TECHNICAL UNIVERSITY OF CRETE

DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING

Selection of Orientation Subsets for High-Dimensional Space-filling Latin Hypercube Sampling

Submitted in partial fulfillment of the requirements for the diploma degree in Electronic and Computer Engineering.

Author

Fountzoulas Ioannis

Thesis Committee

Assist. Prof. George Karystinos (advisor)
Assist. Prof. Aggelos Bletsas
Assist. Prof. Dimitrios V. Rovas

Chania, August 2012

I dedicate my thesis to my beloved parents, Maria and Konstantinos.

Acknowledgements

I would like to thank my advisor Prof. George Karystinos for the topic he trusted me with, for the unlimited support and for the time he devoted me. I would also like to thank all of my friends for standing by my side all of this time.

Contents

1 Introduction

This manuscript is focused on BOSLHS designs which achieve low discrepancy without being regular. Discrepancy is a measure of uniformity (in this literature we limit ourselves to L_2 discrepancy and more specifically to centered L_2 discrepancy) appearing in Koksma-Hlawka inequality which gives an upper bound for the approximation error. Although regular designs are not desirable since lead to biased results, they achieve low discrepancy. BOSLHS technique combines most of the most favorable features of Latin Hypercube Sampling (LHS) and Jittered sampling. Latin Hypercube Sampling designs are extremely useful in the area of computer simulation. Desirably, a sample design is to be space-filling, non regular, non-collapsing and have well-spaced low-dimensional projections. When no details on the functional behavior of the response parameters are available, it is important to be able to obtain information from the entire design space. Therefore, design points should be "evenly spread" over the entire region. To obtain non-collapsing designs, the Latin Hypercube structure is often enforced but LHS are not always space-filling. Due to this fact, BOSLHS designs are proposed which are LHS (i.e., are non-collapsing) and, in addition, they satisfy the binning optimality (i.e., are space-filling). Binning optimality is proposed in [6] as a measure of space-filling and ensures that, in the limit of an infinite number of samples, the sample design will include every point in the input space. Moreover, a fast way to generate these designs through a random sample points assignment into a pre-selected set of well-spaced bins for $M \leq 16$ is described. Producing these bins is equivalent to finding a list of maximally spaced orientations (rotated sets of orthogonal axis). For $M \geq 32$, memory requirements preclude even listing the $2^M/2M$ orientation leaders and for this purpose an alternative approach is proposed. BOSLHS designs are compared with various techniques of sampling and is shown that they achieve lower mean centered L_2 discrepancy ($\mathcal{CL}_2$).

2 Koksma-Hlawka Inequality

The basic problem considered by numerical integration is to compute an approximate solution to a define integral

$$\mathcal{I}(f) = \int_{\mathcal{C}^M} f(\mathbf{x})\, d\mathbf{x}$$

as the average of the function evaluated at a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$. The Koksma-Hlawka inequality gives a tight upper bound of error on the approximation of an integral by the sample average of integrand values. Koksma-Hlawka inequality has the following form

$$|\mathcal{I}(f) - \hat{\mathcal{I}}(f, \mathcal{P})| \leq D_p(\mathcal{P})\mathcal{V}_q(f), \tag{1}$$

where $p^{-1} + q^{-1} = 1$, $\mathcal{P}$ is defined as a set of N points on $C^{\mathrm{M}} = [0,1)^{\mathrm{M}}$, $\mathrm{D}_p(\mathcal{P})$ is the p-th norm of discrepancy of $\mathcal{P}$ over the domain C^{M} that will be defined in (2) and $\mathcal{V}_q(f)$ is the q-th norm of the variation of f [2].

Definition 1 *Discrepancy is a quantitative measure of the uniformity of the distribution of points in M dimensional space and it has the following form*

$$D_p(\mathcal{P}) = \left[\sum_{u \neq \emptyset} \int_{C^u} \left| \frac{|\mathcal{P}_u \cap [0, \mathbf{x}_u)|}{N} - Vol([0, \mathbf{x}_u)) \right|^p d\mathbf{x}_u \right]^{1/p}, \tag{2}$$

where u is an index running over some or all of the subsets of $\mathcal{M} = \{1, \ldots, M\}$, $\mathcal{P}_u$ denotes the projection of the sample $\mathcal{P}$ into the cube C^u, $\mathbf{x}_u$ denotes the vector containing the components of $\mathbf{x}$ whose indices are in u, and Vol(A) denotes the volume of A.

2.1 Centered Discrepancy

While the bound of Koksma-Hlawka inequality holds for all L_p-discrepancies, the choice of which discrepancy to use to measure uniformity is an important one. Hickernell [3] pointed out some weaknesses of the L_p-discrepancies and proposed several modified L_p-discrepancies, among which the centered L_2-discrepancy ($\mathcal{CL}_2$) seems most interesting. In this manuscript we focus on $\mathcal{CL}_2$, the definition of which is the following.

$$\mathcal{CL}_2^2 = \sum_{u \neq \emptyset} \int_{C^u} \left| \frac{|\mathcal{P}_u \cap J_{\mathbf{x}_u}|}{N} - \mathrm{Vol}\left(J_{\mathbf{x}_u}\right) \right|^2 d\mathbf{x}_u,$$

where $\mathcal{P}_u$ and Vol have the same definition as above and $J_{\mathbf{x}_u}$ is the projection of $J_{\mathbf{x}}$ on C^u.

Definition 2 *$J_{\mathbf{x}}$ is an M-dimensional interval uniquely determined by $\mathbf{x}$. Let α denote the unique vertex of C^M which is closest to $\mathbf{x}$. We define*

$$J_{\mathbf{x}} = \{y \in C^M | min(\alpha_j, \mathbf{x}_j) \leq y_j < max(\alpha_j, \mathbf{x}_j), for\ 1 \leq j \leq M\}.$$

For $\mathcal{CL}_2$, Hickernell [3] derived the analytical expression

$$\begin{aligned}\mathcal{CL}_2^2(\mathcal{P}) =& \left(\frac{13}{12}\right)^M - \frac{2}{N} \sum_{k=1}^{N} \prod_{j=1}^{M} \left(1 + \frac{1}{2}|x_{kj} - 0.5| - \frac{1}{2}|x_{kj} - 0.5|^2\right) \\ &+ \frac{1}{N^2} \sum_{k=1}^{N} \sum_{j=1}^{N} \prod_{i=1}^{M} \left[1 + \frac{1}{2}|x_{ki} - 0.5| + \frac{1}{2}|x_{ji} - 0.5| - \frac{1}{2}|x_{ki} - x_{ji}|\right].\end{aligned}$$

3 Z-Order

In BOSLHS designs we have to know in which bin every point lies. This is a way to ensure that the design remains binning optimal in every step. Moreover, knowing the bin in which a point lies, enables us to distinguish which points are nearby. In order to assign the respective bin indices in every point, we refer to the Z-order function which maps multidimensional data to one dimension. The z-value of a point in multidimensions is simply calculated by interleaving the binary representations of its coordinate values, as seen in Fig.1 and Fig.2.

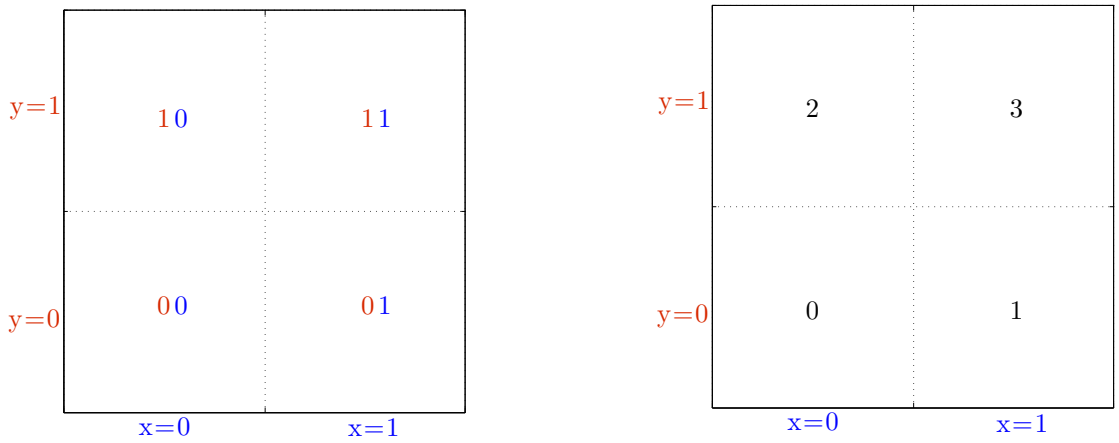

Figure 1: On the left figure we present the binary representation of bin id's while the right figure contains the decimal representation (with the most right bit being the LSB).

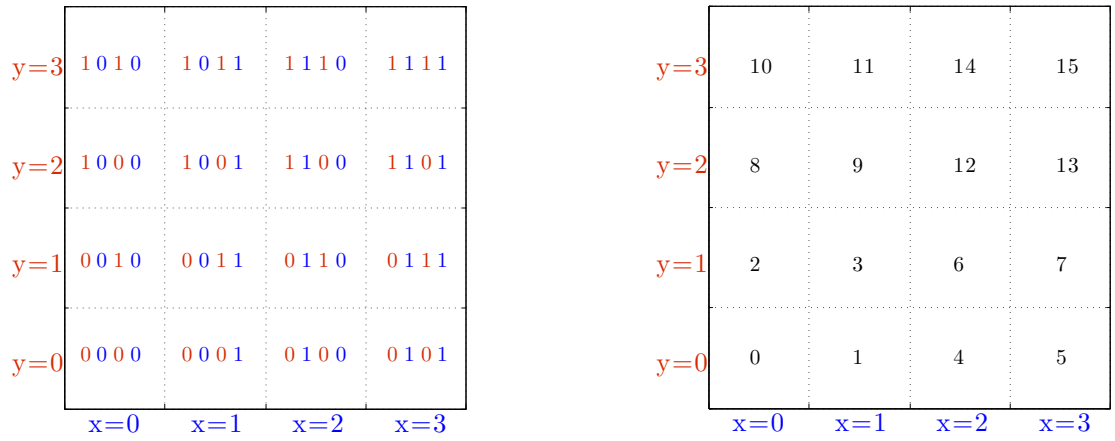

Figure 2: On the left figure we present the binary representation of bin id's while the right figure contains the decimal representation (with the most right bit being the LSB).

