)\right)$ operations. A design is "Binning Optimal" in base b, if when you recursively divide the hypercube into b^M congruent disjoint sub-cubes, each sub-cube of a particular generation has the same number of points until the sub-cubes are small enough that they all contain either 0 or 1 points. The $\mathcal{O}\left(N \log(N)\right)$ cost of determining if a design is binning optimal comes from quick-sorting the points into Morton order, i.e. sorting the points according to their position on a space-filling Z-curve. We also present a $\mathcal{O}\left(N \log(N)\right)$ fast algorithm to generate Binning Optimal Symmetric Latin Hypercube Sample (BOSLHS) designs. These BOSLHS designs combine the best features of, and are superior in several metrics to, standard LHS and JS designs. Our algorithm takes significantly less than 1 second to generate $M = 8$ dimensional space-filling LHS designs with $N = 2^{16} = 65536$ points compared to previous work which requires "minutes" to generate designs with $N = 100$ points.

Nomenclature

M	the number of input dimensions
N	the number of sample points
$\underline{x}$	a M by 1 input vector
$\mathcal{X}$	the set of N points, $\underline{x}$, in a sample design
$\underline{\underline{X}}$	a N by M sample design matrix, each row represents a point, $\underline{x}^T$
$f(\underline{x})$	a black box function (simulator) or **one** of its output variables
C^M	the unit hypercube $[0,1]^M$
c^M	a M dimensional rectangular subset of C^M
$\mathrm{Vol}\left(c^M\right)$	the hypervolume or "content" of c^M
$E(f)$	the true mean of function $f(\underline{x})$ over C^M
$E_{\mathcal{X}}(f)$	the sample mean of f evaluated at all $\underline{x} \in \mathcal{X}$
$D_p\left(\mathcal{X}\right)$	a L_p discrepancy of the sample design $\mathcal{X}$
$V_q(f)$	a L_q variance of the function $f(\underline{x})$ over C^M
$WD_2\left(\mathcal{X}\right)$	the wrap around L_2 discrepancy of the sample design $\mathcal{X}$
$CD_2\left(\mathcal{X}\right)$	the centered L_2 discrepancy of the sample design $\mathcal{X}$
b	a prime number base, $b = 2$ is used
(t, m, s)	the (t, m, s)-net rating of a design, t =quality rating, $N = b^m$, $s = M$
P	the smallest integer such that $b^{MP} \geq N$
(g, s)	the quality metrics for the degree of binning non-optimality (a different s than in the (t, m, s)-net rating)
g	the generation of the smallest bin size that all contain the same number of points
s	the maximum number of points contained by a bin with edge length b^{-P}
Subscript	
i_X, j_X	indices for the row in matrix $\underline{\underline{X}}$, i.e. the point in $\mathcal{X}$
k	the index for the column of $\underline{\underline{X}}$, i.e. the factor or input dimension

*technical staff, Optimization and UQ, not an AIAA member.
†Assistant Professor, Department of Electronic and Computer Engineering, not an AIAA member.

Copyright © 2010 by the American Institute of Aeronautics and Astronautics, Inc. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Go

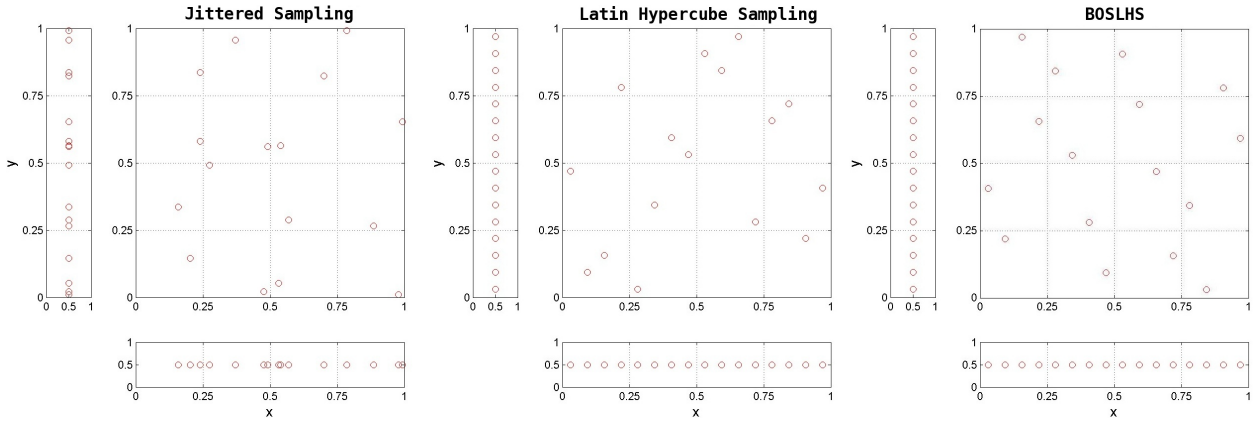

Figure 1. Examples of $N = 16$ points $M = 2$ dimensional designs. From left to right these are: Jittered Sampling (JS), Latin Hypercube Sampling (LHS) with randomly paired dimensions, and Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS). Jittered Sampling is space-filling in the full M dimensional space but not in its 1 dimensional projections. Latin Hypercube Sampling is space-filling in the 1 dimensional projections but not in the full M dimensional space. BOSLHS is space-filling in the full M dimensional space and the 1 dimensional projections; BOSLHS is also symmetric with respect to reflection through the center of the design which reduces correlations between dimensions. Unlike Jittered Sampling, BOSLHS is not restricted to a number of samples, N, that is exponential in the number of dimensions, M.

I. Introduction

Pure random sampling, also referred to as Monte Carlo Sampling (MCS), is the most robust and universally applicable method of uncertainty quantification. Unfortunately, the error in a mean computed by MCS decreases very slowly as the number of samples, N, increases. The "Koksma-Hlawka-like inequality" bounds the error in a computed mean in terms of the sample design's discrepancy, which is a common metric of uniformity. However, even the "fast" formulas available for certain useful L_2 norm discrepancies require $\mathcal{O}\left(N^2 M\right)$ operations, where M is the number of dimensions. Latin Hypercube Sampling (LHS) and Jittered Sampling (JS) both achieve better convergence than standard MCS by using stratification to obtain a more uniform selection of samples, although LHS and JS use different stratification strategies. As illustrated in Figure 1, JS is space-filling in the full M dimensional space but not in the 1 dimensional projections, while LHS is space-filling in the 1 dimensional projections but not in the full M dimensional space.

In this paper, we define "binning optimality," a new metric of the space-filling property which can be evaluated in $\mathcal{O}\left(N \log\left(N\right)\right)$ operations, and present an $\mathcal{O}\left(N \log\left(N\right)\right)$ fast Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) algorithm. BOSLHS combines the best features of JS and LHS to produce designs that are space-filling in the full M dimensional space **and** the 1 dimensional projections (see Figure 1). In addition, BOSLHS designs are symmetric with respect to reflection through their center, which reduces correlation between dimensions. Experimental results show that BOSLHS designs are superior to conventional JS and LHS in several quality metrics. Compared to other algorithms which are reported in the literature to require "minutes" to generate a $M = 10$ dimensional space-filling LHS designs with $N = 100$ points, our BOSLHS algorithm can generate $N = 2^{16} = 65536$ point $M = 8$ dimensional designs in significantly under a second. Our algorithm is currently limited to a power of 2 dimensions, however high quality LHS designs with non-power of 2 dimensions can be obtained by discarding excess dimensions, and the degree of binning non-optimality can be used to quickly screen candidate combinations of dimensions.

A. Background

A common standard case within the broad field of sample design is that of Uniform Design (UD). Let $f(\underline{x})$ be a black box function of M different inputs, $\underline{x}$, with one or more outputs, whose domain is the unit hypercube $C^M = [0,1]^M$, i.e., $\underline{x} \in C^M$. The objective of uniform design is to select a sample design, $\mathcal{X}$, that is appropriate for the task of integrating (with respect to uniformly weighted $\underline{x}$) and/or constructing a response surface of one or more of the function's outputs over the unit hypercube. *A priori*, we do not know which regions in input space are most important, i.e. cause the most significant variation in the output, nor do we know which of the M inputs are more important for which outputs. The subset of important inputs could potentially vary among the different outputs. $\mathcal{X}$ should be selected in such a way as to produce an accurate representation of the output of $f(\underline{x}) \;\; \forall\, \underline{x} \in C^M$. Each point in $\mathcal{X}$ is represented by a row of the N by M design matrix $\underline{\underline{X}}$.

High quality uniform designs are important because

- For many problems with non-uniform distributions of inputs, an appropriate sample design can be obtained from a design for inputs uniformly distributed over the unit hypercube by a suitable combination of mappings (frequently through a cumulative distribution function, CDF) and/or importance sampling.

- If access to a supercomputing cluster is available, it may take significantly less time to concurrently execute simulations at a preselected set of sample inputs than to use a "sequential design," i.e. a set of input points chosen sequentially between runs based on the knowledge obtained from simulations at earlier points in the sequence.

- It may also be desired to use a cluster to concurrently evaluate $f(\cdot)$ at sequential batches of inputs, and check the output between batches to determine if "enough" data has been obtained to satisfy some stopping criteria. It is reasonable to use Uniform Design to generate the final design, $\mathcal{X}_{fin}$, in such a sequence. Here "final" means having the maximum number of samples/runs that is feasible given the available computational resources. Successively earlier members of the sequence could be obtained by a separate pruning process.

Since $\underline{x}$ is uniformly distributed over the unit hypercube, the samples in $\mathcal{X}$ should also be "uniformly distributed" over the unit hypercube. A quantitative measure of uniformity is required, and discrepancy is one such popular metric.

B. Discrepancy

Conceptually, a sample design's discrepancy is an appropriate norm of the difference between the number of points per sub-volume and a uniform smearing of points. This norm is taken over all M-dimensional hyper-rectangular sub-volumes within the unit hypercube. As such, discrepancy is a measure of the non-uniformity of a sample design. Let $|\mathcal{S}|$ denote the number of points in a set, $\mathcal{S}$, then discrepancy

$$D\left(\mathcal{X}\right) = \left|\left| \frac{\left|\mathcal{X} \bigcap c^M\right|}{N} - \mathrm{Vol}\left(c^M\right) \right|\right| \tag{1}$$

where $||\cdot||$ represents an appropriate norm over all M dimensional rectangular subsets, c^M, of C^M. Notationally, this compares a subset's fraction of the total number of points to its fraction of the total volume. The more uniform a sample design $\mathcal{X}$ is, the smaller its discrepancy will be.

Using a L_p-discrepancy as a measure of uniformity is popular is because it can be used to bound the error resulting from the numerical integration of a function $f\left(\underline{x}\right)$. This is the "Koksma-Hlawka-like inequality" which Hickernell[1] proved

$$|E(f) - E_{\mathcal{X}}(f)| \leq D_p\left(\mathcal{X}\right) V_q(f) \qquad \left(p^{-1} + q^{-1} = 1\right), \tag{2}$$

here $E(f)$ is the expectation of the function $f(\underline{x})$

$$E(f) = \int_{C^M} f\left(\underline{x}\right)\ d\underline{x},$$

$E_{\mathcal{X}}(f)$ is the sample mean of $f(\underline{x})$

$$E_{\mathcal{X}}(f) = \frac{1}{N} \sum_{i_X=1}^{N} f\left(\underline{x_{i_X}}\right),$$

$D_p\left(\mathcal{X}\right)$ is the L_p discrepancy of sample design $\mathcal{X}$, and $V_q(f)$ is the L_q variance of function $f(\underline{x})$.

While this bound holds for all L_p-discrepancies, the choice of which discrepancy to use to measure uniformity is an important one. Fang, Li, and Sudjianto note that both the star discrepancy and the star L_2-discrepancy are unsuitable for searching UDs[2]. The star discrepancy (for which all c^M have the point $\underline{0}$ as one of their corners) is insufficiently sensitive and the star L_p-discrepancy ($p < \infty$) ignores differences $\left|\left| \frac{\left|\mathcal{X} \bigcap [\underline{0}, \underline{x})\right|}{N} - \mathrm{Vol}\left([\underline{0}, \underline{x})\right) \right|\right|$ on any low-dimensional subspace. That the lower dimensional structure of a design is more important than the high dimensional structure is known from the so-called *hierarchical ordering principle* of Wu and Hamada[3]: lower-order effects are more likely to be important than higher-order effects; main effects are more likely to be important than interactions; and effects of the same order are equally likely to be important.

