ON HOW TO IMPLEMENT AN AFFORDABLE OPTIMAL LATIN HYPERCUBE

Felipe Antonio Chegury Viana* Gerhard Venter†
Vladimir Balabanov‡ Valder Steffen, Jr§

Abstract

In general, the choice of the location of the evaluation points is important in the process of response surface generation. In this scenario, Design of Experiments is used to help the task of point location through the design space. One popular technique is the so called Latin Hypercube. This is a fast-to-generate design, however, due the random nature of the generation process, it can present some disadvantages, such as, the possibility of a bad design in terms of fitting a meta model. To overcome this difficulty, one can use the Optimal Latin Hypercube. The big deal with this design is the computational cost associated with its generation. Therefore, solving this problem requires an optimization technique for search the desing space. This paper describes a method for generating the optimal latin hypercube design of experiments. Following, two key-points to speed up the process are presented. Finally, numerical results are reported, illustrating the success of using the methodology presented in generation of the Optimal Latin Hypercube.

1 Introduction

Approximation methods have been widely used in engineering design for a long time [1, 2]. Just as an example, the Ford Motor Company reports that a crash simulation in a full model for a car can spend from 36 to 160 hours [3]. Following this example, a large number of the complex problems in Optimization uses approximation methods, such as Design of Experiments combined with Response Surface Methodology, for approximating objective or constraint functions. The metamodel is used in the place of the original code to facilitate the optimization task, design space exploration and reliability analysis [4]. Bates et al. in [1] justified the use of this approach based on two needs:

1. to minimize the number of responses evaluations, and

*PhD student, Federal University of Uberlândia, School of Mechanical Engineering / fchegury@mecanica.ufu.br.
†VisualDOC Project Manager, Vanderplaats Research and Development, Inc. / gventer@vrand.com.
‡Senior Research and Development Engineer, Vanderplaats Research and Development, Inc. / vladimir@vrand.com.
§Professor, Federal University of Uberlândia, School of Mechanical Engineering / vsteffen@mecanica.ufu.br.

2. to reduce the effect of numerical noise.

Several commonly used approaches are available to achieve these goals e.g., response surface and Kriging approximations. Both these approaches require the evaluation responses at a number of design points in the design space for constructing the resulting approximations. Typically, Design of Experiments is used to identify the design points at which to evaluate the response values. Each point in the design space is defined by a specific combination of the input parameters (design variables). The evaluations of responses may constitute physical experiments or computer evaluations. Once the response values are known at the specified points, an approximation model can be 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 design of experiments. One such criterion is a space filling design that aims at covering as much of the design space as possible and as evenly as possible. One well known space filling design is the Latin Hypercube sampling method , proposed by McKay et al. in [5] and Iman and Conover in [6].

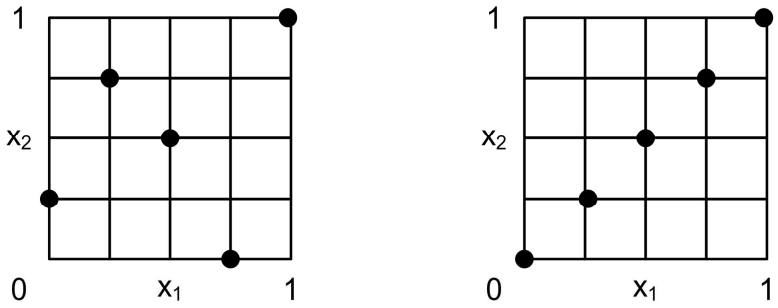
There are several advantages to using the Latin Hypercube design:

- the number of samples (points) is not fixed,
- orthogonality of the sampling points, and
- 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) [1] for each level. The final Latin Hypercube design then has N samples. Figure 1 shows two possible Latin Hypercube designs for $M = 2$ and $N = 5$. The following remarks should be made about these designs:

- The Latin Hypercube design is constructed using random points so that there are many possible answers, depending on the random number generator. In addition, any one of these designs is as good as the other in terms of the Latin Hypercube conditions.
- The design can be scaled to fit any range of the design variables.
- A bad design in terms of fitting a meta model to it is possible, even if all the Latin Hypercube requirements are satisfied, as illustrated in Fig. 1(b).

