ON HOW TO IMPLEMENT AN AFFORDABLE OPTIMAL LATIN HYPERCUBE

Felipe Antonio Chegury Viana, fchegury@mecanica.ufu.br
Valder Steffen, Jr, vsteffen@mecanica.ufu.br
Federal University of Uberlândia, School of Mechanical Engineering, Avenida João Naves de Ávila, 2121, Campus Santa Mônica, 38400-902, Uberlândia, Minas Gerais, Brazil.

Gerhard Venter, gventer@vrand.com
Vladimir Balabanov, vladimir@vrand.com
Vanderplaats Research and Development, Inc., 1767 S. 8th Street, Colorado Springs, CO, 80906, USA.

Abstract. *For meta model generation, the choice of experimental data points is very important. Design of Experiments provides a tool that can aid the process of point selection in the design space. One popular design of experiments technique is the Latin Hypercube technique. This is a fast-to-generate design, that is widely used. However, due the random nature of the generation process, the Latin Hypercube design can produce a bad design for fitting a meta model. The Optimal Latin Hypercube design was introduced to overcome this difficulty. It satisfies the same requirements as the Latin Hypercube design, but ensures that the points are distributed to uniformly cover the design space. A problem with using the Optimal Latin Hypercube design is the computational cost required to generate the design. This paper describes an optimization algorithm for generating the Optimal Latin Hypercube design in a cost efficient manner. A method for directly generating the design without performing optimization is also presented. Finally, numerical results are reported, illustrating the proposed methodologies.*

Keywords: *Design of Experiments, Optimal Latin Hypercube, Approximation Methods.*

1. INTRODUCTION

Approximation methods have been widely used in engineering design to reduce the required computational cost (Bates et al., 2004; Jin et al. 2005). For example, the Ford Motor Company reports that a single crash simulation of a car can take between 36 and 160 hours to complete (Gu, 2001). Approximation methods, for example Design of Experiments (DOE) combined with Response Surface Methodology (RSM), is used to approximate the objective or constraint functions using a small number of numerical simulations. The approximation, or meta model, is then used in place of the numerical simulation when performing the optimization task, design space exploration and reliability analysis (Simpson et al., 2004). Bates et al. (2004) justified the use of this approach based on two needs:

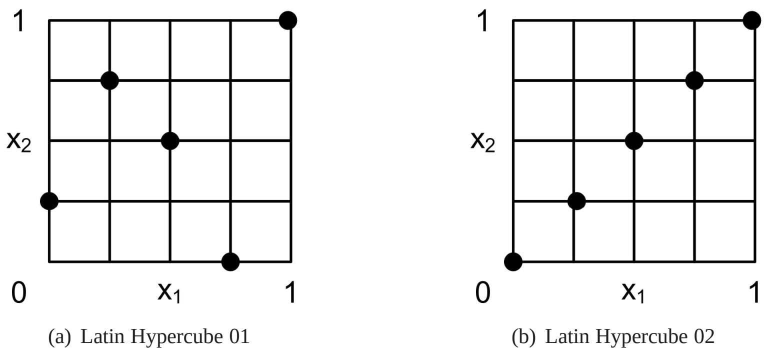
1. To minimize the number of responses evaluations
2. To reduce the effect of numerical noise

Several commonly used approaches are available to achieve these goals e.g., RSM and Kriging approximations. Both these approaches require the evaluation of the response functions at a number of design points when constructing the resulting approximations. Typically, DOE is used to identify the design points at which to evaluate the response functions. Each point in the design space is defined by a specific combination of the input parameters (design variables). The evaluation of the response functions may constitute physical experiments or computer simulations. Once the response values are known at the DOE points, an approximation is constructed that provides the user with an explicit, approximate relationship between the design variables and responses. Such an approximation model is often referred to as a meta model. Design of Experiments aims to extract as much information as possible from a limited number of design points. There are many different criteria available for creating a design of experiments. One such criterion is a space filling design that aims to cover as much of the design space. One well known space filling design is the Latin Hypercube design, proposed by McKay et al. (1979) and Iman and Conover (1980).

There are several advantages to using the Latin Hypercube design:

- The number of samples (points) is not fixed
- Orthogonality of the sampling points
- The sampling points do not depend on the meta model that will be constructed

The Latin Hypercube design is constructed in such a way that each one of the M design variables is divided into N equal levels and that there is only one point (or experiment) for each level (Bates et al., 2004). The final Latin Hypercube design then has N samples. Figure 1 shows two possible Latin Hypercube designs for $M = 2$ and $N = 5$. Note that the Latin Hypercube design is constructed using a random procedure. This process results in many possible designs, each being equally good in terms of the Latin Hypercube conditions. However, a design that is ill suited for creating a meta model is possible, even if all the Latin Hypercube requirements are satisfied, as illustrated in Fig.1(b).



(a) Latin Hypercube 01 (b) Latin Hypercube 02

Figure 1. Latin Hypercube DoE for $M = 2$ and $N = 5$.