Furthermore, because the origin plays a special role, the star L_p-discrepancy is not invariant under rotating coordinates of points. This makes it necessary to place additional restrictions on acceptable measures of uniformity. The following requirements are paraphrased from Fang, Li, and Sudjianto[2]:

1. invariance under permutation of factors and/or runs;

2. invariance under coordinate rotation;

3. ability to measure not only uniformity of $\mathcal{X}$ over C^M but also the projection uniformity of $\mathcal{X}$ over C^u, where u is a non-empty subset of $\{1, ..., M\}$;

4. some reasonable geometric meaning;

5. easy to compute;

6. applicability in the Koksma-Hlawka-like inequality, Eqn. (2) above;

7. consistency "with other criteria in experimental design." This vague statement means that criteria 1-6 are necessary but **not** sufficient to define an acceptable metric of uniformity.

Fang, Li, and Sudjianto[2] further state that Hickernell[1] proved that the centered L_2-discrepancy and wrap-around L_2-discrepancy satisfy requirements 1 through 6.

The wrap around L_2-discrepancy, $WD_2(\mathcal{X})$, proposed by Hickernell[4] is a modified L_2 discrepancy defined as in Eqn. (1) where the L_2 norm is used and the definition of the set of subsets c^M of C^M has been expanded to include all hyper-rectangles which can be drawn on a periodic version of C^M. In other words, it includes rectangles with an edge that starts at coordinate $x_0 < 1$ increases to 1, wraps around to 0 and the increases to $x_1 < x_0$.

The wrap-around L_2-discrepancy satisfies the "easy to compute" requirement of Fang, Li, and Sudjianto[2] because its square can be computed via the following equivalent analytical form

$$\begin{aligned}(WD_2(\mathcal{X}))^2 &= -\left(\frac{4}{3}\right)^M + ... \\ &\quad \frac{1}{N^2}\sum_{j_X=1}^{N}\sum_{i_X=1}^{N}\prod_{k=1}^{M}\left(\frac{3}{2} - |X_{j_X,k} - X_{i_X,k}|\left(1 - |X_{j_X,k} - X_{i_X,k}|\right)\right)\end{aligned} \tag{3}$$

Note that Eqn. (3) requires $\mathcal{O}\left(N^2 M\right)$ operations to evaluate. The centered L_2 discrepancy also has an analytical form similar to Eqn. (3)

$$\begin{aligned}(CD_2(\mathcal{X}))^2 &= \left(\frac{13}{12}\right)^M - \frac{2}{N}\sum_{i_X=1}^{N}\prod_{k=1}^{M}\left(1 + \frac{1}{2}\left|X_{i_X,k} - \frac{1}{2}\right| + \frac{1}{2}\left|X_{i_X,k} - \frac{1}{2}\right|^2\right) ... \\ &\quad + \frac{1}{N^2}\sum_{j_X=1}^{N}\sum_{i_X=1}^{N}\prod_{k=1}^{M}\left(1 + \frac{1}{2}\left|X_{j_X,k} - \frac{1}{2}\right| + \frac{1}{2}\left|X_{i_X,k} - \frac{1}{2}\right| - \frac{1}{2}|X_{j_X,k} - X_{i_X,k}|\right)\end{aligned} \tag{4}$$

Although Eqn. (4) is slightly more expensive to compute than Eqn. (3), the centered L_2 discrepancy is generally the more appropriate metric for sample designs unless the inputs are periodic variables, i.e., unless the variables wrap-around.

Unfortunately, $\mathcal{O}\left(N^2 M\right)$ operations is still quite expensive when the number of sample points, N, is large. Since this can prohibit the strategy of randomly generating a large number of sample designs and selecting the one with the lowest discrepancy, it is desirable to find a faster way to generate uniform designs. Some other common design quality metrics are:

- **the t quality metric:** (lower is better, worst possible is $t = m$) when a design is considered to be a (t, m, s)-net; in our notation, $M = s$ and $N = b^m$ for a base b, in what follows we use $b = 2$; the cost of computing t is $\mathcal{O}\left(b^m s \text{ nchoosek}(m - t + 1 + s, s)\right)$, where $\text{nchoosek}(n, k)$ is the number of combinations of n items taken k at a time. Niederreiter[5] provides an excellent account of quasi-Monte Carlo methods, and pays special attention to (t, m, s)-nets generated from $t - s$ sequences;

- **the "coverage":** (larger is better) of the design; Hemez, Atamturktur, and Unal[6] define coverage as the ratio of the content (hyper-volume) of the convex hull of $\mathcal{X}$ to the content of the domain of inputs; for small M this can be quickly computed, even for large N, by using an algorithm such as qhull, however, the cost quickly grows with dimension;

- **the condition number of the sample design's correlation matrix:** (smaller is better, a condition number of 1 means the input dimensions are uncorrelated/orthogonal with respect to each other) the covariance matrix is defined as $\left(\underline{\underline{X}} - 0.5\right)^T \left(\underline{\underline{X}} - 0.5\right)$; the correlation matrix can be obtained from it by normalizing with respect to the diagonal entries; the cost of the required matrix multiplication is linear in N, specifically $\mathcal{O}\left(M^2 N\right)$; the cost of computing the condition number of an M by M matrix is generally negligible for typical values of M; Cioppa and Lucas[7] used this as part of the criteria for their "computer-intensive algorithm to generate Latin hypercubes (LHs) that have good space-filling properties and are nearly orthogonal;" however, orthogonality by itself is not a sufficient metric of the "goodness" of a design,

Part of what made Cioppa and Lucas[7] LHS designs (not the generation of them) "efficient" is their space-filling property. Space-filling designs have a high degree of uniformity, so generating space-filling Latin Hypercubes has long been an active area of research[7–20].

II. A New Space-Filling Metric: Binning Optimality

The literature on the topic of space-filling designs is quite extensive. While *maximin*[‡], *minimax*[§], and some other types of *optimal designs*[¶] are unanimously agreed to be space-filling designs, notably absent from the literature is a technical definition of the space-filling property. The most specific definition we could find was this quote from Lin et al[21]:

Computer experiments commonly use space-filling designs. As the number of factors increases, the sparsity of the design points increase. Space-filling designs place all the points about the same distance (quite far) apart.

Motivated by this lack of technical specification and a desire to quantify the space-filling property, we propose the Boolean metric of *binning optimality* which can be computed efficiently even for large designs.

Let b be a prime number ≥ 2, and P be defined as

$$P = \text{ceil}\left(\frac{\log_b(N)}{M}\right)$$

Then, a design is "binning optimal" with respect to base b if 2 conditions are met.

1. When C^M is divided into an uniform grid of cube bins with volume $vol = b^{-PM}$ no bin contains more than 1 point.
2. When C^M is divided into an uniform grid of cube bins with volume $vol = b^{-(P-1)M}$ (i.e. one generation larger) every bin contains the same number of points.

Binning optimality can be computed efficiently in the following way

1. compute the bin identifiers (henceforth "ids") of each point at depths $d = P$ and $d = P - 1$
2. tally the number of occurrences of each bin id between 0 and $b^{dM} - 1$; quick-sorting (which costs $\mathcal{O}(N\, log(N))$ operations) the bin ids facilitates this process
3. check if the tallied number of points per bin satisfy the two requirements for binning optimality

One way to assign ids to bin is

$$\underline{\text{bin}} = \text{floor}\left(b^d \underline{\underline{X}}\right) \underline{\beta} \tag{5}$$

where $\underline{\beta} = \left[b^0, b^1, b^2, ..., b^{M-1}\right]^T$. This method costs $\mathcal{O}\left(NM\right)$ operations and produces bin ids that are **not** locality preserving. In rough terms, a bin numbering system is locality preserving if sorting by the identifiers induces sequences of bins that are close to each other in physical space.

A second and slightly more expensive option, which costs $\mathcal{O}\left(NMP\right)$ operations, is to compute locality preserving bin ids, for example the index of the point's location on a space-filling curve (SFC). If $b = 2$, two possible candidates for the SFC are the Hilbert curve and the Z-curve (also known as a Morton-order curve). The Hilbert curve is known to possess better locality preserving properties than the Z-curve. However, Z-curve indices are significantly faster to compute. For those wishing to use Hilbert curve indices, we recommend Moore's "fast" C program[22]. For the purpose of illustration, the following MATLAB code computes Z-curve ids

[‡] A maximin design maximizes the minimum distance between points

[§] A minimax design minimizes the maximum distance between points

[¶] An optimal design is a design found by optimizing for some criteria. Some examples of space-filling criteria are maximum entropy, minimum Integrated Mean Square Error (IMSE), minimum Audze-Eglais potential energy, and minimum discrepancy[16].

```
function z=ComputeZId(X,nBitsPerDim)
%0<x<1 is required for the matrix of N points "X" in
%all M dimensions
    [N,M]=size(X);
    if(nargin<2)
        nBitsPerDim=max(ceil(log2(N)/M),1);
    end
    TwoToM=2^M;
    X=floor(X*(2^nBitsPerDim));
    xbittozbit=2.^(0:M-1)';
    z=bitget(X,1)*xbittozbit;
    for iBitPerDim=2:nBitsPerDim;
        xbittozbit=xbittozbit*TwoToM;
        z=z+bitget(X,iBitPerDim)*xbittozbit;
    end
end
```

The Z-curve ids needed to check the first condition for binning optimality can be computed as `z=ComputeZId(X,P);`. These are then sorted. The bin ids to check the second condition can then be computed by `z=floor(z*(2^-M));`. One advantage of using locality preserving bin ids is that you only need to sort the ids once, i.e for first condition. This ordering can be reused when checking the second condition. Also, as will be seen in section IV. below, sorting by locality preserving bin ids, particularly the Z-curve ids, is extremely useful in the fast generation of space-filling Latin hypercube designs.

Four subtle issues qualify the applicability of binning optimality:

First, binning optimality is a rather weak sort of optimality. It verifies that there are the "right" number of points in bins with an edge length of b^{-P} or larger. It does **not** verify that the subgrid-scale location of each point is optimal in any sense. Said another way, binning optimality does not imply the optimal spacing of **bins** that contain points for bins of edge length b^{-P} or smaller. Therefore there is no uniqueness guarantee and some other means must choose between binning optimal designs.

Second, although maximal spacing of the smallest bins is not measured by binning optimality, we can use SFC index bin ids to engineer locally (within bins of edge length $2^{-(P-1)}$) maximal spacing of bins of edge length 2^{-P} into the LHS construction procedure. The Z-curve's simpler mapping from SFC index to $\Re^M$ makes it more conducive to implementing this locally maximal spacing than the Hilbert SFC.

Third, a design which is binning optimal is space-filling (in the limit of $N \to \infty$), however, a design need not be binning optimal to be space-filling.

Fourth, although binning optimality is a binary criteria, the *degree* of binning non-optimality can be used to compare designs that are *not* binning optimal. The degree of binning non-optimality can be represented by a pair of integers (g, s). The first, g, is the minimum number of generations above bins of the smallest size, i.e. bins of edge length b^{-P}, at which all bins will contain the same number of points, these bins have edge length b^{-P+g}. The second, s, is the maximum number of points, s, contained by any of the bins of the smallest size. The smaller each of these numbers is, the more space-filling the design. A binning optimal design has $(g, s) = (0, 1)$. The degree of binning non-optimality can also be determined in $\mathcal{O}\left(N \log(N)\right)$ operations.

III. Prior Work on Sampling Methods

This section reviews prior work done on sampling methods to motivate a new approach proposed in Section IV..

A. Tensor Product Sampling (TPS)

When sampling in one dimension, a reasonable first thought is to equally space points, i.e. put $N = q$ samples at the centers of q equal bins. Indeed, this obtains both minimum discrepancy and binning optimality. When sampling in multiple dimensions, a reasonable first thought is to to use a tensor product grid of points. While a tensor product sampling (TPS) grid is binning optimal, it also has large wrap-around and centered L_2 discrepancies.