To overcome the above mentioned problem, the so called *Optimal Latin Hypercube* was introduced to improve the space-filling property. The Optimal Latin Hypercube design augments the Latin Hypercube design by requiring that the sample points be distributed as uniformly as possible through out the design space. However, the task of obtaining an Optimal Latin Hypercube design is difficult. For example, to optimize the location of 10 samples in 4 dimensions (the 10×4 Latin Hypercube) the number of distinct designs to consider is more than 10^{22}. If the number of design variables is increased to 5 for optimizing the 10×5 Latin Hypercube, the number of distinct designs is as large as 6×10^{32}.



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

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

One way to solve this problem is to define the generation of the Optimal Latin Hypercube design as an optimization problem. Audze and Eglais in [7] propose a method that uses the potential energy of the sample points to generate a uniform distribution of the points. Johnson et al. in [8] introduce a distance criterion as an objective function to be used in the optimization problem. Morris and Mitchell in [9] presented the ϕ_p criterion as an alternative to the distance criterion. Based on the previous criteria, several authors report different strategies to solve the final optimization problem. Morris and Mitchell in [9] adapted a version of the simulated annealing algorithm. Ye et al. in [10] used a columnwise-pairwise algorithm. Bates et. al in [1] used a special implementation of the genetic algorithm. However, the computational cost of optimization performed with these existing algorithms is generally high [1, 2]. For example, Ye et al. in [10] reported that generating an optimal 25×4 Optimal Latin Hypercube 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, without spending too much computational resources? One possible answer is that it is more practical to solve the Optimum Lathin Hypercube design-related optimization problem approximately, i. e., to obtain an answer that is good enough in terms of space-filling but maybe it is not the "optimum" design. 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 [2].

2 Optimal Latin Hypercube Design of Experiments

The Optimal Latin Hypercube design problem is defined as the searching of a design $\mathbf{X}^*$, which minimizes a given optimality criterion f:

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

The optimality criterion is used to achieve the space-filling property, and as a consequence, to avoid situations such as that illustrated in Fig. 1(b). In this paper, it was used the ϕ_p criterion [9, 2]. As can be seen through Eqs. 2 and 3, this optimality criterion is related to the maximization of the point-to-point distance of the desing [8]. 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 [2]),
- 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:

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

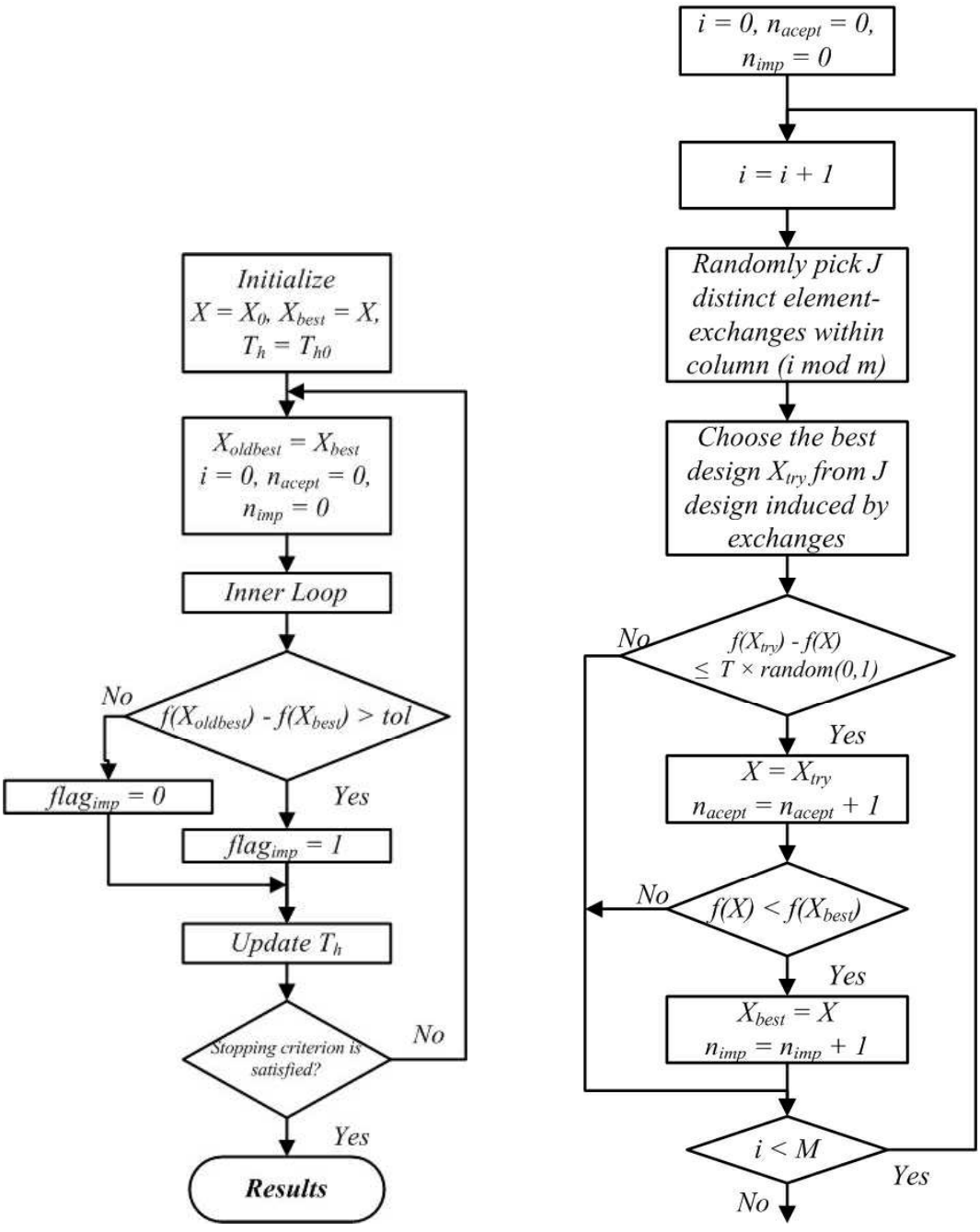
Up to this point, only the optimization problem was defined. However, in order to efficiently obtain an Optimal Latin Hypercube design, more than the problem definition is required. The appropriate optimization algorithm along with an enhanced way of evaluating the optimality criterion are important. To overcome the difficulties associated with the existing approaches the methods 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 optimality criterion 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 in [2] as an enhanced version of the Stochastic Evolutionary Algorithm, developed by Saab and Rao [11]. 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 [2], this dynamic adjust of the acceptance criterion enable the algorithm to avoid local minima designs. In the entire process, $\mathbf{X_{best}}$ is used to keep track of the updated best design.