To overcome the above mentioned problem, the *Optimal Latin Hypercube* design was introduced to improve the space-filling property of the Latin Hypercube design. The Optimal Latin Hypercube design augments the Latin Hypercube design by requiring that the sample points be distributed as uniformly as possible throughout the design space. Unfortunately, the Optimal Latin Hypercube design results in a time consuming optimization problem. For example, to optimize the location of 10 samples in 4 dimensions (the 10×4 Latin Hypercube design) the optimizer has to select the best design from more than 10^{22} possible designs. If the number of design variables is increased to 5 (the 10×5 Latin Hypercube design), the number of possible designs are more than 6×10^{32}.

To solve the Optimal Latin Hypercube design, it is necessary to formulate an optimization problem that describes the best design. Audze and Eglais (1977) proposed a method that uses the potential energy of the sample points to generate a uniform distribution. Johnson et al. (1990) introduced a distance criterion as an objective function. Morris and Mitchell (1995) presented the ϕ_p criterion as an alternative to the distance criterion. Based on the previous criteria, several authors reported different strategies to solve the resulting optimization problem. Morris and Mitchell (1995) adapted a version of the simulated annealing algorithm. Ye et al. (2000) used a columnwise-pairwise algorithm. Bates et. al (2004) used a special implementation of the genetic algorithm. However, the computational cost of these algorithms is generally high (Bates et al., 2004; Jin et al. 2005). For example, Ye et al. (2000) reported that generating an 25×4 Optimal Latin Hypercube design using the columnwise-pairwise algorithm could take several hours on a Sun SPARC 20 workstation.

The question then arises how to obtain an optimal or nearly optimum Latin Hypercube design, while limiting the required computational resources? One possible solution is to solve the resulting optimization problem in an approximate sense, i. e., to obtain a good answer quickly, rather than finding the best possible solution. In this paper we used an algorithm that is not only able to quickly construct a good design of experiments, but also possess global properties, allowing it to move away from a locally optimal design (Jin et al. 2005).

2. OPTIMAL LATIN HYPERCUBE DESIGN OF EXPERIMENTS

The Optimal Latin Hypercube design problem is defined as searching for a design $\mathbf{X}^*$, which minimizes the objective function f:

$$\mathbf{X}^* = \min f(\mathbf{X}) \quad . \tag{1}$$

The objective function is formulated to achieve the required uniform space-filling property and, as a result, to avoid situations such as that illustrated in Fig. 1(b). In this paper, the ϕ_p criterion (Morris and Mitchell, 1995; Jin et al., 2005) was used for the objective function. From Eqs. 2 and 3 it is clear that this objective function results in the maximization of the point-to-point distance in the design (Johnson et al., 1990). A design is called a ϕ_p-optimal design, if it minimizes:

$$\phi_p = \left[\sum_{i=1}^{s} J_i d_i^{-p} \right]^{1/p} \quad , \tag{2}$$

where

- p is a positive integer (with a very large p, the ϕ_p criterion is equivalent to the maximin distance criterion (Jin et al., 2005),

- d_i's are distinct distance values with $d_1 < d_2 < \ldots < d_s$
- J_i is the number of pairs of points in the design separated by d_i, and
- s is the number of distinct distance values.

By sorting all the point-to-point distance $d_{ij} (1 \leq i, j \leq n, i \neq j)$, the distance list $(d_1, d_2, ..., d_s)$ and the index list $(J_1, J_2, ..., J_s)$ can be obtained. To close the ϕ_p definition, the inter-sited distance can be expressed as follows:

$$d(\mathbf{x_i}, \mathbf{x_j}) = d_{ij} = \left[\sum_{k=1}^{M} |x_{ik} - x_{jk}|^t \right]^{1/t} \quad , \quad t = 1 \text{ or } 2 \quad . \tag{3}$$

Up to this point, only the optimization problem was defined. However, in order to efficiently obtain an Optimal Latin Hypercube design, more than the problem definition is required. An appropriate optimization algorithm, with a cost efficient means of evaluating the objective function, are also important. To overcome the high computational cost associated with the existing approaches the method used in this work is supported by (a) an adaptation and an enhancement of a global search algorithm, i.e., the Enhanced Stochastic Evolutionary Algorithm, and (b) an efficient method for evaluating the objective function to reduce the computational cost.

3. ENHANCED STOCHASTIC EVOLUTIONARY ALGORITHM

The algorithm used to solve this problem is the *Enhanced Stochastic Evolutionary Algorithm* (ESEA), introduced by Jin et al. (2005). The ESEA is a modified version of the Stochastic Evolutionary Algorithm, developed by Saab and Rao (1991). The ESEA, as shown in Fig. 2, consists of an inner loop and an outer loop. The inner loop constructs new designs by an element-exchange approach and dictates whether to accept the designs based on a certain acceptance criterion dealing with the location of the points. A set of J Latin Hypercube designs is taken by exchanging two elements within the column $(i \, mod \, m)$ and then the best of this J designs is chosen to be compared with the current best solution. The outer loop controls the entire optimization process by adjusting the threshold value T_h in the acceptance criterion of the inner loop. According to Jin et al. (2005), this dynamic adjustment of the acceptance criterion enable the algorithm to avoid local minima designs. During this process, $\mathbf{X_{best}}$ is used to keep track of the best design.