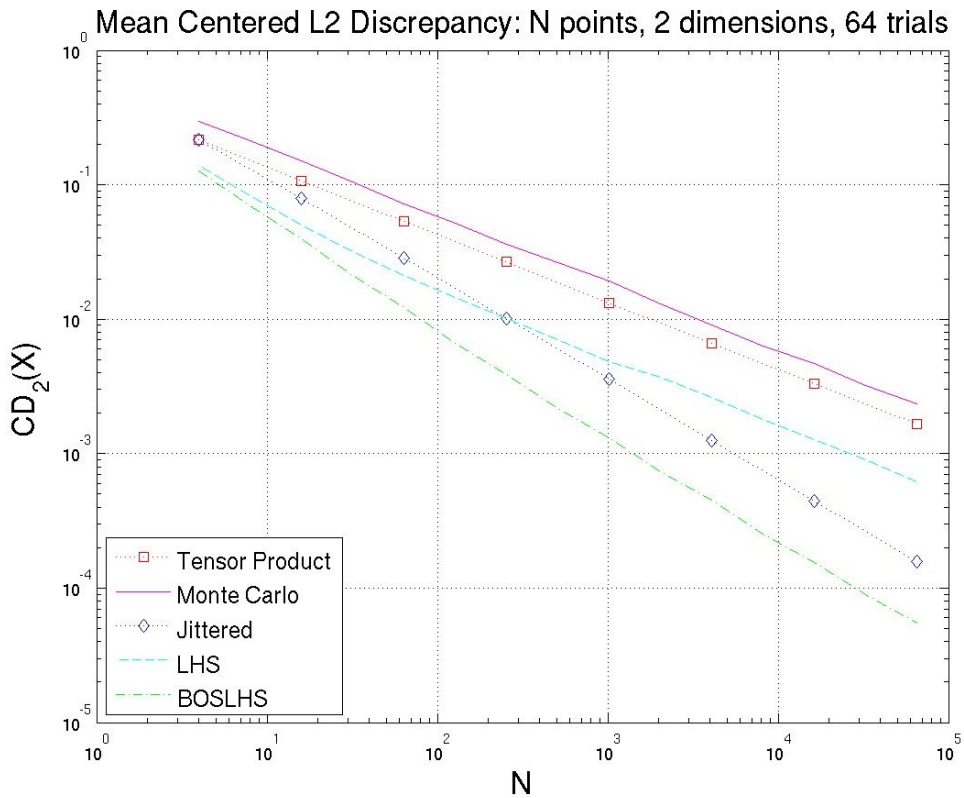

Figure 2. Plots of the centered L_2-discrepancy as a function of the number of samples for various methods of sampling in 2 dimensions. In order of approximately decreasing discrepancy, these methods are: Monte Carlo Sampling (MCS), Tensor Product Sampling (TPS), Jittered Sampling (JS), cell centered Latin Hypercube Sampling (LHS) with randomly paired dimensions, and Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) generated by our algorithm.

The slow decrease in discrepancy for tensor product sampling can be explained by the hierarchical ordering principle and requirement 3 of Fang, Li, and Sudjianto[2] presented in Subsection I.. When projected into a u dimensional subspace there are only q^u unique points with q^{M-u} duplicates of each of the unique points, and lower dimensional projections are more important. The wrap-around L_2-discrepancy for a tensor product grid $\mathcal{X}_{TPS}$ is given by

$$WD_2\left(\mathcal{X}_{TPS}\right) = \sqrt{-\left(\frac{4}{3}\right)^M + \left(\frac{4}{3} + \frac{1}{6}N^{-2/M}\right)^M} \tag{6}$$

The centered L_2 discrepancy for tensor product sampling in $M = \{2, 4, 8\}$ dimensions is plotted as the dotted red line with squares in Figures 2, 3 and 4 respectively.

B. Monte Carlo Sampling (MCS)

Monte Carlo Sampling (MCS) is the name given to the practice of generating design points through purely random sampling. Drawing from a uniform distribution naturally produces sample designs that are approximately uniform. Unfortunately, the standard error in a mean computed by random sampling converges as $\frac{\sigma}{\sqrt{N}}$, which means that in general, 1 million samples will be needed for 3 significant figures of accuracy. Fang and Ma[23] obtained a similar result (the upper bound on the error is proportional to $N^{-1/2}$) by deriving the following formula for Monte Carlo's expected wrap-around L_2- discrepancy

$$E\left(WD_2\left(\mathcal{X}_{MCS}\right)\right) = \sqrt{\frac{1}{N}\left(\left(\frac{3}{2}\right)^M - \left(\frac{4}{3}\right)^M\right)} \tag{7}$$

Note that for more than two dimensions, the discrepancy for MCS decreases at a faster rate than does the discrepancy for TPS. This can be readily seen for $M = \{2, 4, 8\}$ dimensions in Figures 2, 3 and 4 respectively, where the centered L_2 discrepancy for MCS is plotted as the solid magenta line. An obvious way to improve the quality of Monte Carlo designs is to constrain the random sampling to be more uniform. This has been and continues to be the focus of much active research.

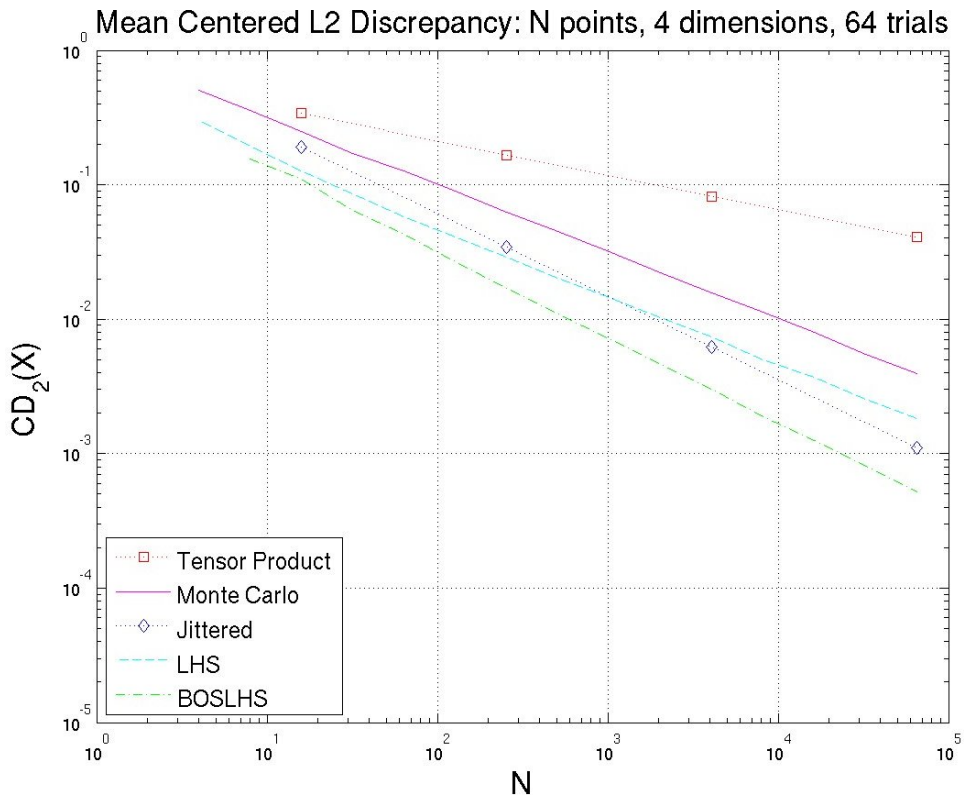

Figure 3. Plots of the centered L_2-discrepancy as a function of the number of samples for various methods of sampling in 4 dimensions. In order of approximately decreasing discrepancy, these methods are: Tensor Product Sampling (TPS), Monte Carlo Sampling (MCS), Jittered Sampling (JS), cell centered Latin Hypercube Sampling (LHS) with randomly paired dimensions, and Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) generated by our algorithm.

C. Jittered Sampling (JS)

"Jittered Sampling" (JS) is a hybrid of tensor product and Monte Carlo sampling. An example of JS is the design resulting from perturbing each point in a tensor product grid by a random amount equal to $(\alpha(w-0.5))/q$ where $w \sim \mathcal{U}[0,1]$, α scales the magnitude of the perturbation, and q is the number of bins in each direction. The expected wrap around L_2 for this design is

$$\begin{aligned} E\left(WD_2\left(\mathcal{X}_{JS}\right)\right) &= \left(-\left(\frac{4}{3}\right)^M + \left(\frac{4}{3} + \frac{(1-\alpha)^2}{6N^{2/M}}\right)^M \ldots \right. \\ &\quad \left. -\frac{1}{N}\left(\frac{3}{2} - \frac{\alpha}{3N^{1/M}} + \frac{\alpha^2}{6N^{2/M}}\right)^M + \frac{1}{N}\left(\frac{3}{2}\right)^M\right)^{\frac{1}{2}} \end{aligned} \tag{8}$$

A JS design is binning optimal and its discrepancy for $\alpha = 1$ decreases at a faster rate than that of MCS. This can be readily seen for $M = \{2, 4, 8\}$ dimensions in Figures 2, 3 and 4 respectively, where the centered L_2 discrepancy for JS is plotted as the dotted blue lines with diamonds. A disadvantage of JS is the number of samples must be exponential in the number of input dimensions, i.e. $N = q^M$ (assuming q is the same for all dimensions).

D. Latin Hypercube Sampling (LHS)

"Latin Hypercube Sampling" (LHS) was developed by McKay et al.[24], and later Stein[25] showed that the variance of the LHS sample mean is asymptotically (i.e. in the limit of a large number of samples) lower than traditional MCS. Owen[26] extended Stein's work to prove a central limit theorem for LHS. LHS is similar to JS in that it achieves better convergence than MCS through a "better" choice of samples. In fact, LHS is equivalent to JS when there is only one dimension. Both involve stratifying the distribution of the random variable, and selecting randomly inside each stratum. However, they differ in the multidimensional case. JS is a perturbation on a full tensor product grid of samples. In LHS, each dimension is stratified separately and then values from each dimension are paired to form the coordinates of points in the M-dimensional sample space. McKay et al.[24] used a random pairing of dimensions, this type of LHS can be implemented as follows.

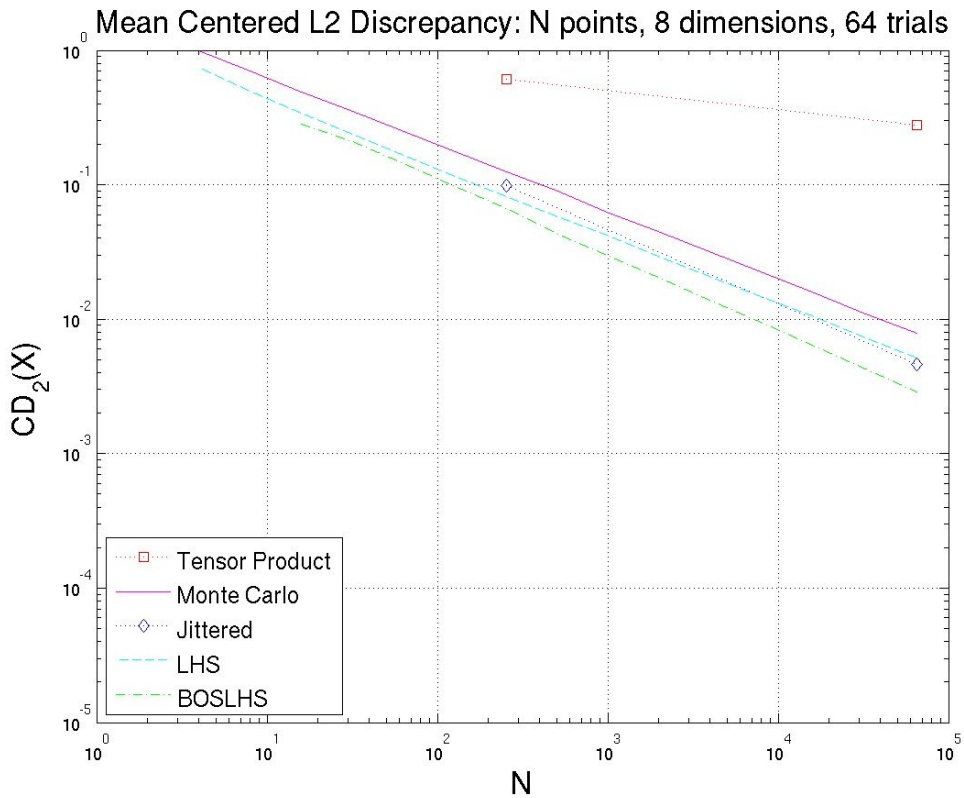

Figure 4. Plots of the centered L_2-discrepancy as a function of the number of samples for various methods of sampling in 8 dimensions. In order of approximately decreasing discrepancy, these methods are: Tensor Product Sampling (TPS), Monte Carlo Sampling (MCS), Jittered Sampling (JS), cell centered Latin Hypercube Sampling (LHS) with randomly paired dimensions, and Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) generated by our algorithm.