4 Binning Optimality

Because of the lack of space-filling designs definition, in [6] "Binning Optimality" is proposed as a measure to quantify the space-filling property. In general, it is desirable for designs to be non-collapsing and space-filling; that is designs that spread points evenly throughout the experimental region.

Let b be a prime number greater than or equal to 2 and P be defined as

$$P = \left\lceil \frac{\log_b n}{M} \right\rceil.$$

Then a design is Binning Optimal with respect to base b, if the following two conditions are met

1. when C^M is divided into a uniform grid of cube bins with volume b^{-PM}, no bin contains more than 1 point
2. when C^M is divided into a uniform grid of cube bins with volume $b^{-(P-1)M}$, every bin contains the same number of points.

A design can be space-filling without being binning optimal, but any binning optimal design is also space-filling. Thus, the space-filling property is ensured by constructing binning optimal designs.

5 Binning Optimal Symmetric Latin Hypercube Sampling (BOSLHS)

5.1 BOSLHS Algorithm

To begin with, we introduce the BOSLHS algorithm, initially presented in [5].

1. Start with $n = 2M$ points that are well distributed in $(0, 1)^M$.
2. Select $n/2$ 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. Reflect the current n points through the origin to create n additional mirror points. 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 set $n = 2n$.
5. Repeat steps 2 through 4 until the desired number of points has been obtained (i.e. until $n = N$).

5.2 Algorithm for Initial Orthogonal Axes LHS Design

In $M = 2^p$ dimensions, $p = 0, 1, 2, \ldots$, a $2M$ point symmetric cell centered Latin Hypercube which constitutes a set of orthogonal axis can be generated by tiling simple patterns. The first pattern determines the magnitudes of the coordinates, while the second pattern dictates the signs of those coordinate magnitudes.

Cell centered coordinate magnitudes between 0 and $2M$ for the first points of BOSLHS design can be generated as

$$|\mathbf{x}_{1,i}| \;=\; 2i - 1, \quad \text{for i=1,2,...,M.} \tag{3}$$

The magnitudes for the second point can be generated by reversing the order within each disjoint pair of consecutive coordinates in $|\mathbf{x}_1|$, that is

$$|\mathbf{x}_2| \;=\; [3, 1, 7, 5, 11, 9, \ldots, 2M - 1, 2M - 3]^T.$$

The magnitudes for $|\mathbf{x}_3|$ and $|\mathbf{x}_4|$ can be generated by reversing the order within every disjoint group of 4 consecutive coordinates in $|\mathbf{x}_1|$ and $|\mathbf{x}_2|$ respectively, that is

$$|\mathbf{x}_3| \;=\; [7,5,3,1,15,13,11,9,\ldots,2M-1,2M-3,2M-5,2M-7]^T$$

and

$$|\mathbf{x}_4| \;=\; [5,7,1,3,13,15,9,11,\ldots,2M-3,2M-1,2M-7,2M-5]^T.$$

The magnitudes for $|\mathbf{x}_{n+1}|$ through $|\mathbf{x}_{2n}|$, $n = 2^p,\ p = 0,1,2,\ldots$, are generated by reversing the order within each disjoint group of n consecutive coordinates for points $|\mathbf{x}_1|$ through $|\mathbf{x}_n|$, respectively. For $M = 8$ dimensions the aforementioned pattern takes the form

$$\left|[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_8]^T\right| = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & 1 & 7 & 5 & 11 & 9 & 15 & 13 \\ 7 & 5 & 3 & 1 & 15 & 13 & 11 & 9 \\ 5 & 7 & 1 & 3 & 13 & 15 & 9 & 11 \\ 15 & 13 & 11 & 9 & 7 & 5 & 3 & 1 \\ 13 & 15 & 9 & 11 & 5 & 7 & 1 & 3 \\ 9 & 11 & 13 & 15 & 1 & 3 & 5 & 7 \\ 11 & 9 & 15 & 13 & 3 & 1 & 7 & 5 \end{bmatrix}. \tag{4}$$

We use the following simple series of Hadamard matrices as the pattern to generate the signs,

$$\begin{bmatrix} + \end{bmatrix}$$

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

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

and

$$\begin{bmatrix} + & - & - & + & - & + & + & - \\ + & + & - & - & - & - & + & + \\ + & - & + & - & - & + & - & + \\ + & + & + & + & - & - & - & - \\ + & - & - & + & + & - & - & + \\ + & + & - & - & + & + & - & - \\ + & - & + & - & + & - & + & - \\ + & + & + & + & + & + & + & \end{bmatrix}. \tag{5}$$

The entrywise product of the signs in (5) and the magnitudes in (4) results in the following set of $M = 8$ orthogonal vectors

$$[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_8]^T = \begin{bmatrix} 1 & -3 & -5 & 7 & -9 & 11 & 13 & -15 \\ 3 & 1 & -7 & -5 & -11 & -9 & 15 & 13 \\ 7 & -5 & 3 & -1 & -15 & 13 & -11 & 9 \\ 5 & 7 & 1 & 3 & -13 & -15 & -9 & -11 \\ 15 & -13 & -11 & 9 & 7 & -5 & -3 & 1 \\ 13 & 15 & -9 & -11 & 5 & 7 & -1 & -3 \\ 9 & -11 & 13 & -15 & 1 & -3 & 5 & -7 \\ 11 & 9 & 15 & 13 & 3 & 1 & 7 & 5 \end{bmatrix}. \tag{6}$$

The vectors in (6) are then reflected through zero to add a set of M mirror points and then the 2M points are translated and scaled to $(0,1)^M$.

5.3 Octant Assignment

Next, we consider producing all the octants in which points will be assigned. The main idea is that nearby points have to be sent in octants that are "maximally-spaced" (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 $M/2$). To derive maximally-spaced octants, we use Hadamard matrices and the Standard array. A Hadamard matrix is a square matrix with elements in $\{\pm 1\}$ and has the property that rows are mutual orthogonal. If the elements of the first row of the matrix are all equal to $\{1\}$, then it is a normalized Hadamard matrix. Normalized Hadamard matrices can be obtained through Sylvester construction as it follows

$$\check{H}_{2M} = \begin{bmatrix} \check{H}_M & \check{H}_M \\ \check{H}_M & -\check{H}_M \end{bmatrix}, \tag{7}$$

starting from the basic matrix

$$\check{H}_1 = \begin{bmatrix} +1 \end{bmatrix}. \tag{8}$$

Then, we construct the matrix $\check{C}$ as

$$\check{C} = \begin{bmatrix} \check{H}_M \\ -\check{H}_M \end{bmatrix}, \tag{9}$$

and denote its $2M$ rows by $\check{c}_1, \check{c}_2, \ldots, \check{c}_{2M}$.

Then, we map the alphabet $\{\pm 1\}$ to the binary field $\mathbb{F}_2$ as it follows

$$\{\pm 1\}: \begin{array}{c} +1 \longleftrightarrow 0 \\ -1 \longleftrightarrow 1 \end{array} : \mathbb{F}_2.$$

The matrix $\check{C}$ in the field $\mathbb{F}_2$ is denoted by C where $C = \begin{bmatrix} H_M \\ \bar{H}_M \end{bmatrix}$ and $\bar{H}_M$ is the complement of H_M that corresponds –under the above mapping– to the normalized Hadamard matrix $\check{H}_M$.

The Standard array is constructed as follow. In the first super row we place $c_1, c_2, \ldots, c_{2M}$. Then, we choose one of the remaining $2^M - 2M$ vectors with minimum weight (weight is defined as the number of its nonzero elements) and place it as first entry in the second super row (coset leader). Subsequently, we fill out the super row by adding coset leader to matrix C and choose a vector from the remaining $2^M - 4M$ (with minimum weight) to place it as first entry in the third super row and so on. This procedure continues until all 2^M vectors in $\mathbb{F}_2^M$ have been used. For clarity purpose, we illustrate the above procedure in Fig.3.

$$\begin{bmatrix} c_1 = \mathbf{0} & c_2 & \cdots & c_i & \cdots & c_{2M} \\ e_2 & e_2 + c_2 & \cdots & e_2 + c_i & \cdots & e_2 + c_{2M} \\ \vdots & \vdots & & \vdots & & \vdots \\ e_k & e_k + c_2 & \cdots & e_k + c_i & \cdots & e_k + c_{2M} \\ \vdots & \vdots & & \vdots & & \vdots \\ e_{\frac{2M}{2M}} & e_{\frac{2M}{2M}} + c_2 & \cdots & e_{\frac{2M}{2M}} + c_i & \cdots & e_{\frac{2M}{2M}} + c_{2M} \end{bmatrix}$$

Figure 3: Standard array for $\{c_1, c_2, \ldots, c_{2M}\}$.