3.1 Inner Loop

The inner loop is responsible for constructing new designs by using an element-exchange approach. The new designs can be accepted or rejected, based on an acceptance criterion that deals with the location of the points. As can be seen in Fig. 2(b), the inner loop has M iterations. At iteration i, a set of J Latin Hypercube designs is created by exchanging two elements within the column $(i \, mod \, m)$ of the current design, $\mathbf{X}$, with the best of these J designs, $\mathbf{X}_{try}$, selected for comparison with the current solution, $\mathbf{X}$. The substitution of $\mathbf{X}$ by $\mathbf{X}_{try}$ depends on the acceptance criterion given by:

$$\Delta f \leq T_h \times random\,(0,1) \quad , \tag{4}$$

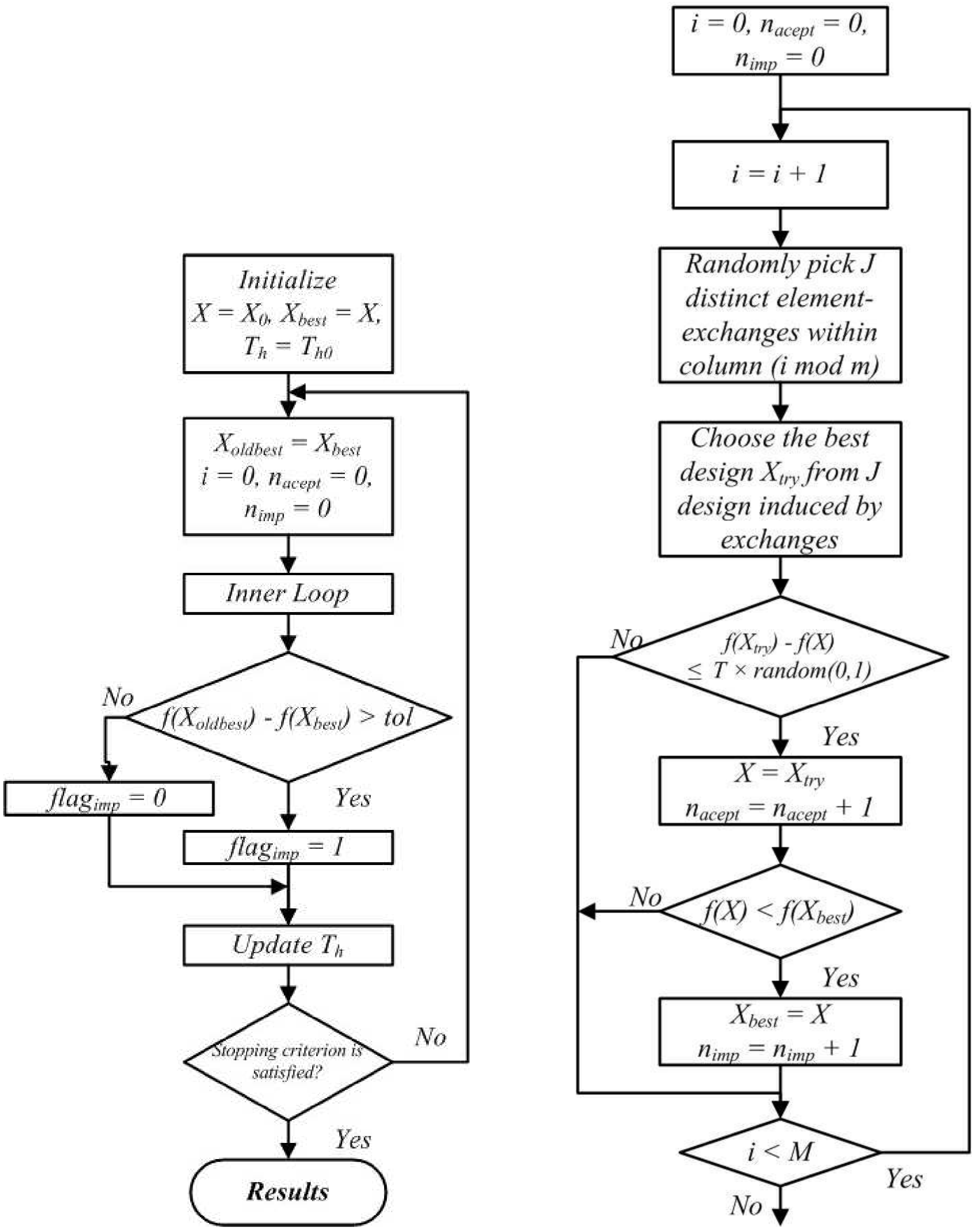
where:

- $\Delta f = f\,(\mathbf{X}_{try}) - f\,(\mathbf{X})$,
- $random\,(0,1)$ is a function that generates uniform random numbers between 0 and 1, and
- $T_h > 0$ is the threshold control parameter.

$\mathbf{X}_{try}$ will be accepted only if it satisfies $0 < \Delta f < T_h$ and the probability of acceptance given by:

$$P\,(S \geq \Delta f / T_h) = 1 - \frac{\Delta f}{T_h} \quad , \tag{5}$$

Using the scheme guided by Eqs. 4 and 5, the algorithm attempts to avoid local solutions. It can be noticed that a temporarily worse design, $\mathbf{X}_{try}$, could be accepted to replace the current design, $\mathbf{X}$. Table 1 gives the values for the inner loop parameters, according to Jin et al. (2005).