1. For each random direction (random variable or input), divide that direction into n bins of equal probability.
2. Select one random value in each bin.
3. Divide each bin into 2 bins of equal probability; the random value chosen above lies in one of these sub-bins.
4. Select a random value in each sub-bin without one.
5. Pair the newly generated coordinates to form points in the multidimensional space.
6. Repeat steps 3, 4 and 5 until desired level of accuracy is obtained.

The advantage of LHS is can be explained in terms of reducing the discrepancy. The *hierarchical ordering principle* states that low order effects are more likely to be important, that is it is crucial to have samples uniformly distributed in low dimensional projections of the sample space. Using a LHS design with samples placed at bin centers obtains optimal uniformity in each **one-dimensional projection**, i.e. in the lowest, and therefore most important, dimensional projections. This allows LHS to readily capture the non-linear influence of a relative few dominant input variables.

Fang and Ma[23] derived the following analytical formula for the expected wrap-around L_2-discrepancy for cell centered LHS with randomly paired dimensions

$$E\left(WD_2\left(\mathcal{X}_{LHS_{random}}\right)\right) = \sqrt{\frac{1}{N}\left(\frac{3}{2}\right)^M - \left(\frac{4}{3}\right)^M + \left(1-\frac{1}{N}\right)\left(\frac{4}{3}-\frac{1}{6N}\right)^M} \tag{9}$$

The centered L_2 discrepancy for cell centered LHS with $M = \{2, 4, 8\}$ randomly paired dimensions can be seen in Figures 2, 3 and 4 respectively, where it is plotted as the dashed aqua line. In these plots, LHS is everywhere lower than MCS, but after the first few samples it has the same slope as MCS. Although LHS has an early advantage over JS, the latter's steeper slope will eventually overtake LHS as the number of samples becomes large.

Centered L_2-Discrepancy for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	0.155417	0.192171	0.361553		
16	0.109056	0.126985	0.255332	0.19617	0.341986
32	0.066096	0.0841458	0.18077		
64	0.0426463	0.058953	0.129307		
128	0.0273056	0.040291	0.0913536		
256	0.0171111	0.029254	0.0640173	0.0348634	0.164806
512	0.0107588	0.0206964	0.0452712		
1024	0.00714498	0.0148297	0.0317103		
2048	0.00466168	0.0102545	0.0225104		
4096	0.00298355	0.00737347	0.0167189	0.00623625	0.0816321
8192	0.00188375	0.0050792	0.0111803		
16384	0.00124693	0.00369942	0.00819728		
32768	0.000820843	0.00253052	0.00564142		
65536	0.000526777	0.00184354	0.00402392	0.00110992	0.0407197

Table 1. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the Centered L_2-Discrepancy (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

LHS also allows for better representation of all of the input variables through more even coverage of the hyper-dimensional input space. However, achieving this, and hence the increase in speed of convergence, largely depends on the quality of the pairing of coordinates from the different dimensions used to form points in the multidimensional input space. Consequently, any estimate of the asymptotic error would need to specifically account for how the dimensions are paired.

To determine the minimum possible centered L_2 discrepancy for $M = 2$ dimensional cell centered LHS designs with $N = 4, 8$ points, we performed an exhaustive search. Both designs were $(0, m, s)$-nets, orthogonal, binning optimal and symmetric. In symmetric designs, each sample point has a counterpart which is its reflection through the center of the design. Symmetry is known to help reduce the correlation between input dimensions. Due to there being $16! \approx 2.09 \times 10^{13}$ different cell-centered LHS designs in $M = 2$ dimensions, it was not possible to conduct and exhaustive search for $N = 16$ points. However, we were able to exhaustively search among 16 point symmetric LHS designs. The best of these designs was also an $(0, m, s)$-net, orthogonal, and binning optimal. Based on these 3 data points, the centered L_2 discrepancy for minimum discrepancy designs in 2 dimensions appears to decay as approximately $\mathcal{O}(N^{-0.943})$.

Unfortunately, solving the hyper-dimensional pairing problem efficiently is non-trivial. A common practice is to randomly generate a "large" number of sets of pairings and the selecting the best, according to some criteria, set. Some common criteria are

- maximizing the minimum, abbreviated as maximin, (of some measure of) distance between sample points
- minimizing the maximum, abbreviated as minimax, (of some measure of) distance between sample points
- minimizing the correlation between coordinates of points in input space

Note that the first 2 of these 3 criteria is that the design be space-filling, and the third targets orthogonality.

The pairing algorithm of Iman and Conover[27] which is employed in Sandia National Labs' non-cell centered "LHS" package[28,29] was designed to (approximately) induce user specified correlations between input dimensions. It does not attempt to minimize discrepancy. Despite this, empirical experiments indicate that requesting an LHS with uncorrelated uniform random variables appears to have an $\mathcal{O}(N^{-\frac{1}{2}})$ discrepancy "line" (not shown) that is slightly lower than randomly paired cell centered LHS.

Owen's[30] method of pairing attempts to minimize the root mean square of the correlations between different dimensions. Owen reported that the rate of convergence in that "error metric" for his method empirically appeared to be

$$\text{error}_{\text{LHS}_{\text{Owen}}} \sim \mathcal{O}\left(N^{-\frac{3}{2}}\right). \tag{10}$$

Wrap Around L_2-Discrepancy for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	0.264011	0.263119	0.486157		
16	0.145193	0.16434	0.340079	0.308026	0.643404
32	0.0983049	0.108923	0.238759		
64	0.0601434	0.0737442	0.173591		
128	0.0390652	0.0506619	0.12265		
256	0.0251623	0.0357784	0.0874663	0.0599227	0.316115
512	0.0163723	0.0254185	0.060521		
1024	0.0106821	0.0179332	0.0437154		
2048	0.0070766	0.0124202	0.0298354		
4096	0.00463723	0.00887728	0.0223775	0.0111964	0.157365
8192	0.00297308	0.00630293	0.0150964		
16384	0.00195182	0.00446912	0.0107136		
32768	0.00128037	0.00315229	0.00766298		
65536	0.000833047	0.0022078	0.00548722	0.00202252	0.0785962

Table 2. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the Wrap Around L_2-Discrepancy (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

However, orthogonal LHS designs have zero correlation between different dimensions and simply being orthogonal is not a sufficient condition for a design to have zero (or even minimum) error in a computed mean. Indeed, tensor product sampling, which is generally considered to be a very poor sampling method (evidenced by a slow rate of decrease in discrepancy), produces designs that are orthogonal.

Morris and Mitchell[10] developed an algorithm based on simulated annealing to generate good maximin designs. Unfortunately finding these pairings proved to be prohibitively expensive for large numbers of points in high numbers of dimensions. Consequently, finding good pairings efficiently is an area of active research.

Ye et al[11] proposed restricting the set of candidate sample point designs to those which are symmetric. Tang[31] used orthogonal arrays to generate LHS designs. Loeppky et al[32] proposed a Gaussian Process based sequential updating strategy to add batches of points to existing orthogonal array based LHS designs. Jin et al[33] claim that their enhanced stochastic evolutionary algorithm is able to find a good 100 point 10 dimensional LHS design in a matter of minutes, and contrasted this against the "hours" spent by the algorithm of Ye et al[11]. However the computational cost of determining a good design with a significantly larger number of samples has remained out of reach until now.

IV. Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS)

Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) combines the best features of JS and regular LHS, i.e., it is space-filling in the full M dimensional space and each of the M dimensions taken individually. In addition, BOSLHS designs are symmetric with respect to reflection through the center of the design, which reduces correlations between dimensions. In this section, we present an geometric algorithm that will generate a N point BOSLHS design in $\mathcal{O}(N \log(N))$ operations by quick-sorting SFC index bin ids as was discussed in Section I..

The algorithm can be outlined as follows

1. Start with $n = 2M$ points that are well distributed in $(0,1)^M$.
2. Select $\frac{n}{2}$ of the coordinates in each dimension other than the first to negate in such a way as to obtain n points that are well distributed in $(0,1) \otimes (-1,1)^{M-1}$.
3. Add n mirror points in $(-1,0) \otimes (-1,1)^{M-1}$ which are reflections of the current n points through the origin; this ensures that the design is symmetric.
4. Translate the $2n$ points from $(-1,1)^M$ to $(0,2)^M$, scale them to $(0,1)^M$, and then set $n = 2n$.
5. Repeat steps 2 through 4 until the desired number of points has been obtained, i.e. until $n = N$.