For example, for $M = 4$, code C takes the form

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

and the standard array becomes

$$\begin{bmatrix} 0000 & 0101 & 0011 & 0110 & 1111 & 1010 & 1100 & 1001 \\ 1000 & 1101 & 1011 & 1110 & 0111 & 0010 & 0100 & 0001 \end{bmatrix}. \tag{10}$$

Returning to the alphabet $\{\pm 1\}$, (10) becomes

$$\begin{bmatrix} ++++ & +-+- & ++-- & +--+ & ---- & -+-+ & --++ & -++- \\ -+++ & --+- & -+-- & ---+ & +--- & ++-+ & +-++ & +++- \end{bmatrix}. \tag{11}$$

Interestingly, standard array in (11), contains all 2^M octants in which a point may lie. Moreover, in each super row, all octants starting with '+' have Hamming distance equal to $M/2$ (i.e, are maximally-spaced) while the octants starting with '-' are the complementary octants of the same super row and they will be used in the third step of the BOSLHS algorithm. As a result, points that are close to each other will be sent in the same orientation (with the term orientation we refer to each super row). In case of $M \geq 4$ there are more than two orientations, thus we need a measure to determine the next orientation the points will be sent in. For this purpose, we introduce the definition of the dot product of two orientations $\check{W}^{(i)}$ and $\check{W}^{(j)}$ as initially presented in [5],

$$\text{dot}\left(\check{W}^{(i)}, \check{W}^{(j)}\right) = \max\left(\text{mod}\left(abs\left(\check{W}^{(i)}\left(\check{W}^{(j)}\right)^T\right), M\right)\right), \tag{12}$$

where $\check{W}^{(i)} = \begin{bmatrix} \check{c}_1 \odot \check{e}_i \\ \check{c}_2 \odot \check{e}_i \\ \vdots \\ \check{c}_{2M} \odot \check{e}_i \end{bmatrix}$, $i = 1, 2, \ldots, \frac{2^M}{2M}$, and $\check{e}_i$ is the i-th coset leader in alphabet $\{\pm 1\}$.
Since the following equality holds

$$\delta\left(\check{W}^{(i)}, \check{W}^{(j)}\right) = \left(M - dot\left(\check{W}^{(i)}, \check{W}^{(j)}\right)\right)/2, \tag{13}$$

where $\delta\left(\check{W}^{(i)}, \check{W}^{(j)}\right)$ denotes the minimum Hamming distance, the lower the dot product of two orientations, the higher the Hamming distance separating the two closest octants.

5.4 Sorting the List of Orientations into Maximally-Spaced Order

In $M = 16$ dimensions there are $2^{16}/32 = 2048$ orientations and coset leaders. We can split these leaders into two groups. In the first group, leaders have an even number of '-' and in the second group leaders have an odd number of '-'. First group can be divided into 16 sub-groups of 64 leaders each, such that the minimum Hamming distance between sub-groups is 2 and the minimum Hamming distance within a sub-group is 4 (i.e., the dot product into sub-groups is less than or equal to 8). These sub-groups are constructed by "Step 2 sort" in [6]. The basic idea is that $\binom{1024}{2} = 523,776$ dot products are calculated and stored in a symmetric matrix $D_{1024\times 1024}$ and then D is sorted, in order to obtain the list of maximally spaced orientations. We observe that D is symmetric, since $D_{i,j} = \text{dot}\left(\check{W}^{(i)}, \check{W}^{(j)}\right) = \text{dot}\left(\check{W}^{(j)}, \check{W}^{(i)}\right) = D_{j,i}$.
In this work an alternative approach is given, since memory requirements preclude even listing the $2^M/2M$ leaders for $M \geq 32$.

5.4.1 Alternative Approach For M=16

Since leader $\mathbf{e}_1 = \mathbf{0}^T_{16\times 1}$ is included in the first sub-group and we wish minimum Hamming distance of 4, we start from the leaders of weight 4. Let us denote by $\underline{x}^{(i)} = [x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}]$ the vector containing indices of units in leader i. We choose as first leader the one with minimum indices $\{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}\}$ among all possible leaders. The respective vector of units of the second leader $\underline{x}^{(2)}$ has to differ in at least 2 indices with the previous leaders (i.e., $\underline{x}^{(1)}$) and $\{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}\}$ have to be as close as possible to $\{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}\}$ ($x_1^{(2)}$ is more important to be closest to $x_1^{(1)}$ than $x_2^{(2)}$ to $x_2^{(1)}$, $x_2^{(2)}$ is more important to be closest to $x_2^{(1)}$ than $x_3^{(2)}$ to $x_3^{(1)}$ and $x_3^{(2)}$ is more important to be closest to $x_3^{(1)}$ than $x_4^{(2)}$ to $x_4^{(1)}$). The set of indices of the third leader $\underline{x}^{(3)}$ has to differ in at least in 2 indices with the previous leaders (i.e $\underline{x}^{(1)}$, $\underline{x}^{(2)}$) and the set $\{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}, x_4^{(3)}\}$ has to be as close as possible to $\{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}\}$. This procedure continues until we are unable to construct any more leaders with these properties. Then, we construct leaders of weight 6. We choose as first leader the one with minimum indices among all possible leaders of weight 6. The set of indices of second leader $\{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}, x_5^{(2)}, x_6^{(2)}\}$ of weight 6 has to differ in at least 3 indices with the leaders of weight 4 and at least in 2 indices with leaders of weight 6. Then, the set $\{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}, x_5^{(2)}, x_6^{(2)}\}$ has to be as close as possible to $\{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}, x_5^{(1)}, x_6^{(1)}\}$. The procedure continues until we are unable to construct any more leaders with these properties. Quite interestingly, not only the minimum Hamming distance separating leaders is 4, but all octants that can be generated from these leaders have a minimum Hamming distance of 4 as well. Most importantly, this procedure delivers constructively in the same sets of indices as the procedure "Step 2 sort" in [6] without imposing costly comparisons and sorting.

5.4.2 Alternative Approach For M=32

Our objective is to produce the first sub-group in $M = 32$ dimensions which according to [6] includes $2^{20} = 1,048,576$ leaders and the minimum Hamming distance separating octants is 4. Since the zero leader $\mathbf{e}_1 = \mathbf{0}^T_{32\times 1}$ is included in this sub-group too, we focus on constructing leaders of weights 4, 6, 8, 10 and 12. Leaders of weight 14 do not exist. Following the procedure described above, we were able to construct 29,016 leaders of weights 4 and 6.

5.5 Example of BOSLHS Design with 16 Points in the 2-Dimensional Case

First we create the $2M$ initial points based on the pattern which was described in subsection 5.2

$$[\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4]^T = \begin{bmatrix} 1 & 3 \\ 3 & -1 \\ -1 & -3 \\ -3 & 1 \end{bmatrix}. \tag{14}$$

Afterwards, we scale them to domain $(0,1)^M$ dividing each element of $[\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4]^T$ by $2 \times$ number of points and then by adding $\frac{1}{2}$ to all elements. In this way we create the following first $2M$ points for the BOSLHS design (the Table and the Figure below illustrates the computed initial points).

$$[\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4]^T = \begin{bmatrix} 0.625 & 0.875 \\ 0.875 & 0.375 \\ 0.375 & 0.125 \\ 0.125 & 0.625 \end{bmatrix}.$$

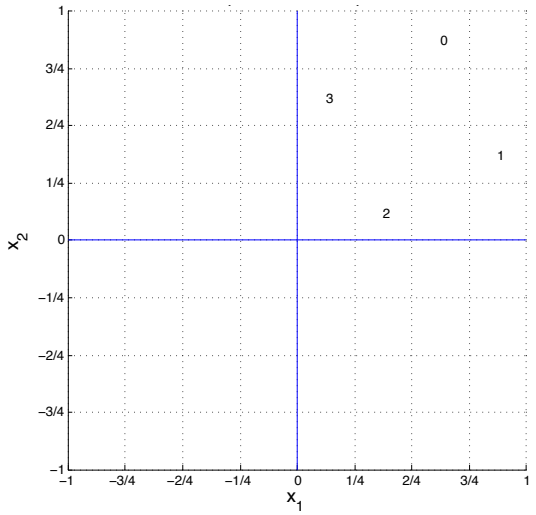

Figure 4: Initial points of BOSLHS design

In the case of two dimensions, there is only one orientation. Thus, the selection of the orientation to assign the points to is trivial. The unique orientation is $[++ \quad +- \quad -- \quad -+]$ and we choose randomly, between $[++]$ and $[+-]$, the octants in which the points will be sent. The only hard restrictions (in this 2-dimensional case) are:

- 2 and 1 should not end up in the same octant
- 3 and 0 should not end up in the same octant

Taking into account the above constrains, we produce an instance points assignment depicted in the following Figure and Table.

Point	Bin Id	Octant
2	0	++
1	1	+−
3	2	+−
0	3	++

Figure 5: Random octant assignment

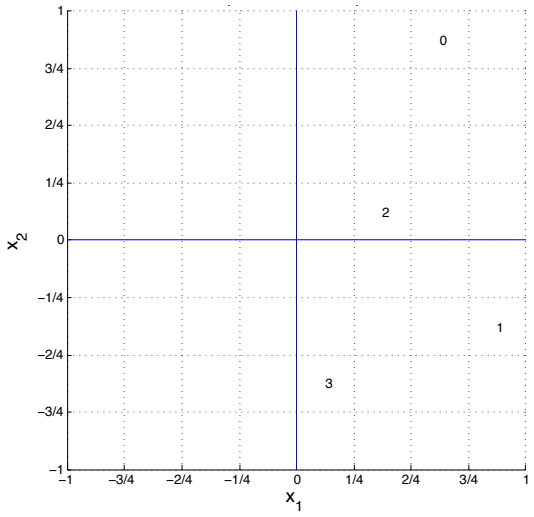

Figure 6: Points assignment according the left table

Then, we add four more points in $(0,1)\otimes(-1,1)^{M-1}$, which are reflections of the current four points through the origin, we translate these eight points from $(-1,1)^M$ to $(0,2)^M$ and, afterwards, we scale them to $(0,1)^M$ (Fig.7 and Fig.8).