Initialize
X = X0, Xbest = X, Th = Th0
Xoldbest = Xbest
i = 0, nacept = 0, nimp = 0
Inner Loop
f(Xoldbest) - f(Xbest) > tol
No
Yes
flagimp = 0
flagimp = 1
Update Th
Stopping criterion is satisfied?
No
Yes
Results
i = 0, nacept = 0, nimp = 0
i = i + 1
Randomly pick J distinct element-exchanges within column (i mod m)
Choose the best design Xtry from J design induced by exchanges
f(Xtry) - f(X) ≤ T × random(0,1)
No
Yes
X = Xtry
nacept = nacept + 1
f(X) < f(Xbest)
No
Yes
Xbest = X
nimp = nimp + 1
i < M
Yes
No

(a) Outer Loop (b) Inner Loop

Figure 2. Basic scheme for ESEA.

Table 1. Inner loop parameters.

Parameter	Value	Comments
J	$n_e/5$, but no larger than 50.	n_e is the number of all possible distinct $k = 2$ element-exchanges in a column. In the Latin Hypercube case $n_e = \binom{N}{k} = \frac{n!}{k!(n-k)!}$.
M	$\dfrac{2 \times N \times M}{J}$, but no larger than 100.	M is the number of iterations in the inner loop.

3.2 Outer Loop

The outer loop controls the entire optimization process by adjusting the threshold value T_h in the acceptance criterion of the inner loop. Initially, T_h is taken as a small fraction of the initial design ($T_h = 0.005 \times \phi_p(\mathbf{X}_0)$) and its value is dynamically updated during the search process. The search process is divided in two main stages: (a) the *improving process*, which attempts to find a locally optimal design, and, (b) the *exploration process*, which try to escape from the locally optimal design. These two stages are discussed in more detail below.

Improving process : The search process switches to the improving process if the ϕ_p-criterion is improved after completing an entire inner loop. This mechanism is controlled by $flag_{imp}$, i.e. $flag_{imp} = 1$. Once in the improving process, T_h is adjusted to rapidly find a local optimal design. This is done keeping the value of T_h small, so that only a better, or a slightly worse, design is accepted to replace $\mathbf{X}$. T_h is updated based on the acceptance ratio n_{acpt}/M (number of accepted design divided by the number of tries in the inner loop) and the improvement ratio n_{imp}/M (number of improved design divided by the number of tries in the inner loop). Thus, there are the following possibilities:

1. T_h will decrease if the acceptance ratio is larger than a small percentage (e.g., 10%) and the improvement ratio is less than the acceptance ratio.
2. T_h will remain constant if the acceptance ratio is larger than the small percentage and the improvement ratio is equal to the acceptance ratio (meaning that T_h is so small that only improving designs are accepted by the acceptance criterion).
3. T_h will increase otherwise.

The following equations are used to decrease and increase T_h respectively:

$$T_h^{new} = \alpha_1 T_h^{old} \qquad , \tag{6}$$

$$T_h^{new} = \frac{T_h^{old}}{\alpha_1} \qquad , \tag{7}$$

where $0 < \alpha_1 < 1$. As in Jin et al. (2005), setting $\alpha_1 = 0.8$ worked well in all tests.

Exploration process : If no improvement is made after completing an entire inner loop, the search process will turn to the exploration process. This mechanism is controlled by $flag_{imp}$, i.e. $flag_{imp} = 0$. During the exploration process, T_h is adjusted to help the algorithm escape from a locally optimal design. Unlike the improving process, T_h fluctuates within a range based on the acceptance ratio. If the acceptance ratio is less than a small percentage (e.g., 10%), T_h will rapidly increase until the acceptance ratio is larger than a large percentage (e.g. 80%). If this happens, T_h will slowly decrease until the acceptance ratio is less than the small percentage. This process will be repeated until an improved design is found.

The following equations are used to decrease and increase T_h, respectively:

$$T_h^{new} = \alpha_2 T_h^{old} \qquad , \tag{8}$$

$$T_h^{new} = \frac{T_h^{old}}{\alpha_3} \qquad , \tag{9}$$

where $0 < \alpha_3 < \alpha_2 < 1$. As in Jin et al. (2005), $\alpha_2 = 0.8$ and $\alpha_3 = 0.7$ worked well in all tests.

As a result, during the exploration process, T_h increases rapidly (so that worse designs could be accepted) to help escape a local optimal design. And, T_h decreases slowly for finding better designs after moving away from the local optimal design.

For the whole algorithm, the maximum number of cycles is used as the stopping criterion.

4. EFFICIENT APPROACH FOR EVALUATING THE OPTIMALITY CRITERION

Since the objective function is evaluated whenever a new design of experiments is constructed, the efficiency of this evaluation becomes very important for creating the Optimal Latin Hypercube design within a reasonable time frame. Consider the evaluation of the ϕ_p based on Eq. 2. It can be seen that this process includes three parts, i.e.:

1. the evaluation of all the point-to-point distances,
2. the sorting of these distances to obtain a distance list and index list, and
3. the evaluation of the ϕ_p value.