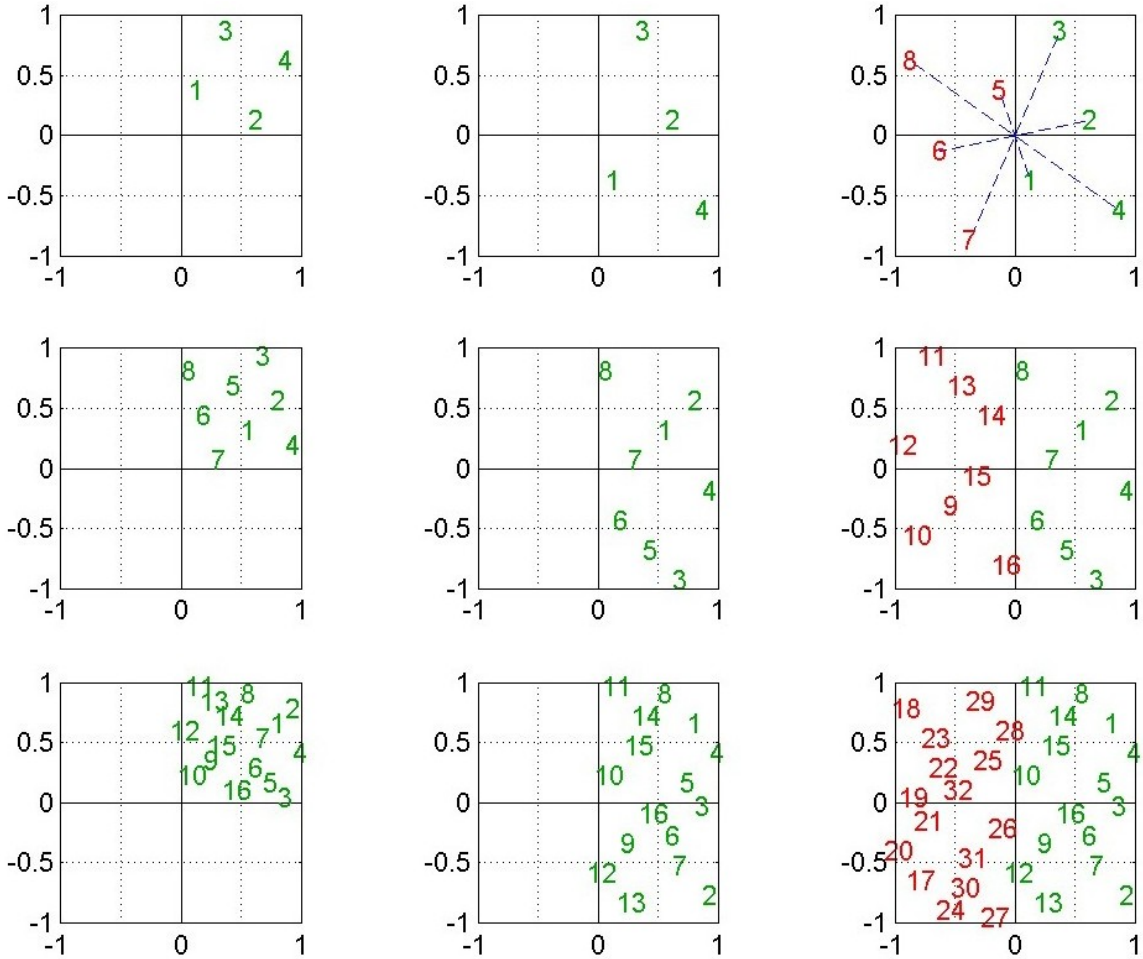

Figure 5. This figure shows the step by step progression that generates a $N = 32$ point design in $M = 2$ dimensions. The basic logic is that starting from N points well distributed in $(0,1)^M$ it is possible to obtain N points well distributed in $(0,1) \otimes (-1,1)^{M-1}$ by negating the signs of $N/2$ coordinates in each dimension other than the first. How well the resulting N points are distributed depends on which coordinates are selected for negation. After having obtained well distributed points in $(0,1) \otimes (-1,1)^{M-1}$, the other half of the $(-1,1)^M$ hypercube can be filled in by adding mirror points, i.e. reflections of the current points through the origin. The addition of the N mirror points ensures that the resulting sample design $\mathcal{X}$ will be symmetric. Adding 1 to each coordinate and then scaling by half shifts the $2N$ points back to $(0,1)^M$. This process to double the existing points can be repeated until the desired (to within a factor of 2) number of points is obtained. An initial set of $2M$ well distributed points can be easily obtained for $M = 2^p$ dimension as the end points of a set of orthogonal axes.

This process is depicted visually for $M = 2$ and $N = 32$ in Figure 5. Since a (pseudo) random number was used to choose between binning optimal alternatives for octant assignment in step 2, it will not be optimal in an absolute sense. This can be seen in the second row of plots; an optimal quadrant/octant assignment would have put points 1, 3, 6 and 8 in one octant and points 2, 4, 5 and 7 in the other.

The original $2M$ points in step 1 can be found as the end points of a set of orthogonal axes. This can be easily done if $M = 2^p$ for non negative integer p and the logic will be discussed shortly.

An efficient algorithm for step 2 can be implemented by sorting sample points according to their locality preserving bin ids and then separating sequential points to ensure "dis-locality" of the samples. To be more specific, nearby points are sent to octants that are far apart from each other. The logic used also requires a 1 time grouping of sets of axes that are maximally spaced from each other; this grouping is used to determine the order in which octants will be filled in.

A. Algorithm for Initial Orthogonal Axes LHS Design

In $M = 2^p$ dimensions, $p = 0, 1, 2, 3, ...$, a $2M$ point symmetric cell centered Latin Hypercube which constitutes a set of orthogonal axes can be generated by tiling simple patterns. The first pattern is for the magnitudes of coordinates (or positions within an array of magnitudes for coordinates). The second pattern is for the signs of those coordinate magnitudes.

"Coverage" for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	0.0717773	0.027062	0.0178996		
16	0.135135	0.104126	0.0916156	0.128427	0.0625
32	0.285717	0.233105	0.219465		
64	0.417035	0.372359	0.361626		
128	0.56022	0.522201	0.511982		
256	0.678416	0.647304	0.645049	0.667668	0.316406
512	0.773748	0.754804	0.749725		
1024	0.843177	0.832896	0.831007		
2048	0.896093	0.890245	0.886593		
4096	0.932229	0.928693	0.927748	0.929509	0.586182
8192	0.956723	0.954248	0.953466		
16384	0.97319	0.97129	0.971217		
32768	0.983415	0.982499	0.982312		
65536	0.989815	0.989387	0.98926	0.98965	0.772476

Table 3. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the "coverage" (higher is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

Cell centered coordinate magnitudes between 0 and $2M$ for the first point can be generated as

$$\begin{aligned} |X_{1,k}| &= 2k-1 \qquad\qquad \text{for } k = 1,2,3,...,M \\ \left|\underline{x_1}\right| &= [1,3,5,...,2M-1]^T \end{aligned} \tag{11}$$

The magnitudes for the second point can be generated by reversing the order within each disjoint pair of consecutive coordinates in $\left|\underline{x_1}\right|$, i.e.

$$\left|\underline{x_2}\right| = [3,1,7,5,11,9,...,2M-1,2M-3]^T \tag{12}$$

The magnitudes for $\left|\underline{x_3}\right|$ and $\left|\underline{x_4}\right|$ can be generated by reversing the order within every disjoint group of 4 consecutive coordinates in $\left|\underline{x_1}\right|$ and $\left|\underline{x_2}\right|$ respectively, i.e.

$$\left|\underline{x_3}\right| = [7,5,3,1,15,13,11,9,...,2M-1,2M-3,2M-5,2M-7]^T \tag{13}$$

$$\left|\underline{x_4}\right| = [5,7,1,3,13,15,9,11,...,2M-3,2M-1,2M-7,2M-5]^T \tag{14}$$

The magnitudes for $\left|\underline{x_{n+1}}\right|$ through $\left|\underline{x_{2n}}\right|$, $n = 2^p$, $p = 0,1,2,,...$ are generated by reversing the order within each disjoint group of n consecutive coordinates for points $\left|\underline{x_1}\right|$ through $\left|\underline{x_n}\right|$ respectively. For $M = 8$ dimensions this is

$$\left|\left[\underline{x_1},\underline{x_2},\underline{x_3},...,\underline{x_8}\right]^T\right| = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & 1 & 7 & 5 & 11 & 9 & 15 & 13 \\ 7 & 5 & 3 & 1 & 15 & 13 & 11 & 9 \\ 5 & 7 & 1 & 3 & 13 & 15 & 9 & 11 \\ 15 & 13 & 11 & 9 & 7 & 5 & 3 & 1 \\ 13 & 15 & 9 & 11 & 5 & 7 & 1 & 3 \\ 9 & 11 & 13 & 15 & 1 & 3 & 5 & 7 \\ 11 & 9 & 15 & 13 & 3 & 1 & 7 & 5 \end{bmatrix} \tag{15}$$

We use the Sylvester construction of Hadamard matrices to generate the pattern of signs,

$$\begin{aligned} &\begin{bmatrix} + \end{bmatrix} \\ &\begin{bmatrix} + & + \\ + & - \end{bmatrix} \\ &\begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix} \end{aligned} \tag{16}$$

Condition Number of the Correlation Matrix for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	1	14.6273	8.23719		
16	3.2505	4.14988	3.75258	2.39394	1
32	1.49974	2.27709	2.15406		
64	1.37672	1.76306	1.82367		
128	1.2064	1.4508	1.49656		
256	1.11022	1.32572	1.33407	1.10916	1
512	1.05589	1.21341	1.2108		
1024	1.0368	1.1546	1.14725		
2048	1.02121	1.09974	1.09939		
4096	1.01246	1.07576	1.07075	1.01254	1
8192	1.00717	1.04643	1.04922		
16384	1.00403	1.03608	1.03365		
32768	1.0027	1.02297	1.02461		
65536	1.00166	1.01872	1.01742	1.00145	1

Table 4. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the condition number of the correlation matrix (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

$$\begin{bmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{bmatrix} \tag{17}$$

Pseudo code for the logic to generate this sequence is

$$\begin{aligned} &\underline{\underline{signs}} = [+1] \\ &\text{for } i = 1 \rightarrow \log_2(M) \\ &\qquad \underline{\underline{signs}} = \begin{bmatrix} \underline{\underline{signs}} & \underline{\underline{signs}} \\ \underline{\underline{signs}} & -\underline{\underline{signs}} \end{bmatrix} \end{aligned}$$

The Hadamard product (also known as the *Schur product* and the *entrywise product*) of the signs in Eqn. (17) and the magnitudes in Eqn. (15) results in the following set of $M = 8$ orthogonal vectors

$$\left[\underline{x_1}, \underline{x_2}, \underline{x_3}, ..., \underline{x_8}\right]^T = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & -1 & 7 & -5 & 11 & -9 & 15 & -13 \\ 7 & 5 & -3 & -1 & 15 & 13 & -11 & -9 \\ 5 & -7 & -1 & 3 & 13 & -15 & -9 & 11 \\ 15 & 13 & 11 & 9 & -7 & -5 & -3 & -1 \\ 13 & -15 & 9 & -11 & -5 & 7 & -1 & 3 \\ 9 & 11 & -13 & -15 & -1 & -3 & 5 & 7 \\ 11 & -9 & -15 & 13 & -3 & 1 & 7 & -5 \end{bmatrix} \tag{18}$$

The vectors in Eqn. (18) are then reflected through zero to add a set of M mirror points and then the $2M$ points are translated and scaled to $(0, 1)^M$.

That these initial $2M$ points constitute an **orthogonal** BOSLHS design is the reason why the condition number for the design's correlation matrix is exactly one for the minimum number of points in Tables 4 and 10.

B. Algorithm For Octant Assignment

The sub-problem in step 2 of the BOSLHS algorithm outlined above is perhaps the most conceptually challenging to solve efficiently. The above statement of the problem, namely to redistribute the points from $(0, 1)^M$ to

Degree of Binning Non-Optimality for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	(0 , 1)	(0.600 , 1.675)	(0.725 , 1.900)		
16	(0 , 1)	(1.000 , 2.675)	(1.000 , 3.100)	(0 , 1)	(0 , 1)
32	(0 , 1)	(2.000 , 1.875)	(2.000 , 2.000)		
64	(0 , 1)	(2.000 , 2.175)	(2.000 , 2.450)		
128	(0 , 1)	(2.000 , 3.225)	(2.000 , 3.425)		
256	(0 , 1)	(2.000 , 4.650)	(2.000 , 4.725)	(0 , 1)	(0 , 1)
512	(0 , 1)	(3.000 , 2.825)	(3.000 , 2.600)		
1024	(0 , 1)	(3.000 , 3.475)	(3.000 , 3.550)		
2048	(0 , 1)	(3.000 , 4.550)	(3.000 , 4.425)		
4096	(0 , 1)	(3.000 , 6.175)	(3.000 , 6.300)	(0 , 1)	(0 , 1)
8192	(0 , 1)	(4.000 , 3.600)	(4.000 , 3.475)		
16384	(0 , 1)	(4.000 , 4.450)	(4.000 , 4.475)		
32768	(0 , 1)	(4.000 , 5.625)	(4.000 , 5.675)		
65536	(0 , 1)	(4.000 , 7.625)	(4.000 , 7.575)	(0 , 1)	(0 , 1)

Table 5. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the degree of binning non-optimality (lower is better, (0 , 1) is binning optimal) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

$(0,1) \otimes (-1,1)^{M-1}$ by negating $\frac{N}{2}$ coordinates from each dimension other than the first, suggests a dimension by dimension algorithm that is likely difficult to implement.

However, reformulating the problem as the assignment of points to 2^{M-1} octants in $(0,1) \otimes (-1,1)^{M-1}$ greatly facilitates its efficient solution. Points which are nearby each other should be sent to octants that are far apart from each other. Quick-sorting according to SFC index bin ids, as presented in Section I. reorders points so that consecutively listed points are known to be nearby each other. The remaining task is to generate a listing of receiving octants such that consecutive octants are maximally spaced from each other. Here, "maximally spaced" is equivalent to the octants' bit-signs representations being separated by the largest possible Hamming distance that is less than or equal to $M/2$ (if more than $M/2$ bits are of different signs, then the signs of one octant are negated for the purposes of calculating the Hamming distance).