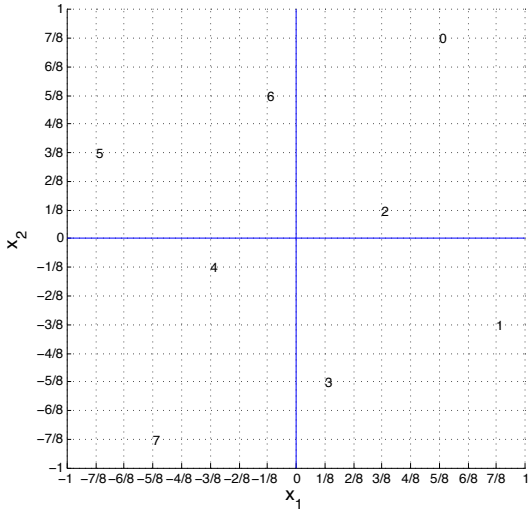

Figure 7: Step 3 of BOSLHS design

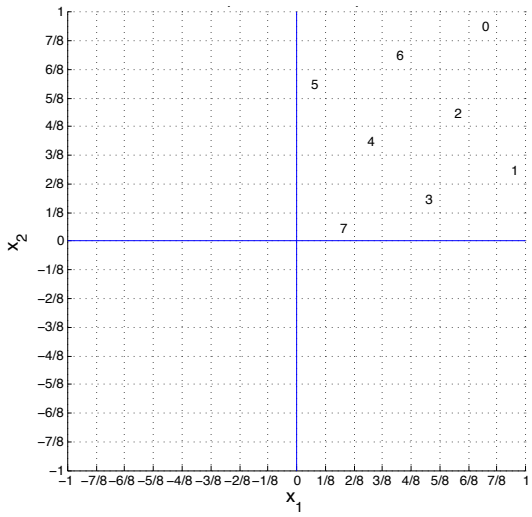

Figure 8: Step 4 of BOSLHS design

In Fig.9 we observe that, as expected, the design is binning optimal (bin indices have been computed as described in Section 3).

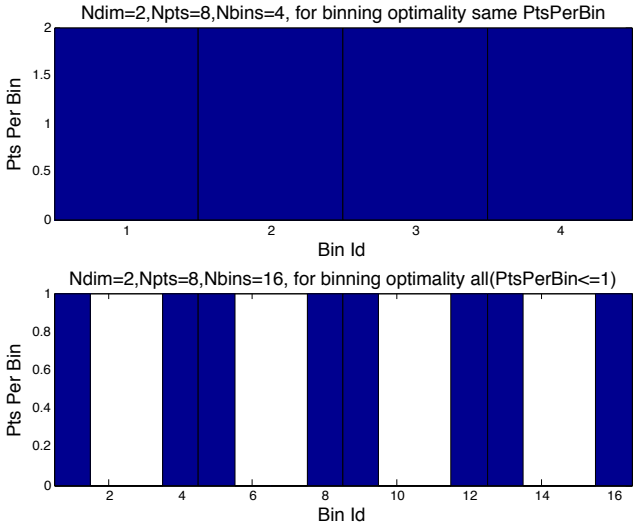

Figure 9: Binning Optimality.

In the sequel, we select an arbitrary octant between [++] and [+−]. The only hard restrictions are:

- 7 and 4 should not end up in the same octant
- 3 and 1 should not end up in the same octant
- 5 and 6 should not end up in the same octant
- 2 and 0 should not end up in the same octant.

Taking into account the above constrains, we produce an instance points assignment depicted in the following Figure and Table.

Point	Bin Id	Octant
7	0	++
4	3	+−
3	4	+−
1	7	++
5	8	+−
6	11	++
2	12	++
0	15	+−

Figure 10: Random octant assignment

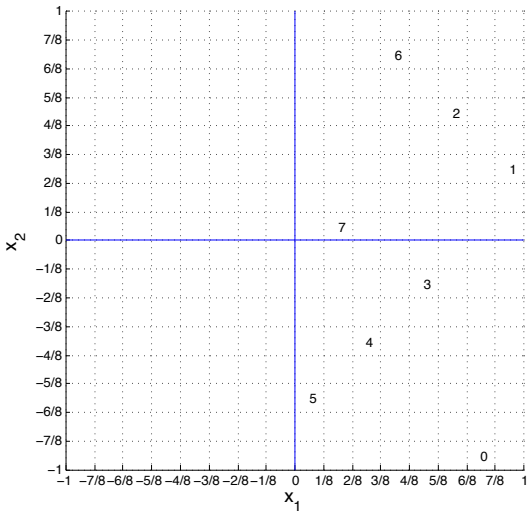

Figure 11: Points assignment according the left table

Then, we add 8 more points in $(0,1) \otimes (-1,1)^{M-1}$, which are reflections of the current 8 points through the origin, we translate these sixteen points from $(-1,1)^M$ to $(0,2)^M$, and scale them to $(0,1)^M$ (Fig.12 and Fig.13). Fig.14 validates that the design remains binning optimal.

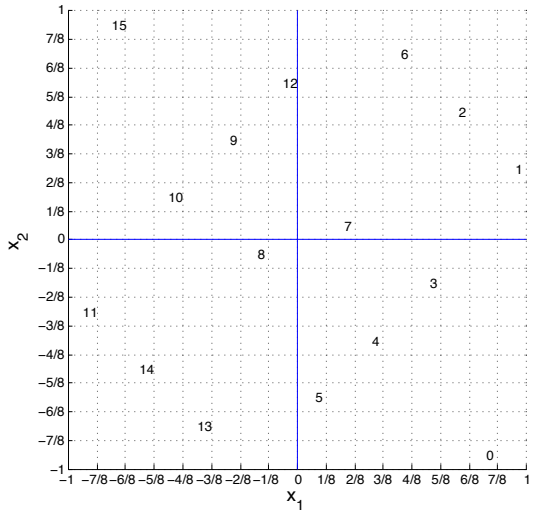

Figure 12: Step 3 of BOSLHS algorithm.

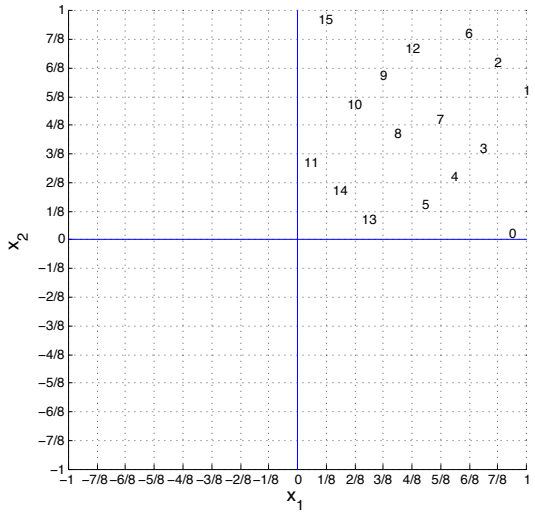

Figure 13: Step 4 of BOSLHS algorithm.

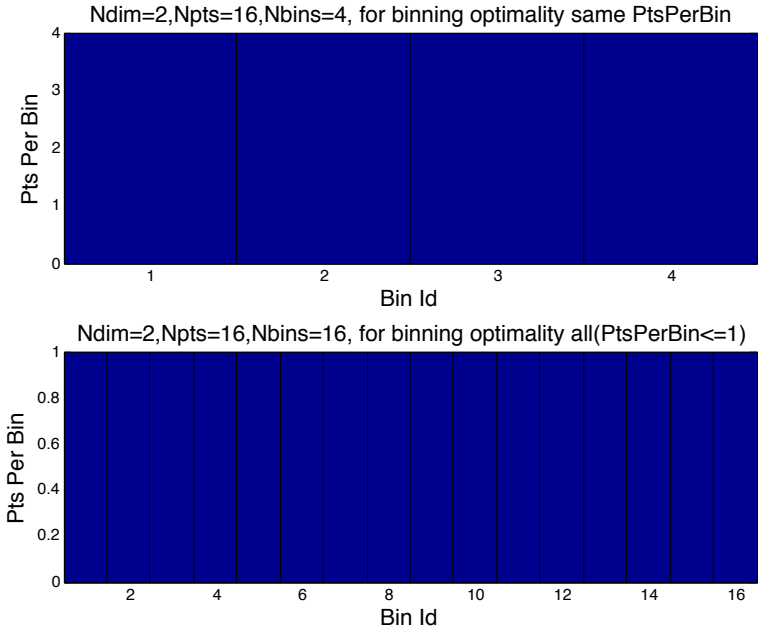

Figure 14: Binning optimality.

5.6 Example of BOSLHS Design with 32 Points in 4-Dimensional Case

First, we create the $2M$ initial points through the pattern which was described in subsection 5.2

$$[\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots, \underline{x}_8]^T = \begin{bmatrix} 1 & -3 & -5 & 7 \\ 3 & 1 & -7 & -5 \\ 7 & -5 & 3 & -1 \\ 5 & 7 & 1 & 3 \\ -1 & 3 & 5 & -7 \\ -3 & -1 & 7 & 5 \\ -7 & 5 & -3 & 1 \\ -5 & -7 & -1 & -3 \end{bmatrix}. \tag{15}$$

Afterwards, we scale them to domain $(0,1)^M$ dividing each element of $[\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots, \underline{x}_8]^T$ by $2\times$*number of points* and then by adding $\frac{1}{2}$ to all elements. In this way, we create the first $2M$ points for the BOSLHS design (the Table and the Figure below illustrates the computed initial points).

$$[\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots, \underline{x}_8]^T = \begin{bmatrix} 0.5625 & 0.3125 & 0.1875 & 0.9375 \\ 0.6875 & 0.5625 & 0.0625 & 0.1875 \\ 0.9375 & 0.1875 & 0.6875 & 0.4375 \\ 0.8125 & 0.9375 & 0.5625 & 0.6875 \\ 0.4375 & 0.6875 & 0.8125 & 0.0625 \\ 0.3125 & 0.4375 & 0.9375 & 0.8125 \\ 0.0625 & 0.8125 & 0.3125 & 0.5625 \\ 0.1875 & 0.0625 & 0.4375 & 0.3125 \end{bmatrix}$$

Point	Bin Id
0	15
1	5
2	3
3	9
4	0
5	10
6	12
7	6

The above are illustrated in Figure 15 on page 19. In case of 4 dimensions there are two orientations, thus the choice of which orientation to assign the points is not trivial any more. The two orientations are illustrated below

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

Points {4,2,1,7} which are nearby (according to Z-order) will be assigned in one randomly chosen orientation and points {3,5,6,0} will be assigned to the other orientation. The order that points will be assigned in octants is also random (Table 1 and Figure 16 on page 20).