3.1 Inner Loop

The inner loop is responsable for the construction of new designs by using an element-exchange approach. These new designs can be accept or not, depending on a acceptance criterion that deals with the location of the points. As can be seen in Fig. 2(b), the innner loop has M iterations. At iteration i, a set of J Latin Hypercube designs is taken by exchanging two elements within the column $(i \, mod \, m)$ of the current design, $\mathbf{X}$, and



(a) Outer Loop (b) Inner Loop

Figure 2: Basic scheme for ESEA.

then the best of this J designs, $\mathbf{X}_{try}$, is chosen to be compared with the current solution, $\mathbf{X}$. The substitution of $\mathbf{X}$ by $\mathbf{X}_{try}$ will depend on a acceptance criterion given by:

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

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 contrl parameter.

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

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

Using the scheme guided by Eqs. 4 and 5, the algorithm intends 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 gives the values for the inner loop parameters, according to what is suggested by Jin et al. in [2].

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	$\frac{2 \times N \times M}{J}$, 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 dinamically updated during the searching process. The search process is devided in two main stages: (a) the *improving process*, which atempts to find a locally optimal design, and, (b) the *exploration process*, which try to scape from the locally optimal design.

Taking a deeper look in these two stages:

Improving process : A search process is turned to the improving process if the ϕ_p-criterion is improved after to run an entire inner loop. This mechanism is controlled by $flag_{imp}$, i.e. $flag_{imp} = 1$. Once turning to the improving process, T_h is adjusted to rapidly find a locally optimal design. This is done keeping the value of T_h small,

so that only a better design or a slightly worse design would be 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 just 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 maintain in the current value 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:

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

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

where $0 < \alpha_1 < 1$. As in [2], the setting of $\alpha_1 = 0.8$ worked well in all tests.

Exploration process : If no improvement is made after to run an entire inner loop, the search process will be turned 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. Differently from what occurs during the improving process, here, 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 be slowly decreased 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:

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

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

where $0 < \alpha_3 < \alpha_2 < 1$. As in [2], the setting of $\alpha_2 = 0.8$ and $\alpha_3 = 0.7$ worked well in all tests.