The $2M$ end points of a M-dimensional orthogonal set of axes are maximally spaced, and therefore the octants containing those endpoints will also be maximally spaced. Since the problem is to distribute the points to only the 2^{M-1} octants in $(0,1) \otimes (-1,1)^{M-1}$, the algorithm only needs to deal with one endpoint from each of the M axes in an orthogonal set. We will hereafter use the word *orientation* to refer to a set of M mutually orthogonal axes in M-dimensional space and also to the set of octants containing the endpoints of those axes. Therefore, it is reasonable and beneficial for the algorithm to distribute a group of M nearest (within a bin) points so that each of the M endpoint octants for an orientation receives 1 point.

Given $2M$ points in a bin, then we should distribute those points to 2 orientations that are maximally spaced from each other. This is equivalent to rotating or reflecting a copy of the original orientation to be maximally spaced from the original. Likewise, if there are $4M$ points in a bin, they should be distributed to 4 orientations that are mutually maximally spaced from each other. If there are $8M$ points in a bin, they should be distributed to 8 orientations that are mutually maximally spaced from each other.

However, the case where a bin contains 2^M points has an additional requirement. Since each of the 2^{M-1} octants in $(0,1) \otimes (-1,1)^{M-1}$ will receive 2 points, we require for points in opposite sub-bins to be sent to the same octant. Using Z-curve index bin ids makes this straightforward. Let Z_{b_1} and Z_{b_2} be the Z-curve ids of sub-bins within the bin with id Z_b, then $Z_{b_1} = Z_b + z_1$ and $Z_{b_2} = Z_b + z_2$ are opposite sub-bins iff $z_1 + z_2 = 2^M - 1$.

Another requirement of the algorithm is that every octant in $(0,1) \otimes (-1,1)^{M-1}$ must receive 1 point before any octant receives 2. In other words, the algorithm must not start over from the top of the list of destination octants just because the next set of points to assign originate from a different bin that the previous set.

For the sake of simplicity and also to avoid regularity, our algorithm randomly permutes the order within sets and successively smaller subsets of orientations such that maximal spacing of sequentially listed orientations is retained.

(t, m, s)-net Rating for $M = 4$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
8	(1 , 3 , 4)	(2 , 3 , 4)	(3 , 3 , 4)		
16	(2 , 4 , 4)	(3 , 4 , 4)	(4 , 4 , 4)	(3 , 4 , 4)	(3 , 4 , 4)
32	(2 , 5 , 4)	(4 , 5 , 4)	(5 , 5 , 4)		
64	(3 , 6 , 4)	(5 , 6 , 4)	(6 , 6 , 4)		
128	(4 , 7 , 4)	(6 , 7 , 4)	(7 , 7 , 4)		
256	(5 , 8 , 4)	(7 , 8 , 4)	(8 , 8 , 4)	(6 , 8 , 4)	(6 , 8 , 4)
512	(5 , 9 , 4)	(8 , 9 , 4)	(9 , 9 , 4)		
1024	(6 , 10 , 4)	(9 , 10 , 4)	(10 , 10 , 4)		
2048	(7 , 11 , 4)	(10 , 11 , 4)	(11 , 11 , 4)		
4096	(8 , 12 , 4)	(11 , 12 , 4)	(12 , 12 , 4)	(9 , 12 , 4)	(9 , 12 , 4)
8192	(8 , 13 , 4)	(12 , 13 , 4)	(13 , 13 , 4)		
16384	(9 , 14 , 4)	(13 , 14 , 4)	(14 , 14 , 4)		
32768	(10 , 15 , 4)	(14 , 15 , 4)	(15 , 15 , 4)		
65536	(11 , 16 , 4)	(15 , 16 , 4)	(16 , 16 , 4)	(12 , 16 , 4)	(12 , 16 , 4)

Table 6. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the (t, m, s)-net rating (lower t is better, $N = b^m$ where $b = 2$, $M = s$) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

Within each orientation, the order in which axes endpoint octants receives points is also randomly permuted. Once the random ordering of 2^{M-1} octant-point pairs has been generated, the octant order is "reversed"[‖] for the next 2^{M-1} points to ensure that points in opposite sub-bins will be sent to the same octant. An independent random ordering is generated in this fashion for every 2^M points.

C. Generating a List of Maximally Spaced Orientations

As just mentioned, there is a need to generate a listing of orientations that are maximally spaced from each other. Note that there are $2M$ end-points/octants per orientation (considering both endpoints of each axis) and a total of 2^M octants in the hypercube. This means there are $\frac{2^M}{2M} = 2^{M-1}/M$ disjoint orientations. The solution to this problem for 8 or fewer dimensions is straightforward:

In $M = 2$ dimensions there is only 1 orientation so the choice of which orientation to assign the next set of $2M = 4$ nearest points is trivial.

In $M = 4$ dimensions there are only 2 orientations. These can be represented by the sign of the first point in Eqn. (16). The first orientation is obtained multiplying each row of Eqn. (16) by $\begin{bmatrix} + & + & + & + \end{bmatrix}$. The second orientation is obtained by negating the Hadamard product of each row of Eqn. (16) and $\begin{bmatrix} - & + & + & + \end{bmatrix}$.

In $M = 8$ dimensions there are 16 orientations. The first 8 orientations can be represented by the signs of their first point.

$$\begin{matrix} + & + & + & + & + & + & + & + \\ - & - & + & + & + & + & + & + \\ - & + & - & + & + & + & + & + \\ - & + & + & - & + & + & + & + \\ - & + & + & + & - & + & + & + \\ - & + & + & + & + & - & + & + \\ - & + & + & + & + & + & - & + \\ - & + & + & + & + & + & + & - \end{matrix}$$

Note that these rows are mutually separated by 2 $\left(\text{i.e } \frac{M}{4}\right)$ of their bit-signs. Any row with a negative in the first dimension can be negated to ensure that points will only be moved to other octants within $(0, 1) \otimes (-1, 1)^{M-1}$ (although, due to symmetry, this step isn't strictly required). As with $M = 4$ dimensions, the rest of the signs for each orientation can be obtained through a row by row Hadamard product with Eqn. (17).

‖A simple reversal is sufficient for Z-curve but not Hilbert curve index ids. The corresponding "reverse" ordering for Hilbert curves is significantly more complicated.

Centered L_2-Discrepancy for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	0.284038	0.34294	0.506724		
32	0.210856	0.238596	0.358231		
64	0.142917	0.167027	0.254876		
128	0.0983407	0.115722	0.180471		
256	0.0669426	0.0826593	0.125864	0.0991629	0.60562
512	0.0430124	0.0588062	0.0882444		
1024	0.0294717	0.0408708	0.062649		
2048	0.0199896	0.0289271	0.0448694		
4096	0.0135911	0.0206844	0.0312016		
8192	0.00929004	0.0146224	0.0225828		
16384	0.00631475	0.010354	0.0156934		
32768	0.00423738	0.00724978	0.011234		
65536	0.00287464	0.00507716	0.00778369	0.00465929	0.27813

Table 7. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the Centered L_2-Discrepancy (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

Note that for these first 8 orientations, the row by row Hadamard product with Eqn. (18) is a rotation matrix. The second set of 8 orientations can likewise be represented as

```
- + + + + + + +
+ - + + + + + +
+ + - + + + + +
+ + + - + + + +
+ + + + - + + +
+ + + + + - + +
+ + + + + + - +
+ + + + + + + -
```

As before, rows with a negative sign in the first dimension (in this case only the first row) would be negated. Note that the row by row Hadamard product of these sets of signs and (a normalized version of) Eqn. (18) is a direction cosines matrix rather than a true rotation matrix (these differ from a rotation matrix by the sign of one column, in other words it represents a reflection of a rotation).

The pattern to use for $M = 16$ and higher dimensions is significantly more complicated. However, the solution of this problem has been obtained using group theory and is the subject of a separate forth-coming paper. The MATLAB implementation of our algorithm to generate the maximally spaced list of orientations for $M = 16$ dimensions takes significantly less than 1 minute on a 2.53 GHz Intel quad-core processor. This solution needs only be computed once and can be saved to a file to be reloaded on subsequent calls of the BOSLHS algorithm.

V. Results

Tables 1 through 12 demonstrate that, BOSLHS designs are typically superior to regular Latin Hypercube, Monte Carlo, Jittered, and Tensor Product sampling in the metrics of centered and wrap-around L_2 discrepancy, coverage, orthogonality (measured by condition number of the correlation matrix), the degree of binning non-optimality, and the t quality metric when the designs are considered to be (t, m, s) nets. The exceptions are

- TPS and JS are equivalent in the metric of binning non-optimality,
- TPS designs are strictly orthogonal,
- JS tend to be slightly more orthogonal (evidenced by the condition number of the correlation matrix) for very low numbers of points and are comparable for larger numbers of points,
- regular LHS had a slightly lower wrap-around L_2 discrepancy for the minimum possible number of points $N = 2M$ in $M = 8$ dimensions.

Wrap Around L_2-Discrepancy for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	0.732593	0.641005	0.987825		
32	0.388246	0.438441	0.702694		
64	0.269391	0.303077	0.502062		
128	0.184903	0.211267	0.356757		
256	0.128041	0.149853	0.249652	0.220813	1.66975
512	0.0889111	0.105794	0.175812		
1024	0.0586689	0.0746833	0.123312		
2048	0.0403764	0.0523196	0.088177		
4096	0.0281528	0.0373708	0.0613596		
8192	0.0191316	0.0263039	0.0433192		
16384	0.0132848	0.0185744	0.0309117		
32768	0.00895188	0.0131266	0.0218121		
65536	0.00622988	0.00928822	0.0152641	0.0113352	0.801021

Table 8. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the Wrap Around L_2-Discrepancy (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

"Coverage" for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	0.00123468	0.000132215	0.000103482		
32	0.00343898	0.00212481	0.00193827		
64	0.0143622	0.0112283	0.0105574		
128	0.041456	0.0361297	0.0334309		
256	0.089389	0.0806103	0.0774927	0.0827604	0.00390625

Table 9. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the "coverage" of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

In order to illustrate that BOSLHS designs are good in the "eyeball metric" Figures 6, 7, and 8 plot the projections of randomly generated BOSLHS designs onto each pairing of 2 out of $M = 4$ dimensions for $N = 128$, 1024 and 4096 points. These figures demonstrate that the randomly generated point sets are visibly well distributed and space-filling.

Figures 2, 3 and 4 show that for $M = \{2, 4, 8\}$ dimensions respectively, our relatively simple algorithm achieves both a lower initial discrepancy and a faster decay of discrepancy than both JS and randomly paired cell centered LHS. In this sense, its performance is greater than that of the sum of its parts. Unlike the Multi-Jittered Sampling of Chiu, Wang and Shirley[8] our BOSLHS is restricted neither to $M = 2$ dimensions nor to a number of points, N, that is exponential in the number of random dimensions. Instead, N can be any power of 2 greater than or equal to $2M$. Conceptually, our algorithm can generate BOSLHS designs for $M = 2^q$ dimensions where q is any non-negative integer. However, the memory requirements of simply listing all $\frac{2^M}{2M}$ orientations is prohibitive for $M \geq 32$. Generating a maximally spaced subset of orientations for large numbers of dimensions is a topic of our current research.

High quality designs for non power of 2 values of M can also be obtained by discarding excess dimensions from a design with the next larger power of 2 number of dimensions. There are $\mathrm{nchoosek}\left(2\hat{}\mathrm{ceil}\left(\log_2\left(M\right)\right), M\right)$ combinations of dimensions to retain and for large N using the discrepancy as a criteria to choose between them would take an exceedingly long time. Fortunately, most of the combinations of dimensions for which the discrepancy needs to be evaluated can be eliminated by using the degree of binning non-optimality, (g, s), as a screening criteria, and then only evaluating the discrepancy for the resulting designs which are tied for the lowest degree of binning non-optimality.