Point	Bin Id	Octant	Point	Bin Id	Octant
4	0	$+-+-$	3	9	$++-+$
2	3	$++++$	5	10	$+++-$
1	5	$++--$	6	12	$+-++$
7	6	$+--+$	0	15	$+---$

Table 1: Random orientation and octant assignment

Then we add eight more points in $(0,1)\otimes(-1,1)^{M-1}$ which are reflections of the current eight points through the origin, we translate these sixteen points from $(-1,1)^M$ to $(0,2)^M$ and scale them to $(0,1)^M$ (Figures 17 and 18).
As above, points {9,7,14,2} will be assigned in one randomly chosen orientation and the next set {12,0,11,5} to the other orientation. The same procedure is being followed for the remaining points (Table 2 and Figures 19 , 20 and 21 on pages 23 , 24 and 25 respectively).

Point	Bin Id	Octant	Point	Bin Id	Octant	Point	Bin Id	Octant	Point	Bin Id	Octant
9	0	$+++$	12	4	$+--$	13	8	$++-$	10	12	$+-+-$
7	1	$++--$	0	5	$+-++$	3	9	$++-+$	6	13	$+--+$
14	2	$+--+$	11	6	$++-+$	8	10	$+-++$	15	14	$++--$
2	3	$+-+-$	5	7	$++--$	4	11	$+---$	1	15	$++++$

Table 2: Random orientation and octant assignment, as described above.

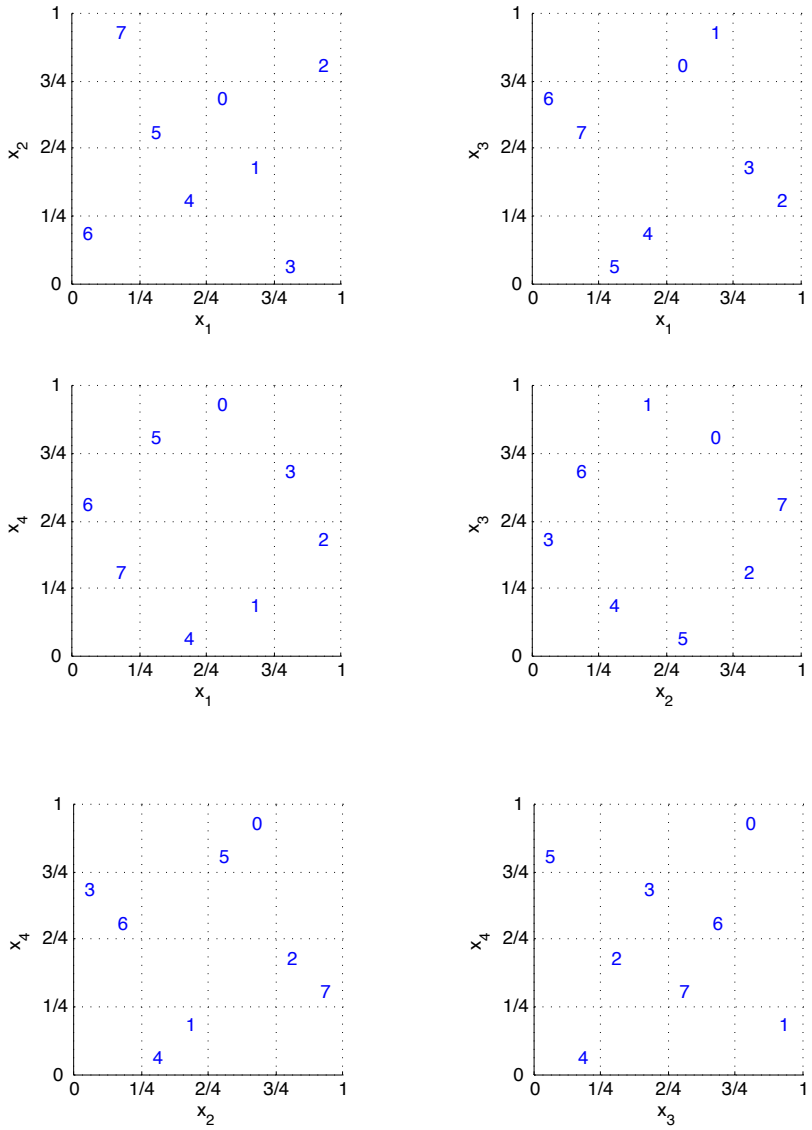

Figure 15: Plots of the projection of the initial points of BOSLHS design onto each pair of dimensions.

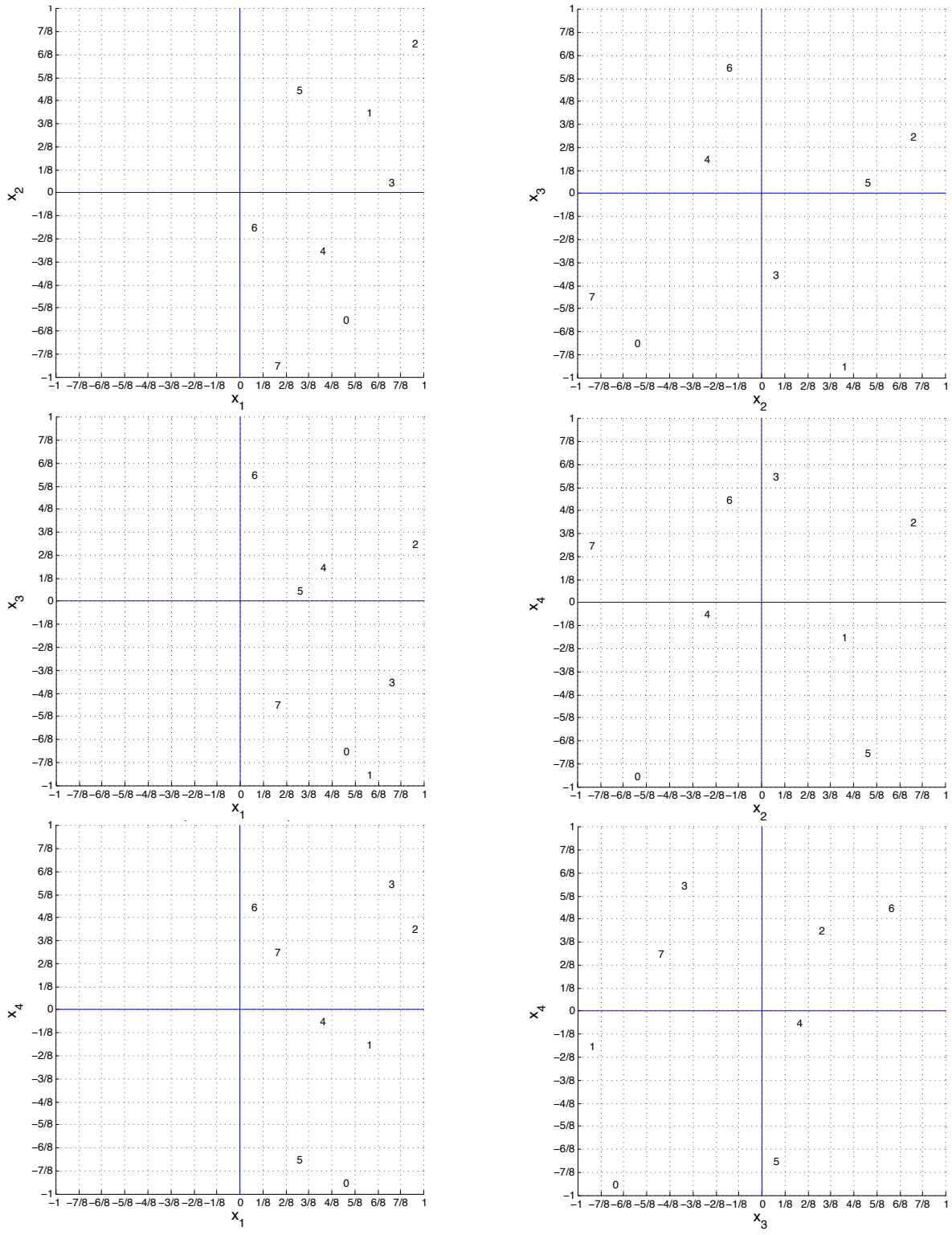

Figure 16: Step 2 of BOSLHS algorithm.