However, it can be observed that after an exchange $(x_{i_1 k} \longleftrightarrow x_{i_2 k})$ only elements in rows i_1 and i_2 and columns i_1 and i_2 are changed in the $\mathbf{D}$ distance matrix. Thus, if Eq. 2 could be written to take advantage of this fact, an efficient way of calculation of ϕ_p is provided. It would avoid unnecessary calculations and the sorting required by Eq. 2. In this case, the new ϕ_p is computed by:

$$\phi'_p = \Bigg[\phi_p^p \quad + \sum_{1\le j\le n, j\ne i_1,i_2} \left(\left(d'_{i_1 j}\right)^{-p} - \left(d_{i_1 j}\right)^{-p} \right) \\ \sum_{1\le j\le n, j\ne i_1,i_2} \left(\left(d'_{i_1 j}\right)^{-p} - \left(d_{i_1 j}\right)^{-p} \right) \Bigg]^{1/p} . \tag{10}$$

where

- $d'_{i_1 j} = d'_{j i_1} = \left[\left(d_{i_1 j}\right)^t + s\left(i_1, i_2, k, j\right) \right]^{1/t}$,
- $d'_{i_2 j} = d'_{j i_2} = \left[\left(d_{i_2 j}\right)^t - s\left(i_1, i_2, k, j\right) \right]^{1/t}$, and
- $s\left(i_1, i_2, k, j\right) = \left|x_{i_2 k} - x_{jk}\right|^t - \left|x_{i_1 k} - x_{jk}\right|^t$.

Table 2 provides an indication of possible savings in computational time when using Eq. 10 to evaluate the objective function. In Table 2, T_{full} is the objective function calculated through Eq. 2 and $T_{enhanced}$ is the enhanced approach, using Eq. 10. Table 2 shows the wall-clock time.

Table 2. Time comparison between two ways of calculating the objective function (adopted from Jin et al. (2005)).

Latin Hypercube	12×4	25×4	50×5	100×10
$T_{enhanced}/T_{full}$	0.454545	0.192308	0.082645	0.032787

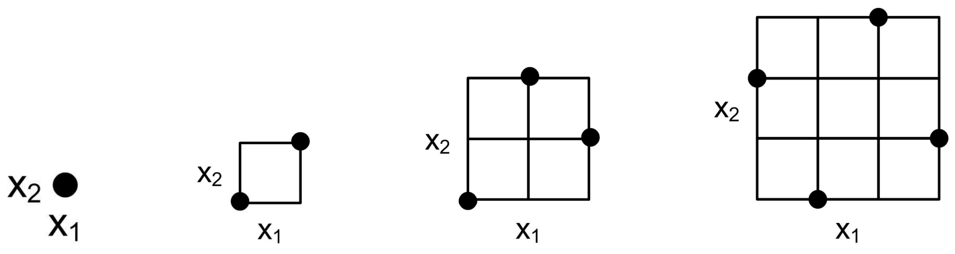
5. AN EMPIRICAL APPROACH TO CREATE STRUCTURED LATIN HYPERCUBE DESIGNS

This section describes an empirical approach to create a well structured design that is reasonably close to an Optimal Latin Hypercube design, without performing formal optimization. The importance of such an approach resides in the fact that it gives the capability to create a Latin Hypercube design that has better space filling properties than the standard Latin Hypercube design, using minimum computational time (at most, seconds). The resulting design can be used either as an initial design for the Optimal Latin Hypercube generator algorithm or as a good approximate Optimal Latin Hypercube design. In this case, one should take into account a compromise between obtaining best Optimal Latin Hypercube design and the computational cost required to generate that design.

The approach is quite simple and is based on the hope that the a simple N-dimensional Latin Hypercube that can be constructed from a N-dimensional seed design. Instead of a formal description of the approach, a practical example will be used to explain the methodology.

Consider the case where a 16×2 Optimal Latin Hypercube design is required, i.e., 16 points in 2-dimensions. First, a small Latin Hypercube will be selected for being used as a seed design in the process. Figure 3 shows some examples of 2-dimensional seed designs. Figure 3(a) shows the seed used in this example. It is important to notice that this seed can be as simple as just a $1 \times N$ design (where N is the number of dimensions of the problem, i.e. number of desing variables).

Second, the design space is divided into blocks, in such a way that each dimension is divided in the same number of blocks. The result is that each block can be filled using the seed design (defined previously). It is clear that these processes



(a) 1×2 seed design. (b) 2×2 seed design. (c) 3×2 seed design. (d) 4×2 seed design.

Figure 3. Examples of seed design for 2-dimensions.

are inter-dependent. The seed size, i. e., the number of points in the seed design, and the final design size will determine the number of blocks in each dimension. In general, the following relations must be observed:

$$LatinHypercubeSize = NumberOfBlocks \times SeedSize \quad , \tag{11}$$

$$NumberOfBlocks = (NumberOfDivisions)^{NumberOfDimensions} \quad , \tag{12}$$

$$SeedSize = \frac{DesignSize}{NumberOfBlocks} \quad . \tag{13}$$

Using our seed design, Fig. 4 shows how the 16×2 Latin Hypercube mesh will be divided into blocks. It is important to point out that the fact of each block has four rows and four columns of the Latin Hypercube mesh does not mean that each block will have four points at the end of the process. Instead of that, this is a way to ensure the minimal distance between point on the final Latin Hypercube.