In terms of computational cost, the MATLAB implementation of our algorithm took approximately 0.78 seconds and $\mathcal{O}(N \log(N))$ operations to generate a $M = 4$ dimensional design with $N = 65,536$ points on one core of a 2.53 GHz Intel quad core processor. By contrast it took 137.2 seconds and $\mathcal{O}(MN^2)$ operations to calculate the centered L_2 discrepancy for the same design. Generating an $M = 8$ dimensional design with $N = 65,536$ points took

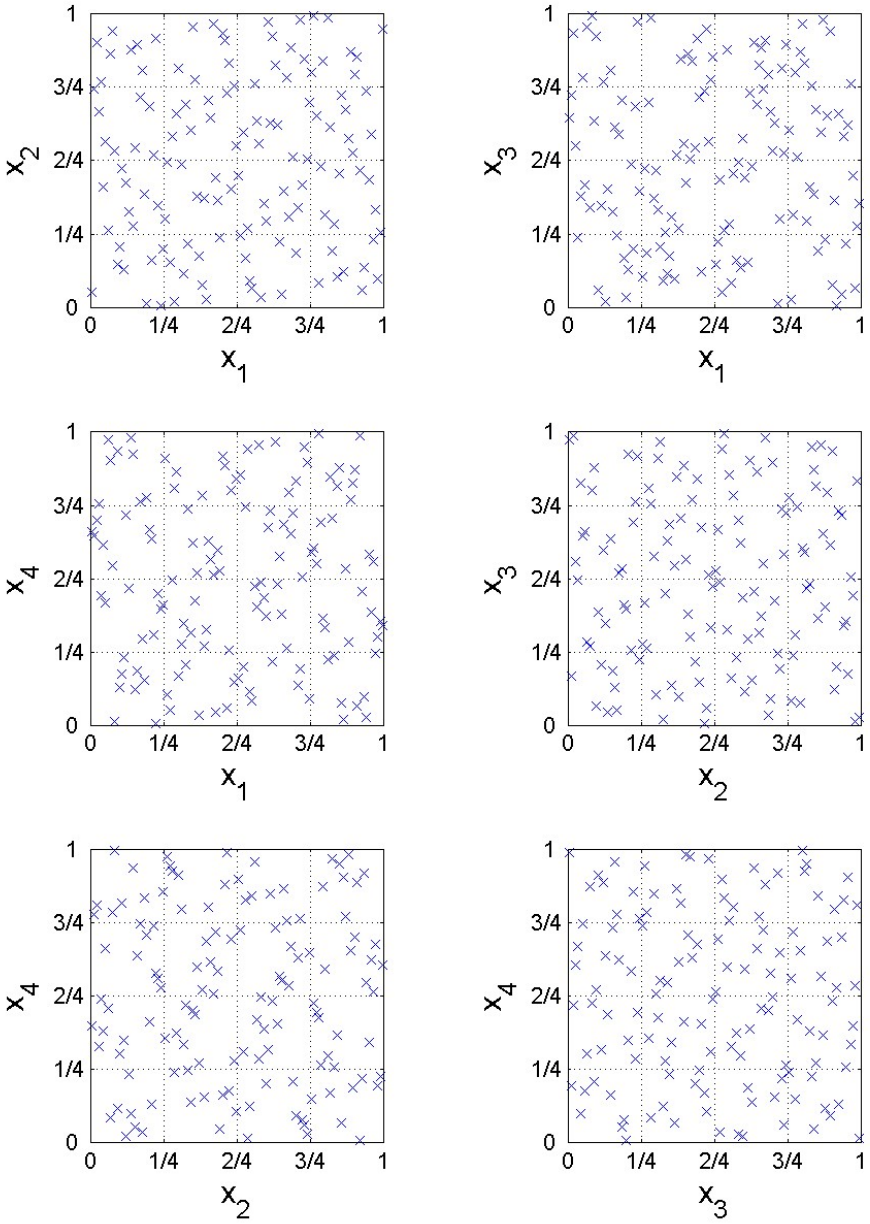

Figure 6. Plots of the projection of a random $M = 4$ dimensional $N = 128$ point binning optimal symmetric Latin hypercube sample design onto each pair of dimensions. When the hypercube is divided into a tensor product of disjoint congruent bins with 2 bins per edge, each of those bins contains 8 points. When divided into a tensor product of bins with 4 bins per edge, each of those bins contains either 0 or 1 point.

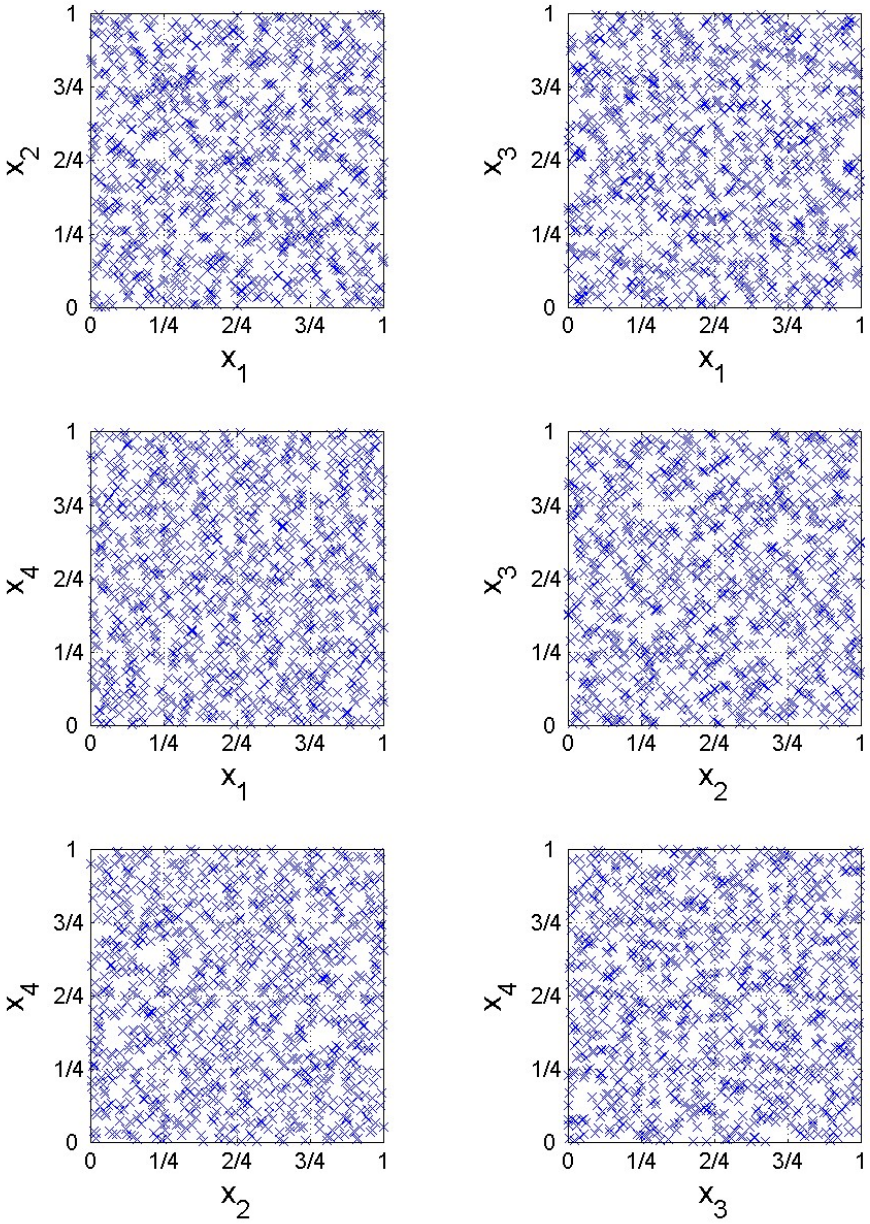

Figure 7. Plots of the projection of a random $M = 4$ dimensional $N = 1024$ point binning optimal symmetric Latin hypercube sample design onto each pair of dimensions. When the hypercube is divided into a tensor product of disjoint congruent bins with 4 bins per edge, each of those bins contains 4 points. When divided into a tensor product of bins with 8 bins per edge, each of those bins contains either 0 or 1 point.

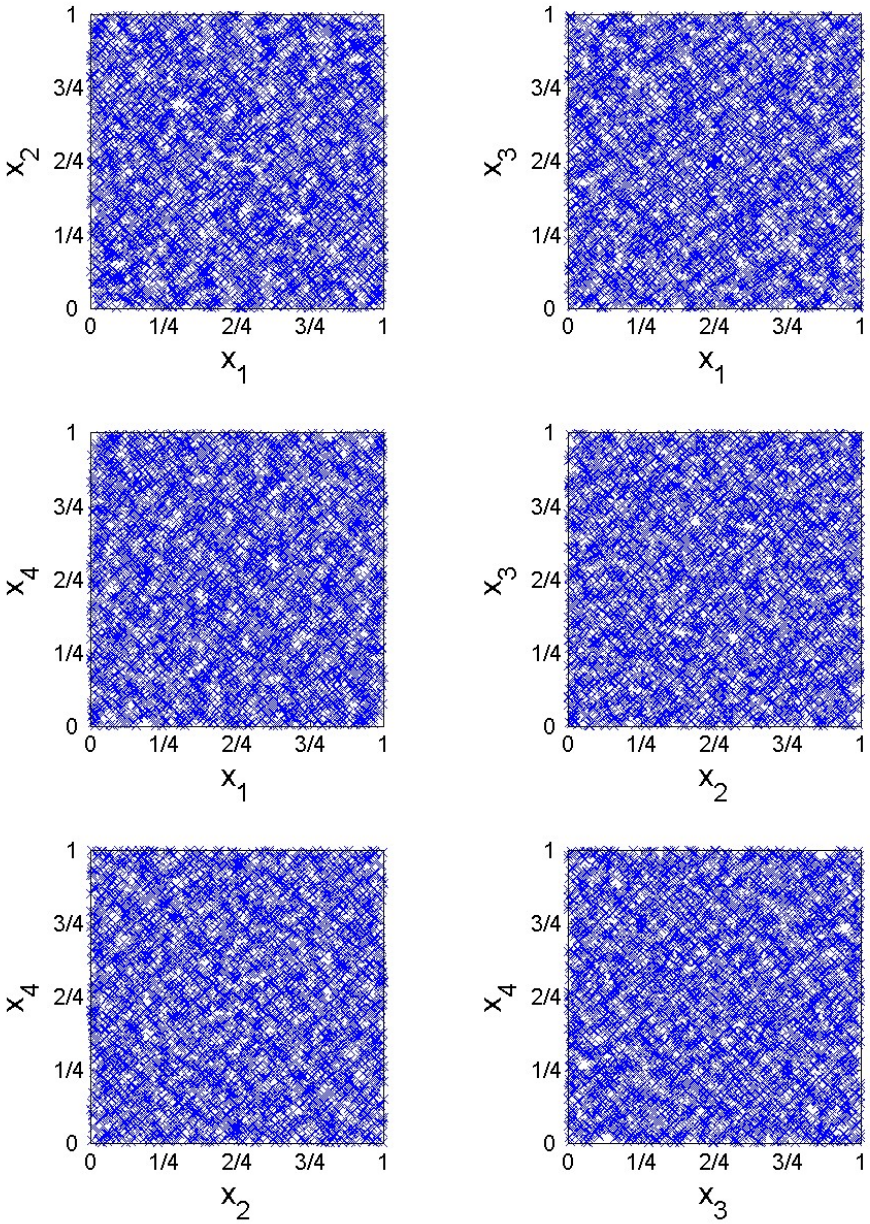

Figure 8. Plots of the projection of a random $M = 4$ dimensional $N = 4096$ point binning optimal symmetric Latin hypercube sample design onto each pair of dimensions. When the hypercube is divided into a tensor product of disjoint congruent bins with 8 bins per edge, each of those bins contains 1 point.

Condition Number of the Correlation Matrix for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	1	18.3821	15.234		
32	4.90902	5.39655	5.28336		
64	2.70441	2.96554	2.82453		
128	1.93065	2.10261	2.09155		
256	1.52916	1.70239	1.67543	1.43407	1
512	1.17397	1.45548	1.44352		
1024	1.15153	1.28963	1.30738		
2048	1.10545	1.20011	1.19761		
4096	1.05825	1.13816	1.13835		
8192	1.04344	1.09823	1.09832		
16384	1.02504	1.06745	1.07064		
32768	1.01813	1.04631	1.04637		
65536	1.00938	1.03061	1.03375	1.01092	1

Table 10. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the condition number of the correlation matrix (lower is better) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

approximately 0.32 seconds (not an error, the time for $M = 8$ dimensions was less than half of that for $M = 4$ dimensions, we attribute this to better cache use); calculating that design's discrepancy took 295.9 seconds. For the sake of comparison, other algorithms to generate space-filling LHS designs are reported in the literature to take "minutes" to generate a $N = 100$ point design in $M = 10$ dimensions.