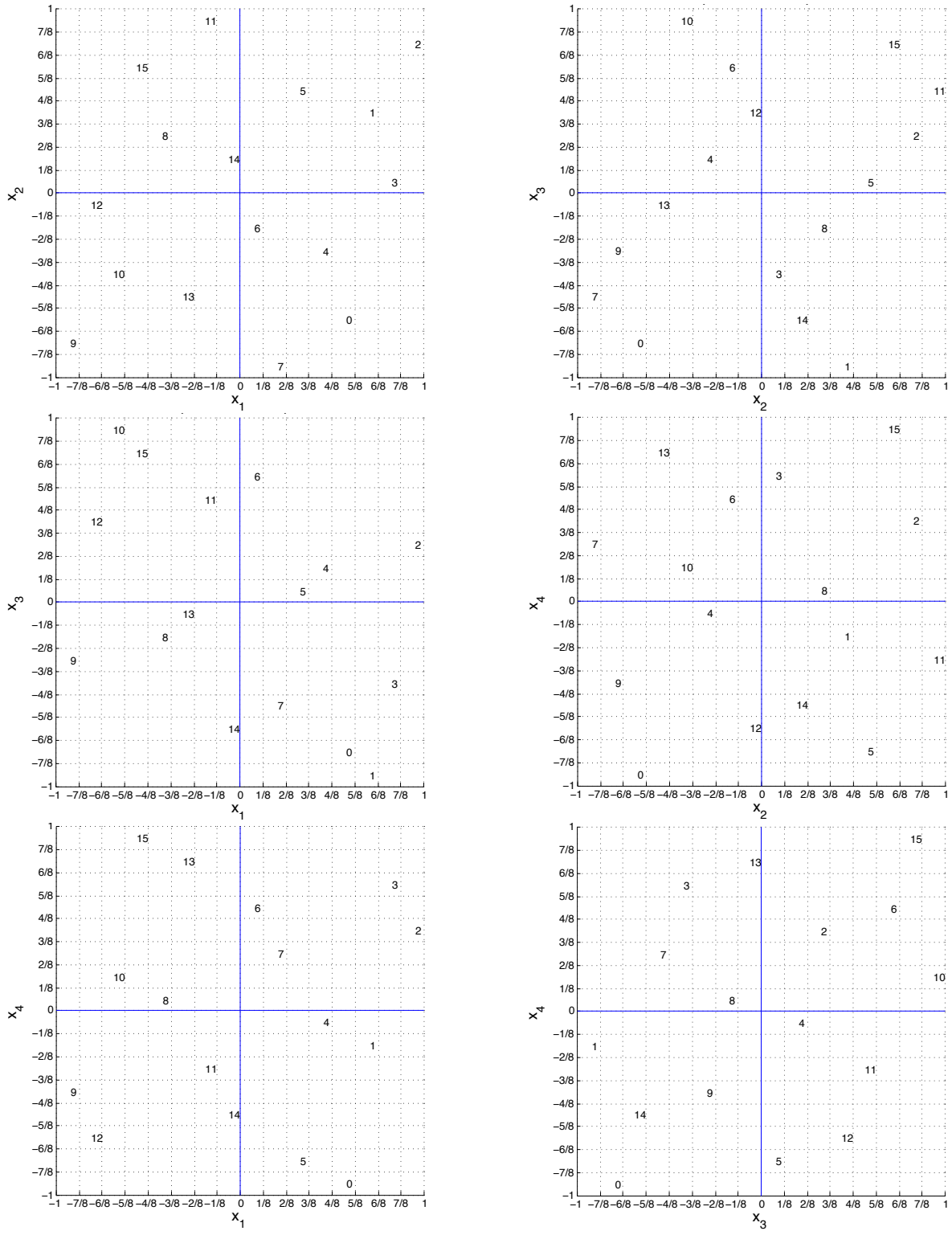

Figure 17: Step 3 of BOSLHS algorithm.

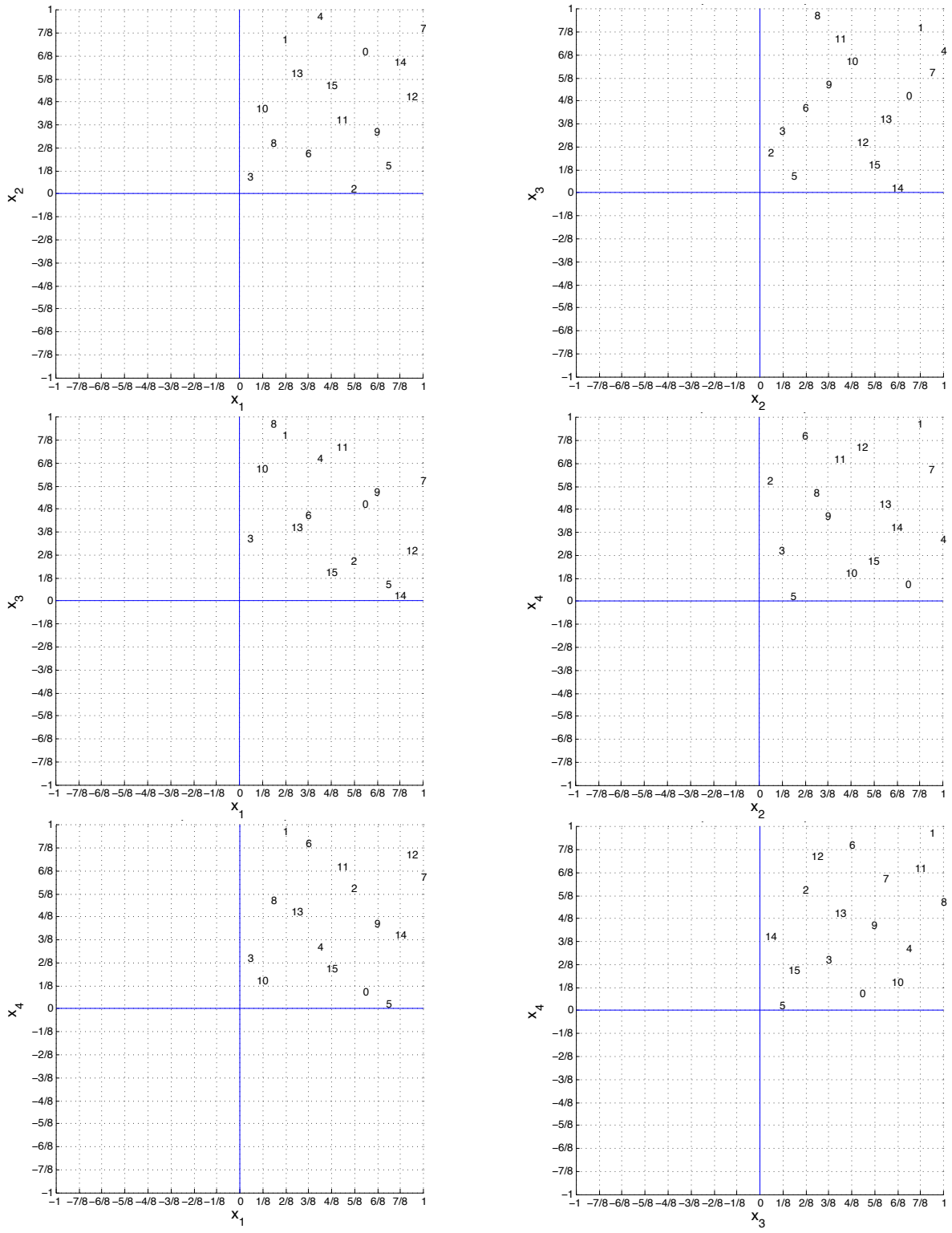

Figure 18: Step 4 of BOSLHS algorithm.

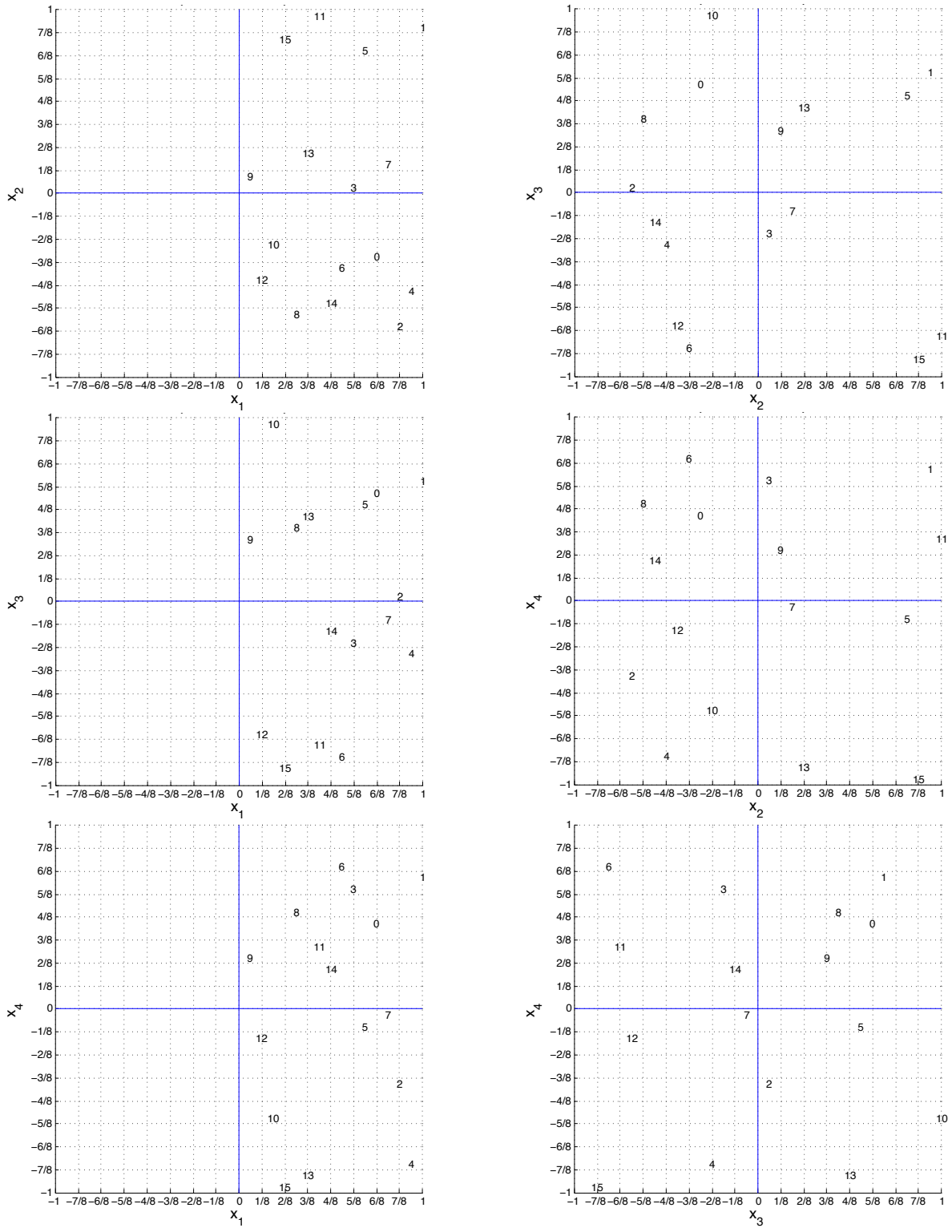

Figure 19: Step 2 of BOSLHS algorithm.