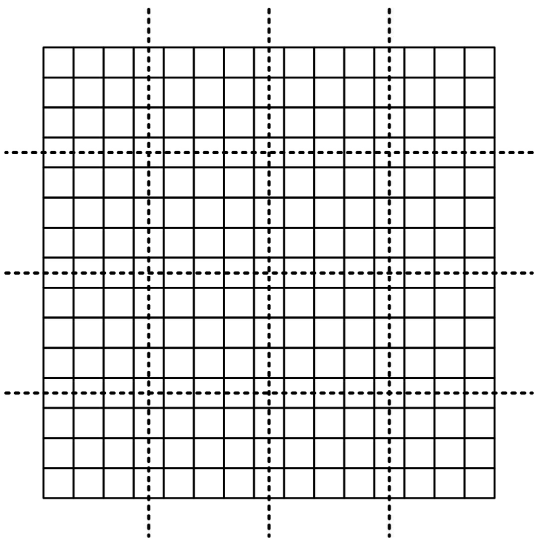


Figure 4. 16×2 Latin Hypercube divided mesh.

The seed design must be properly placed into each of the blocks. Figure 5 illustrates how this process is applied. The first step is to properly scale the “seed design” and then placing it at the origin. Next, a set of “shiftings” must be performed. The first one is to shift the seed to consecutive blocks following one of the dimensions. The second one is to shift the origin of the seed inside the mesh of the block. There is a coupling between these two process. If the block shifting is performed on the rows, the seed-origin shift must be performed on the columns, and vice-versa. This process is repeated until to fill one of the dimensions. After that, the whole set of points placed in that dimension can be used to feedback the “shifting” process that continues filling the next dimensions.

The biggest advantage of this approach is that there are no calculations to perform. All operations can be viewed as translations of an N-points block in an N-dimensional hypercube. Preliminary studies have been conducted that show promising results in low-dimensional cases.

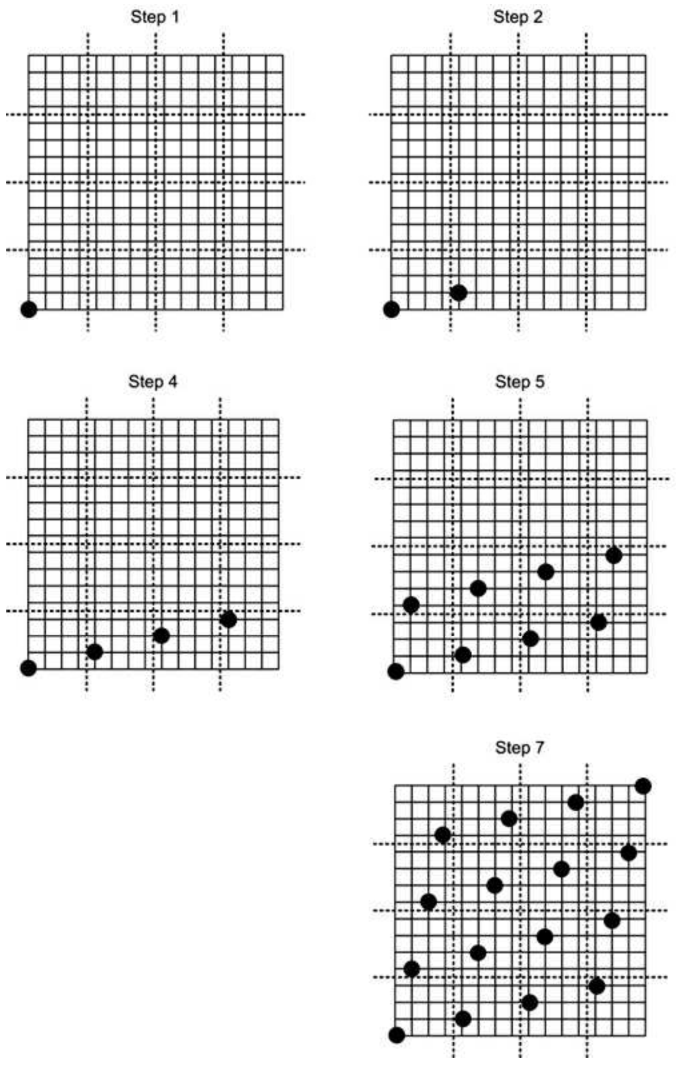
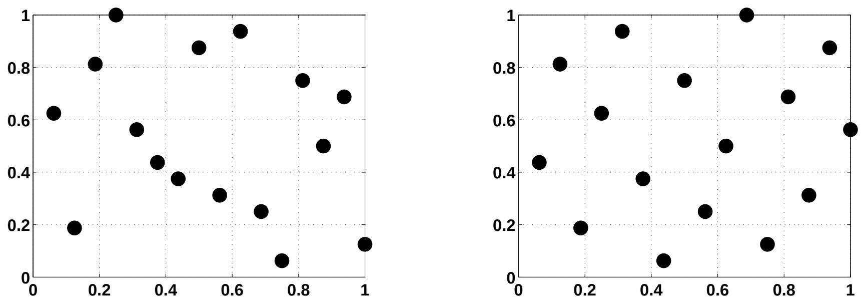


Figure 5. 16×2 Structured Latin Hypercube creation process.