Algorithms that more tightly constrain the randomness can produce BOSLHS designs with lower discrepancy. However, the limited constraints on the random assignment of points to octants we employed facilitates direct comparison of BOSLHS with JS and randomly paired cell-centered LHS. Our simple random algorithm should also be viewed as a baseline against which the larger class of BOSLHS designs should be compared. Two potential ways (both are topics of our current research) that BOSLHS designs could obtain lower discrepancy are by

1. making low dimensional projections binning optimal, and

2. a better selection of orientations of adjacent bins.

VI. Summary

The universal applicability of Monte Carlo Sampling (MCS) as a method of uncertainty quantification is limited by the slow rate of convergence in the error of its sample mean. The "Koksma-Hlawka-like inequality" bounds this error in terms of the sample design's discrepancy, which is a common metric of uniformity and therefore the degree to which a design is space-filling. However, even the "fast" formulas available for certain useful L_2 norm discrepancies require $\mathcal{O}\left(N^2 M\right)$ operations, where M is the number of dimensions and N is the number of points in the design. Latin Hypercube Sampling (LHS) and Jittered Sampling (JS) both achieve better error convergence than standard MCS by using stratification to obtain a more uniform selection of samples, although LHS and JS use different stratification strategies. JS is space-filling in the full M dimensional space but not in the 1 dimensional projections, while LHS is space-filling in the 1 dimensional projections but not in the full M dimensional space.

In this paper, we defined "binning optimality," a new metric of the space-filling property which can be evaluated in $\mathcal{O}\left(N \log\left(N\right)\right)$ operations, and presented an $\mathcal{O}\left(N \log\left(N\right)\right)$ fast Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS) algorithm. BOSLHS combines the best features of JS and LHS to produce designs that are space-filling in the full M dimensional space **and** the 1 dimensional projections. BOSLHS designs are also symmetric with respect to reflection through their center, which reduces correlation between dimensions. Our experimental results show that BOSLHS designs are superior to conventional JS and LHS in several quality metrics. Compared to other algorithms which are reported in the literature to require "minutes" to generate a $M = 10$ dimensional space-filling LHS designs with $N = 100$ points, our BOSLHS algorithm can generate $N = 2^{16} = 65536$ point $M = 8$ dimensional designs in significantly under a second.

Degree of Binning Non-Optimality for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	(0 , 1)	(0.200 , 1.200)	(0.450 , 1.450)		
32	(0 , 1)	(0.775 , 1.800)	(0.975 , 2.025)		
64	(0 , 1)	(1.000 , 2.275)	(1.000 , 2.350)		
128	(0 , 1)	(1.000 , 3.275)	(1.000 , 3.250)		
256	(0 , 1)	(1.000 , 4.450)	(1.000 , 4.800)	(0 , 1)	(0 , 1)
512	(0 , 1)	(2.000 , 1.925)	(2.000 , 1.850)		
1024	(0 , 1)	(2.000 , 2.050)	(2.000 , 2.050)		
2048	(0 , 1)	(2.000 , 2.200)	(2.000 , 2.325)		
4096	(0 , 1)	(2.000 , 2.975)	(2.000 , 3.000)		
8192	(0 , 1)	(2.000 , 3.325)	(2.000 , 3.525)		
16384	(0 , 1)	(2.000 , 4.225)	(2.000 , 4.525)		
32768	(0 , 1)	(2.000 , 5.650)	(2.000 , 5.675)		
65536	(0 , 1)	(2.000 , 7.375)	(2.000 , 7.600)	(0 , 1)	(0 , 1)

Table 11. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the degree of binning non-optimality (lower is better, (0 , 1) is binning optimal) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

Acknowledgments

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Much of the work in this paper was performed during Dalbey's doctoral studies at the University at Buffalo. He acknowledges the role of his adviser, Professor Abani K. Patra, who enabled him to successfully conduct this research. Professor Patra arranged for financial support, instructed Dalbey on many topics, including space-filling curves, and provided the application to motivate this research.

References

[1]Hickernell, F., "A generalized discrepancy and quadrature error bound," *Mathematics of Computation*, Vol. 67, No. 221, 1998, pp. 299–322.

[2]Fang, K.-T., Li, R., and Sudjianto, A., *Design and Modeling for Computer Experiments*, Chapman & Hall/CRC, London, UK, 2006.

[3]Wu, C., Hamada, M., and Wu, C., *Experiments: Planning, Analysis, and Parameter Design Optimization*, Wiley New York, 2000.

[4]Hickernell, F., "Lattice rules: how well do they measure up," *Random and Quasi-Random Point Sets*, edited by P. Hellekalek and G. Larcher, Springer, 1998, pp. 109–166.

[5]Niederreiter, H., *Random Number Generation and Quasi-Monte Carlo Methods*, Vol. 63 of *CBMS-NSF Regional Conference Series In Applied Mathematics*, Society for Industrial Mathematics, Philadelphia, Pennsylvania, 1992.

[6]Hemez, F., Atamturktur, H., and Unal, C., "Defining predictive maturity for validated numerical simulations," *Computers & Structures*, 2010.

[7]Cioppa, T. and Lucas, T., "Efficient nearly orthogonal and space-filling latin hypercubes," *Technometrics*, Vol. 49, No. 1, 2007, pp. 45–55.

[8]Chiu, K., Shirley, P., and Wang, C., "Multi-jittered sampling," *Academic Press Graphics Gems Series*, 1994, pp. 370–374.

[9]Park, J., "Optimal latin-hypercube designs for computer experiments," *Journal of Statistical Planning and Inference*, Vol. 39, No. 1, 1994, pp. 95–111.

[10]Morris, M. and Mitchell, T., "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, Vol. 43, No. 3, 1995, pp. 381–402.

[11]Ye, K., Li, W., and Sudjianto, A., "Algorithmic construction of optimal symmetric latin hypercube designs," *Journal of Statistical Planning and Inference*, Vol. 90, No. 1, 2000, pp. 145–159.

[12]Cioppa, T. M., *Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models*, Ph.D. thesis, Naval Postgraduate School, September 2002.

[13]Bates, S., Sienz, J., and Langley, D., "Formulation of the audze-eglais uniform latin hypercube design of experiments," *Advances in Engineering Software*, Vol. 34, No. 8, 2003, pp. 493–506.

[14]Bates, S., Sienz, J., and Toropov, V., "Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm," *AIAA 2004*, Vol. 2011, 2004, pp. 1–7.

[15]Husslage, B., van Dam, E., and Hertog, D., "Nested maximin latin hypercube designs in two dimensions," *Discussion Paper*, 2005.

[16]Husslage, B., Rennen, G., van Dam, E., and Hertog, D., "Space-filling latin hypercube designs for computer experiments," *Discussion Paper*, 2006.

[17]van Dam, E., Husslage, B., Hertog, D., and Melissen, H., "Maximin latin hypercube designs in two dimensions," *Operations Research*, Vol. 55, No. 1, 2007, pp. 158.

[18]Joseph, V. and Hung, Y., "Orthogonal-maximin latin hypercube designs," *Statistica Sinica*, Vol. 18, No. 1, 2008, pp. 171.

[19]van Dam, E., Rennen, G., and Husslage, B., "Bounds for maximin latin hypercube designs," *Operations Research*, Vol. 57, No. 3, 2009, pp. 595–608.

(t, m, s)-net Rating for $M = 8$ Dimensions: Average of 40 runs					
N	Binning Optimal Symmetric LHS	Cell Centered Random LHS	Monte Carlo Sampling	Jittered Sampling	Tensor Product Sampling
16	(2 , 4 , 8)	(3 , 4 , 8)	(4 , 4 , 8)		
32	(3 , 5 , 8)	(4 , 5 , 8)	(5 , 5 , 8)		
64	(4 , 6 , 8)	(5 , 6 , 8)	(6 , 6 , 8)		
128	(5 , 7 , 8)	(6 , 7 , 8)	(7 , 7 , 8)		
256	(6 , 8 , 8)	(7 , 8 , 8)	(8 , 8 , 8)	(7, 8, 8)	(7, 8, 8)
512	(6 , 9 , 8)	(8 , 9 , 8)	(9 , 9 , 8)		
1024	(7 , 10 , 8)	(9 , 10 , 8)	(10 , 10 , 8)		
2048	(8 , 11 , 8)	(10 , 11 , 8)	(11 , 11 , 8)		
4096	(9 , 12 , 8)	(11 , 12 , 8)	(12 , 12 , 8)		
8192	(10 , 13 , 8)	(12 , 13 , 8)	(13 , 13 , 8)		
16384	(11 , 14 , 8)	(13 , 14 , 8)	(14 , 14 , 8)		
32768	(12 , 15 , 8)	(14 , 15 , 8)	(15 , 15 , 8)		
65536	(13 , 16 , 8)	(15 , 16 , 8)	(16 , 16 , 8)	(14, 16, 8)	(14, 16, 8)

Table 12. The average, over 40 trials (except for Tensor Product Sampling which is completely deterministic), of the (t, m, s)-net rating (lower t is better, $N = b^m$ where $b = 2$, $M = s$) of designs generated by Binning Optimal Symmetric LHS, Cell Centered Random LHS, Monte Carlo Sampling, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

[20]Grosso, A., Jamali, A., and Locatelli, M., "Finding maximin latin hypercube designs by iterated local search heuristics," *European Journal of Operational Research*, Vol. 197, No. 2, 2009, pp. 541–547.

[21]Lin, C. D., Bingham, D., Sitter, R., and Tang, B., "Constructing cascading latin hypercubes," downloaded from http://www.stat.sfu.ca/~dbingham/NICDS_CompExpt/postercclh.pdf,Date accessed: February 10, 2009.

[22]Moore, D., "Fast hilbert curve generation, sorting, and range queries," website: http://web.archive.org/web/20041028171141/www.caam.rice.edu/~dougm/twiddle/Hilbert/, Date accessed: May 25, 2010.

[23]Fang, K. and Ma, C., "Wrap-around L2-discrepancy of random sampling, latin hypercube and uniform designs," *Journal of Complexity*, Vol. 17, No. 4, 2001, pp. 608–624.

[24]McKay, M., Beckman, R., and Conover, W., "Comparison the three methods for selecting values of input variable in the analysis of output from a computer code," *Technometrics*, Vol. 21, No. 2, May 1979, pp. 239–245.

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

[26]Owen, A., "A central limit theorem for latin hypercube sampling," *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 54, No. 2, 1992, pp. 541–551.

[27]Iman, R. and Conover, W., "A distribution-free approach to inducing rank correlation among input variables," *Communications in Statistics-Simulation and Computation*, Vol. 11, No. 3, 1982, pp. 311–334.

[28]Iman, R. and Shortencarier, M., "LHS: a program to generate input samples for multivariate simulations," *American Statistician*, Vol. 39, No. 3, 1985, pp. 212–212.

[29]Wyss, G. and Jorgensen, K., "A user's guide to lhs: sandia's latin hypercube sampling software," *SAND98–0210. Sandia (New Mexico): Sandia National Laboratory*, 1998.

[30]Owen, A., "Controlling correlations in latin hypercube samples," *Journal of the American Statistical Association*, Vol. 89, No. 428, 1994, pp. 1517–1522.

[31]Tang, B., "Orthogonal array-based latin hypercubes," *Journal of the American Statistical Association*, Vol. 88, No. 424, 1993, pp. 1392–1397.

[32]Loeppky, J., Moore, L., and Williams, B., "Batch sequential designs for computer experiments," *Journal of Statistical Planning and Inference*, 2009.

[33]Jin, R., Chen, W., and Sudjianto, A., "An efficient algorithm for constructing optimal design of computer experiments," *Journal of Statistical Planning and Inference*, Vol. 134, No. 1, 2005, pp. 268–287.