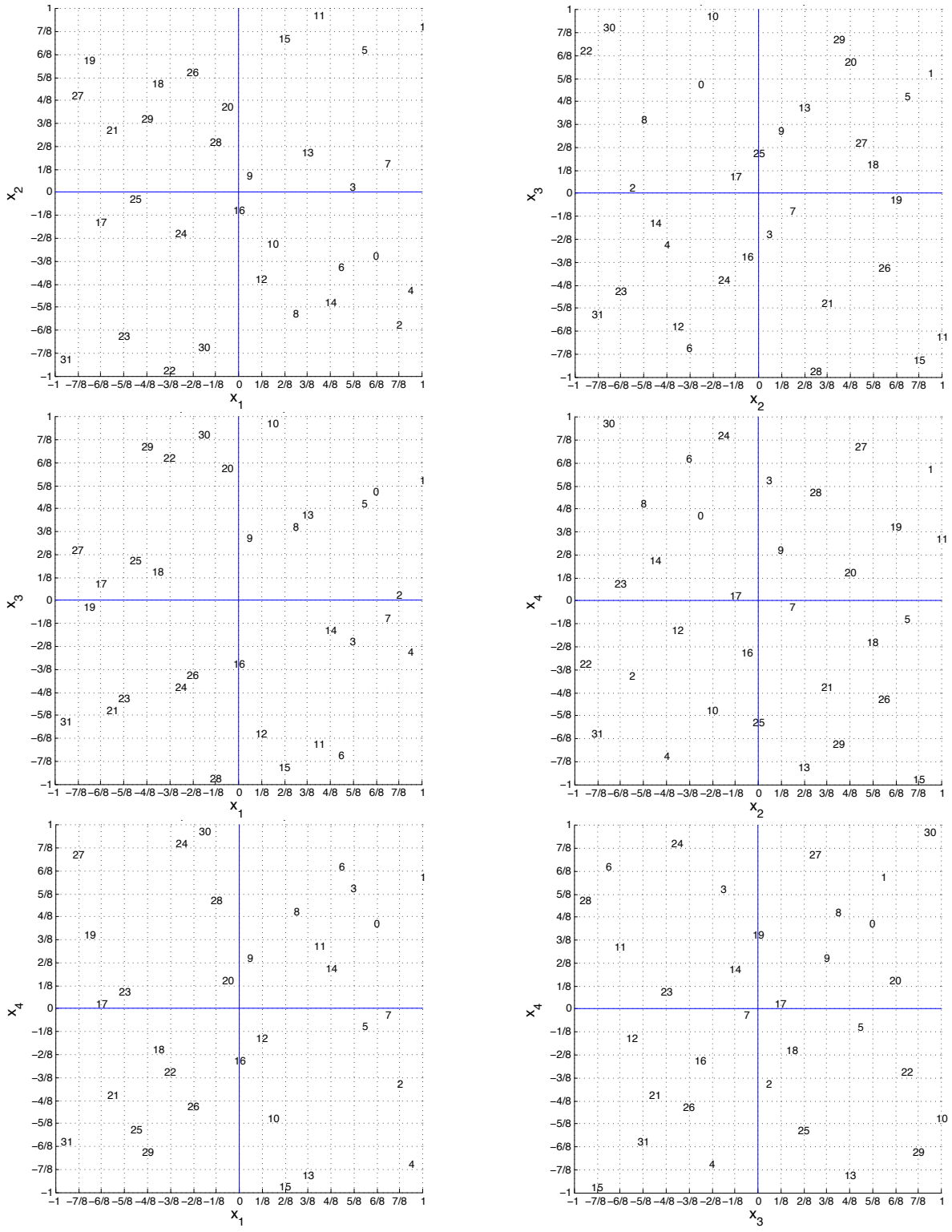

Figure 20: Step 3 of BOSLHS algorithm.

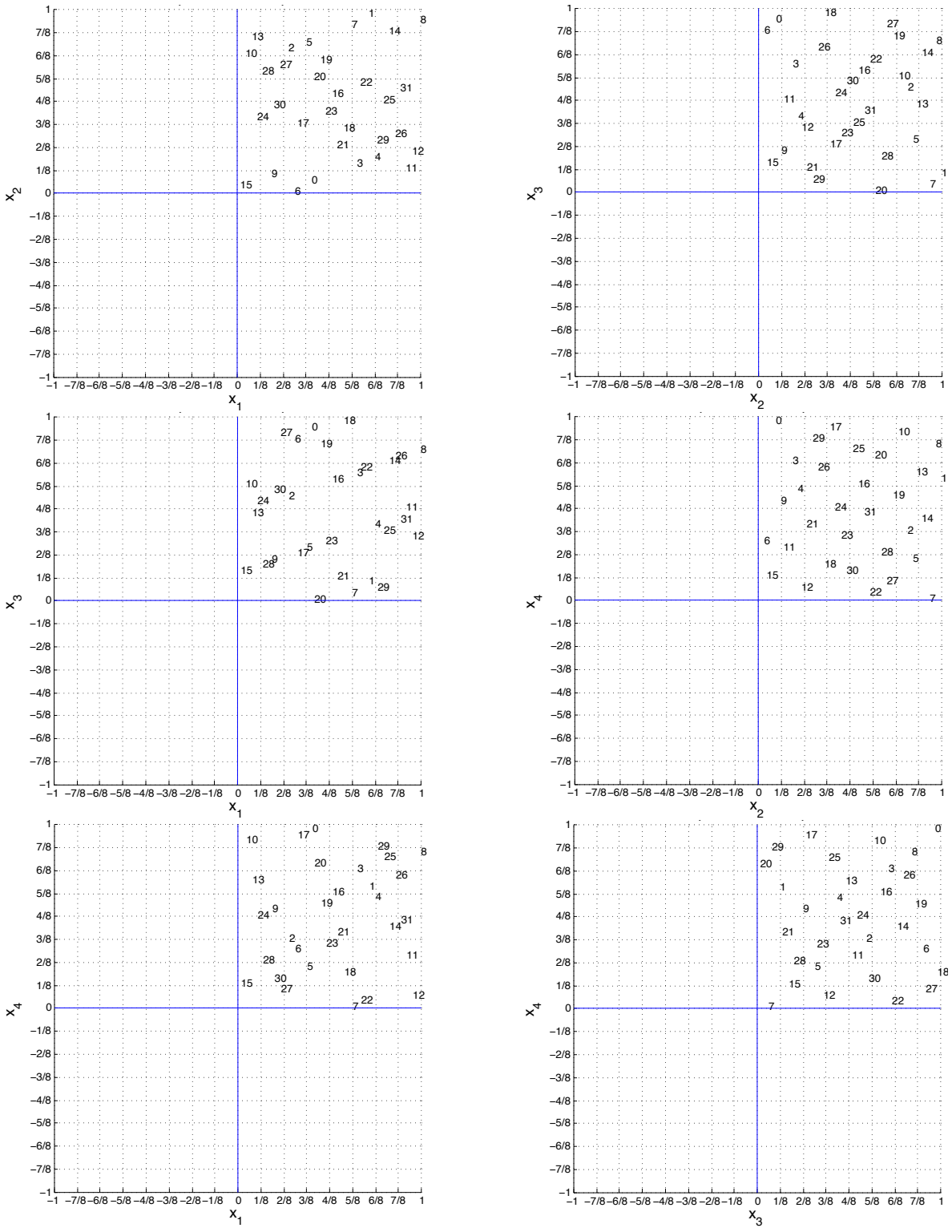

Figure 21: Step 4 of BOSLHS algorithm.

6 BOSLHS v.s. Various Techniques of Sampling

In this section BOSLHS designs are compared with various methods of sampling according to centered- discrepancy ($\mathcal{CL}_2$) metric described in sub-section 2.1. These methods are Tensor Product Sampling (TPS), Monte Carlo Sampling (MCS), Jittered Sampling (JS), and Cell-centered Latin Hypercube Sampling (LHS) with randomly paired dimensions. A brief description of these techniques follows.

6.1 Tensor Product Sampling (TPS)

In the case of 1-dimension sampling, we split the domain into q equal intervals and we place the points at the center of these intervals. In the case of two or more dimensions, we split the domain into q equal bins and put the points into the center of each bin (Fig.22).

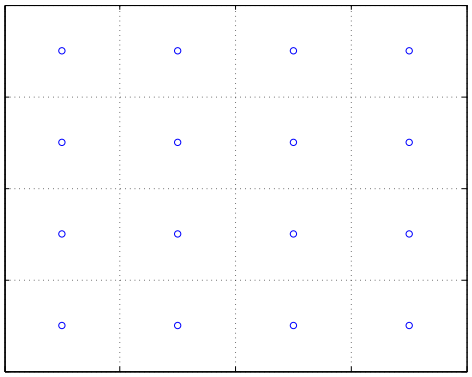

Figure 22: TPS in 2 dimensions. The design has bad 1-D projections and is binning optimal since the binning optimality criterion is satisfied.

6.2 Jittered Sampling (JS)

Jittered Sampling is a combination of Tensor Product Sampling and Monte Carlo Sampling. Specifically, its only difference with the Tensor Product Sampling is that points do not lie in the center of the bins, but there is a uniformly distributed offset, that place them randomly inside the bin (Fig.23).

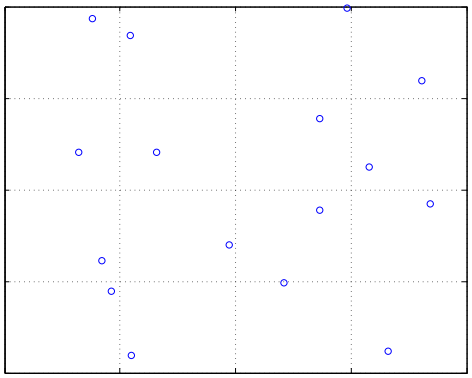

Figure 23: Jittered Sampling (JS) in 2 dimensions. The design has better 1D projections than Tensor Product Sampling and is Binning Optimal.

6.3 Cell Centered Latin Hypercube Sampling (CCLHS)

A Cell-centered Latin Hypercube Sampling can be defined as follows. Let $C = [0,1]^M$ denote the M-dimensional hypercube. For $i = 1,\ldots,N$ and $j = 1,\ldots,M$, let $x_i(j)$ denote the j-th coordinate of the i-th point x_i, let P_j denote M random permutations of the set $\{1,\ldots,N\}$, and, $P_j(i)$ denote the i-th element of P_j. Choosing

$$x_i(j) = \frac{P_j(i) - 0.5}{N}, \quad \text{for } i{=}1,\ldots,N \text{ and } j{=}1,\ldots,M, \tag{17}$$

we obtain a centered LHS for which all the points are located at the centers of their bins. As a result, in each row and each column there is exactly one point (Fig.24).