6. NUMERICAL RESULTS

6.1 Optimal Latin Hypercube

As an illustration of how the entire technique works, Fig. 6 shows both the initial Latin Hypercube, i.e. before optimization, and the final Optimal Latin Hypercube, which is obtained by solving the optimization problem. The initial Latin Hypercube is a random design with good one-dimensional projective properties (in other words, there is only one point for each level), but with a poor space-filling property. This is typically for Latin Hypercube designs. In contrast, the Optimal Latin Hypercube design maintains the projective property while providing an excellent space filling property.



(a) Initial Latin Hypercube (b) Final Optimal Latin Hypercube

Figure 6. 16×2 Latin Hypercube before and after optimization.

Tab. 3 gives the wall-clock time for creating a set of OLH. These results were obtained using a PC with a 1000 Mhz Pentium III Zeon processor, running Linux.

Table 3. Time consumption for several Optimal Latin Hypercube.

Design	**Time [s]**
10×2	< 1
50×2	5
100×2	23
10×5	< 1
50×5	6
100×25	38
200×25	324
100×50	45
200×50	387
200×100	395

At this point, the effectiveness of the solution when using the present approach for generating the Optimal Latin Hypercube design can be illustrated. Table 4 shows a comparison between three different strategies. The two strategies that use Genetic Algorithms are described by Bates et al. by Bates et al. (2004). The two Genetic Algorithms make use of the potential U-criterion for the objective function instead of the ϕ_p-criterion used in the present work. This U-criterion is analogous to the potential energy of the system of material points and can be expressed by:

$$U = \sum_{p=1}^{N} \sum_{q=p+1}^{N} \frac{1}{d_{pq}^2} \qquad , \tag{14}$$

where d_{pq} is the inter-distance between the points p and q of the design.

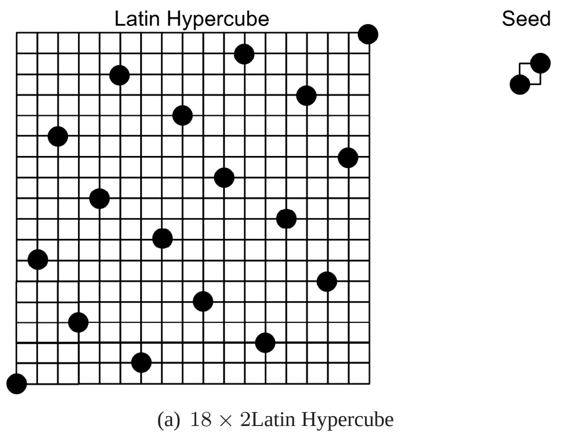
Comparing both the number of function evaluations and the value of U-criterion, it is easy to see that the current approach is a compromise between computational cost and the best final design. The advantage is clear specially when comparing the number of function evaluations for large designs. In general, the present technique quickly generates an answer that has good space-filling properties, but is not necessarily the “optimum” design.

6.2 Structured Latin Hypercube Designs

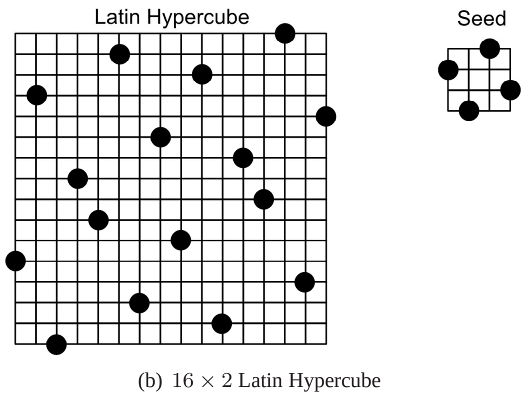
Figure 7 illustrates two examples where the proposed empirical approach was applied.

Table 4. Number of function evaluations required for different Optimal Latin Hypercube generators. The values within parenthesis represent the potential energy U-criterion.

Design	**Binary Genetic Algorithm (data from Bates et al. (2004))**	**Permutation Genetic Algorithm (data from Bates et al. (2004)**	**Enhanced Stochastic Evolutionary Algorithm**
5×2	60 (1.2982)	50 (1.2982)	$2,040$ (1.2982)
10×2	$39,240$ (2.0662)	$1,860$ (2.0662)	$5,085$ (2.1393)
120×2	$22,003,500$ (5.7733)	$130,570$ (5.5174)	$114,000$ (5.7542)
5×3	$5,260$ (0.7267)	$1,922$ (0.7267)	$3,060$ (0.7361)
10×3	$165,980$ (1.0401)	$38,950$ (1.0242)	$4,950$ (1.0359)
120×3	$5,908,540$ (2.0541)	$1,996,920$ (1.9613)	$184,800$ (2.0309)
50×5	$280,000,000$ (0.7348)	$1,996,840$ (0.7270)	$143,000$ (0.7670)
120×5	$59,802,200$ (0.8003)	$1,998,540$ (0.7930)	$475,200$ (0.8167)



(a) 18×2Latin Hypercube



(b) 16×2 Latin Hypercube

Figure 7. Examples of structured Latin Hypercube designs.

The appeal of this approach is that virtually no computational time is required to create the designs. This empirical approach can be used either to obtain a quick answer or to generate a good starting point for a formal Optimal Latin Hypercube optimization. Table 5 shows the performance comparison of the Optimal Latin Hypercube generator starting from three different initial guesses. For all cases, the stopping criterion used was a maximum number of 100 iterations. The three initial guesses used were: (a) the worst (diagonal) initial design (as shown in Fig. 1(b)), (b) an empirical design, and (b) a random initial design.

Table 5. Optimal Latin Hypercube generator with three different initial guesses.

Design size	**Quantity**	**Worst case**	**Structured case**	**Random case**
225×2	Time [s]	143	68	147
	Iterations	251	101	237
	ϕ_p initial	0.55715	0.07052	0.51110
	ϕ_p final	0.07560	0.07052	0.07528
1024×2	Time [s]	11155	3868	7838
	Iterations	284	101	202
	ϕ_p initial	0.57433	0.03527	0.50697
	ϕ_p final	0.04218	0.03527	0.04258
256×4	Time [s]	313	372	469
	Iterations	283	336	424
	ϕ_p initial	0.27929	0.01658	0.03846
	ϕ_p final	0.01101	0.01087	0.01082
243×5	Time [s]	609	556	550
	Iterations	547	498	494
	ϕ_p initial	0.22320	0.01303	0.02668
	ϕ_p final	0.00686	0.00688	0.00679
1024×10	Time [s]	34283	27772	29421
	Iterations	601	489	518
	ϕ_p initial	0.22973	0.00440	0.01086
	ϕ_p final	0.00233	0.00234	0.00232

Note that for the 2D cases, the formal optimization could not improve the ϕ_p-criterion obtained by the empirical approach. This indicates that the for the 2D cases, the empirical approach produced the optimum results at no computation cost. For the higher dimensional cases, the formal optimization were able to make only small improvements to the ϕ_p-values obtained from the empirical approach. This is a sign that the empirical design is, at least near to a local minima in higher dimensions.

7. CONCLUSIONS

In this paper, an efficient and affordable algorithm for constructing the Optimal Latin Hypercube Design of Experiments was introduced. The complete approach includes two major elements: (a) the use of the Enhanced Stochastic Evolutionary Algorithm for performing the search process, and (b) the employment of an efficient method for evaluating the optimality criteria (ϕ_p).

An empirical approach to create structured Latin Hypercube designs was also presented. This approach is based on the idea that a simple “seed design”, with few points, can be used to build a complete design. The main advantage of the technique is that it produces a design requiring virtually no computational cost. Test cases in 2D show that the approach produces designs that could not be improved using a formal optimization approach. In higher dimensions, the formal optimization approach could make small improvements to the designs obtained from the empirical approach. Future work will include both the discussion about the influence of the “seed” on the final design and enhancements on the algorithm in order to work better in higher dimensions.

8. REFERENCES

S. Bates, J. Sienz, and V. Toropov. Formulation of the Optimal Latin Hypercube Design of Experiments Using a Permutation Genetic Algorithm. In *Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Palm Springs, USA*, AIAA-2004-2011, April 19-22 2004.

R. Jin, W. Chena, and A. Sudjianto. An Efficient Algorithm for Constructing Optimal Design of Computer Experiments. *Journal of Statistical Planning and Inference*, 134(1):268–287, September 2005.

L. Gu. A Comparison of Polynomial based Regression Models in Vehicle Safety Analysis. In *ASME Design Engineering Technical Conferences and Design Automation Conference, Pittsburgh, USA*, 2001.

T. W. Simpson, A. J. Booker, D. Ghosh, A. A. Giunta, P. N. Koch, and R. J. Yang. Approximation Methods in Multidisciplinary Analysis and Optimization: a Panel Discussion. *Structural and Multidisciplinary Optimization*, 27(5):302–313, 2004.

M.D. McKay, R.J. Beckman, and W.J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables from a Computer Code. *Technometrics*, 21:239–245, 1979.

R. L. Iman and W. J. Conover. Small Sample Sensitivity Analysis Techniques for Computer Models, with an Application to Risk Assessment. *Communications in Statistics, Part A. Theory and Methods*, 17:1749–1842, 1980.

P. Audze and V. Eglais. New Approach for Planning out of Experiments. *Problems of Dynamics and Strengths*, 35:104–107, 1977.

M. Johnson, L. Moore, and D. Ylvisaker. Minimax and Maximin Distance Designs. *Journal of Statistics Planning and Inference*, 26:131–148, 1990.

M. D. Morris and T. J. Mitchell. Exploratory Designs for Computational Experiments. *Journal of Statistics Planning and Inference*, 43:381–402, 1995.

K. Q. Ye, W. Li, and A. Sudjianto. Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs. *Journal of Statistical Planning and Inference*, 90:145–159, 2000.

Y. G. Saab and Y. B. Rao. Combinatorial Optimization by Stochastic Evolution. *IEEE Transactions on Computer-Aided Design*, 10:525–535, 1991.

9. Responsibility notice

The authors are the only responsible for the printed material included in this paper.