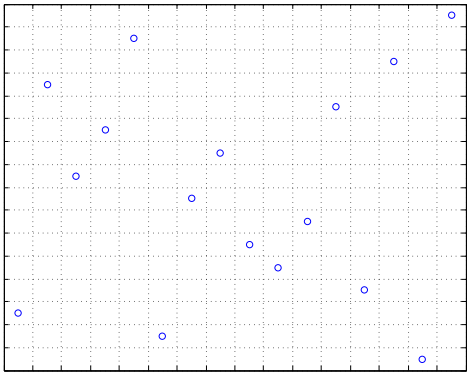

Figure 24: Cell-centered Latin Hypercube Sampling (CCLHS) in 2 dimensions. The design has perfect 1-D projections but is not binning optimal.

Tables 3, 4 and Fig.25, Fig.26 demonstrate that BOSLHS designs achieve smaller $\mathcal{CL}_2$ than any of the above techniques.

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	0.1554	0.1874	0.3471	1.0027	0.6143
16	0.1102	0.1278	0.2534	0.1966	0.342
32	0.0655	0.0863	0.1853	0.1888	0.342
64	0.0418	0.0591	0.126	0.1908	0.342
128	0.027	0.0417	0.0925	0.0693	0.2156
256	0.0169	0.0293	0.0629	0.0352	0.1648
512	0.0107	0.02	0.0448	0.0346	0.1648
1024	0.0071	0.0146	0.0308	0.0195	0.1299
2048	0.0047	0.0105	0.0233	0.0126	0.1091
4096	0.003	0.0072	0.0161	0.0061	0.0816
8192	0.0019	0.0052	0.0111	0.0046	0.0723
16384	0.0012	0.0036	0.0077	0.0028	0.0592
32768	8.3778e-4	25e-4	56e-4	19e-4	0.0501
65536	5.3263e-4	18e-4	41e-4	11e-4	0.0407

Table 3: The average, over 40 trials (except for the 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, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

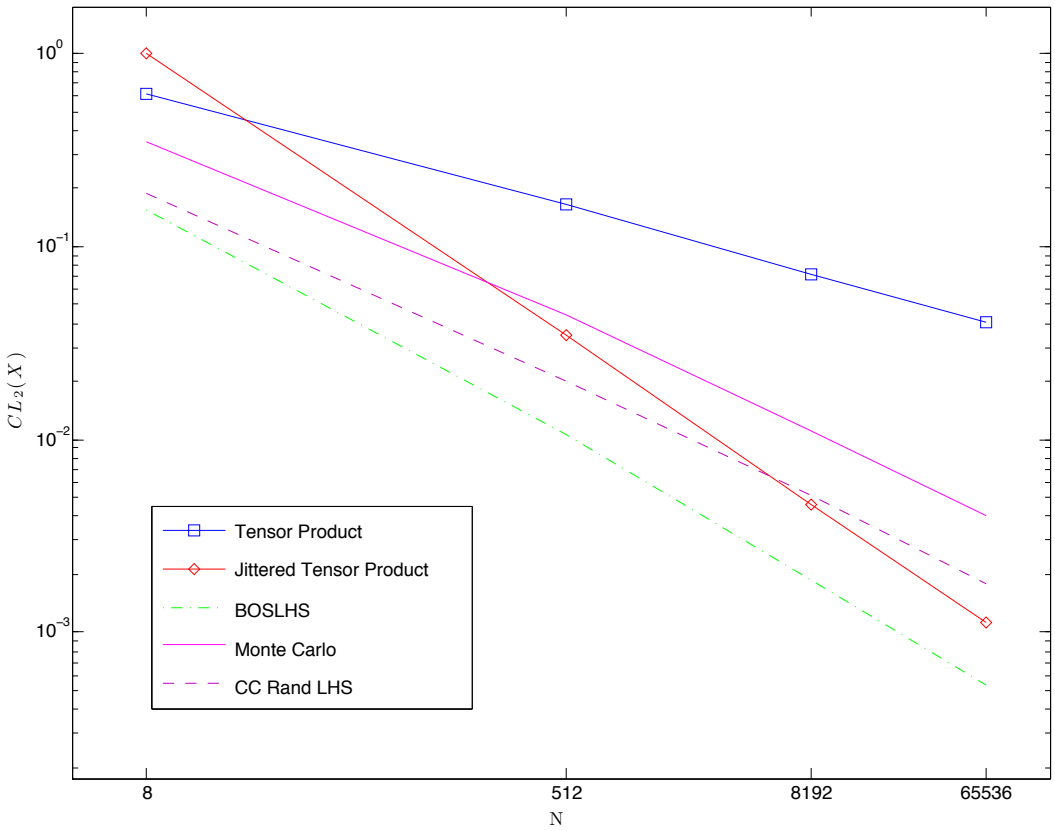

Figure 25: The mean $\mathcal{CL}_2$ discrepancy, over 40 independent realizations, of N points in 4 dimensions.

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
8	0.2840	0.5044	0.7059	1.9648	0.9472
16	0.2840	0.3419	0.5113	1.9548	0.9472
32	0.2112	0.2373	0.3503	1.94	0.9472
64	0.1432	0.1669	0.2501	1.96	0.9472
128	0.0974	0.1169	0.1791	2.039	0.9472
256	0.0671	0.082	0.1256	0.0994	0.6056
512	0.0434	0.0582	0.089	0.0987	0.6056
1024	0.0295	0.0409	0.0619	0.098	0.6056
2048	0.0203	0.0291	0.0434	0.0996	0.6056
4096	0.0138	0.0206	0.0319	0.0167	0.6056
8192	92e-4	146e-4	225e-4	0.0167	0.3548
16384	64e-4	102e-4	157e-4	0.0167	0.3548
32768	43e-4	71e-4	112e-4	0.0167	0.3548
65536	29e-4	51e-4	78e-4	47e-4	0.2781

Table 4: The average, over 40 trials (except for the 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, Jittered Sampling and Tensor Product Sampling as a function of the number of samples.

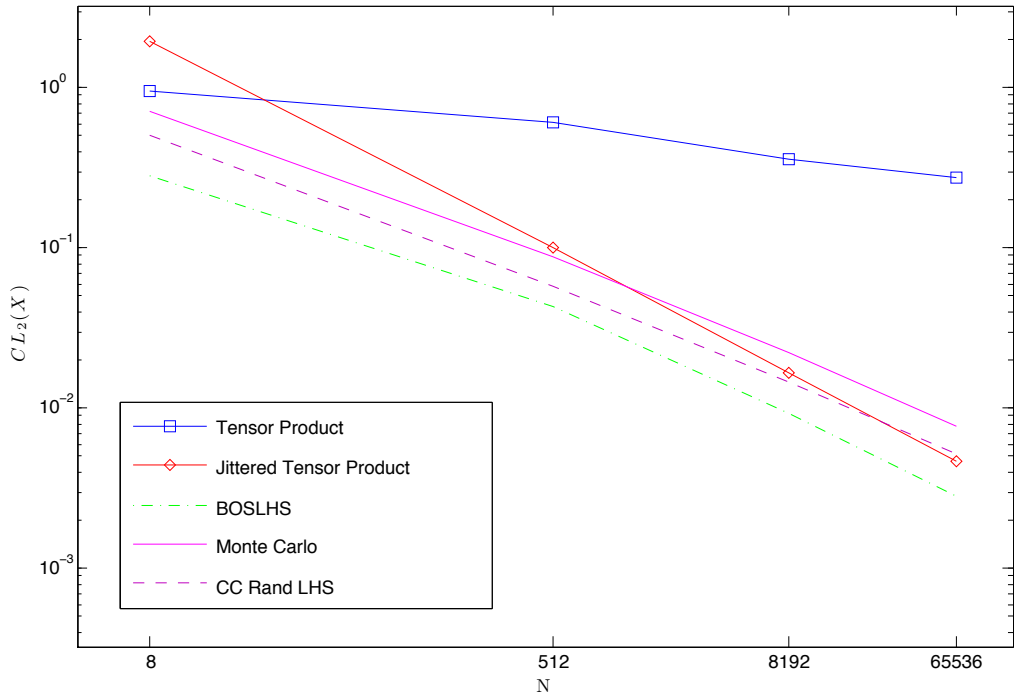

Figure 26: The mean $\mathcal{CL}_2$ discrepancy, over 40 independent realizations, of N points in 8 dimensions.

References

[1] Husslage, B. G. M., Rennen, G., van Dam, E. R. and den Hertog, D., "Space-filling Latin hypercube designs for computer experiments," 2006.

[2] H. Niederreiter, "Random number generation and quasi-Monte Carlo methods," SIAM, Philadelphia, 1992. MR **93h:**65008

[3] F. J. Hickernell, *A generalized discrepancy and quadrature error bound*, Math. Comp. **67** (1998), 299-322. MR **98c:**65032

[4] Fang, K.T., Ma, C.X. and Winker, P. (2002), *Centered L_2-discrepancy of Random Sampling and Latin hypercube Design, and Construction of Uniform Designs*, Mathematics of Computation, 71, 275-296

[5] K. R. Dalbey and G. N. Karystinos, Fast generation of space-filling Latin Hypercube Sample designs, In *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September 2010. AIAA 2010-9085.

[6] Dalbey, Keith Richard, George N. Karystinos, "Generating a Maximally Spaced Set of Bins to Fill for High Dimensional Space-filling Latin Hypercube Sampling," *Journal Article, International Journal for Uncertainty Quantification*, Accepted/Published December 2011.

[7] Saka Y, Gunzburger M, Burkardt J (2007) "Latinized, improved LHS, and CVT point sets in hypercubes," *International Journal of Numerical Analysis and Modeling*, Volume 4, Number 3-4, Pages 729-743.

[8] H. Niederreiter, "Random Number Generation and Quasi-Monte Carlo Methods", (*Society for Industrial and Applied Mathematics*, 1992)

[9] J. C. Moreina, P.G, Farell,"Essensials of Error-Control Coding", Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005. 98.