This way, during the exploration process, T_h increases rapidly (so that more worse designs could be accepted) to help moving away from a locally optimal design. And, T_h decreases slowly for searching 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 optimality criterion is evaluated whenever a new design of experiments is constructed, the efficiency of this evaluation becomes very important for optimizing the design of experiment 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 those inter distances to obtain a distance list and index list, and
3. the evaluation of the ϕ_p

However, it can be observed that after an exchange ($x_{i_1k} \longleftrightarrow x_{i_2k}$) 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 in such a way to take advantage of this fact, the new way of calculation of ϕ_p is provided. It would be avoid double unnecessary calculations and the sorting required by Eq. 2. So, the new ϕ_p is computed by:

$$\phi_p' = \left[\phi_p^p + \sum_{1\leq j\leq n, j\neq i_1,i_2} \left(\left(d'_{i_1j}\right)^{-p} - \left(d_{i_1j}\right)^{-p}\right) \sum_{1\leq j\leq n, j\neq i_1,i_2} \left(\left(d'_{i_1j}\right)^{-p} - \left(d_{i_1j}\right)^{-p}\right)\right]^{1/p}. \tag{10}$$

where

- $d'_{i_1j} = d'_{ji_1} = \left[\left(d_{i_1j}\right)^t + s\left(i_1, i_2, k, j\right)\right]^{1/t}$,
- $d'_{i_2j} = d'_{ji_2} = \left[\left(d_{i_2j}\right)^t - s\left(i_1, i_2, k, j\right)\right]^{1/t}$, and
- $s\left(i_1, i_2, k, j\right) = \left|x_{i_2k} - x_{jk}\right|^t - \left|x_{i_1k} - x_{jk}\right|^t$.

Tab. 2 provides an idea how the efficiency during the evaluation of the objective function can save time in the task of searching for the Optimal Latin Hypercube (T_{full} is the objective function calculated through Eq. 2 and $T_{enhanced}$ is the enhanced approach, i.e. Eq. 10). Tab. 2 shows the wall-clock time used to calculate the objective function using the exact and approximate approaches.

Table 2: Time comparison between two ways of calculating the objective function (adopted from [2]).

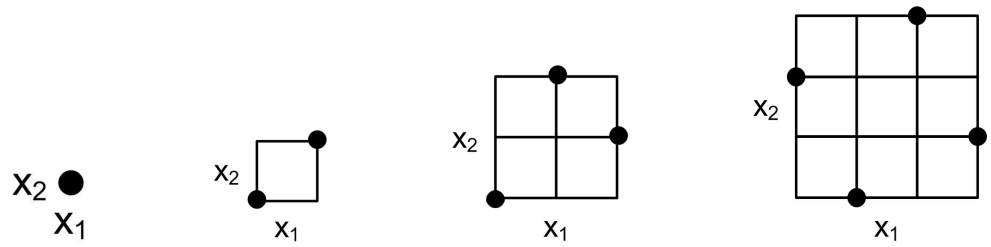
Latin Hypercube	$T_{enhanced}/T_{full}$
12×4	0.454545
25×4	0.192308
50×5	0.082645
100×10	0.032787

5 An Empirical Approach to Create Structured Latin Hypercube Designs

This section describes an empirical approach to create a well structured design (rather than based on random number generation) that is reasonably close to Optimal Latin Hypercube design without performing optimization. The importance of such approach resides in the fact that it gives the capability to create a Latin Hypercube design that is better than just randomly generated Latin Hypercube design using minimum computational time (at most, seconds). Further on, this design can be used either as an initial guess for the Optimal Latin Hypercube generator algorithm or as the final design. In this case, one should take into account a compromise somultion between the final design and the computational cost for generating the Optimal Latin Hypercube.

The approach is quite simple and it is based on the hope that the most simple Latin Hypercube that can be constructed for a specific N-dimensional problem can be used as a seed for a complete design. Instead of a formal description of the approach, it will be used a practical example to explain how to build a structured design from a basic one.

Consider the case in which is desired an 16×2 Latin Hypercube, i.e., 16 points in 2-dimensions. First of all, it is chosen an Latin Hypercube that will be used as a seed for the final design. Fig. 3 shows the 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).



(a) 1×2 seed design.

(b) 2×2 seed design.

(c) 3×2 seed design.

(d) 4×2 seed design.

Figure 3: Exemples of seed design for 2-dimensions.

Secondly, the Latin Hypercube is divided in blocks in such a way that each dimension is divided in the same number of divisions. Thus, each block can be filled using the seed design (defined previously). It is clear that all this process is inter-dependent. So, the seed size, i. e., the number of points of the seed design, and the final design size will determine the number of divisions and as a consequence the number of resulting blocks. In general, the following relations must be observed:

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

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

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

Following this ideas, Fig. 4 shows how the 16×2 Latin Hypercube mesh will be divided.

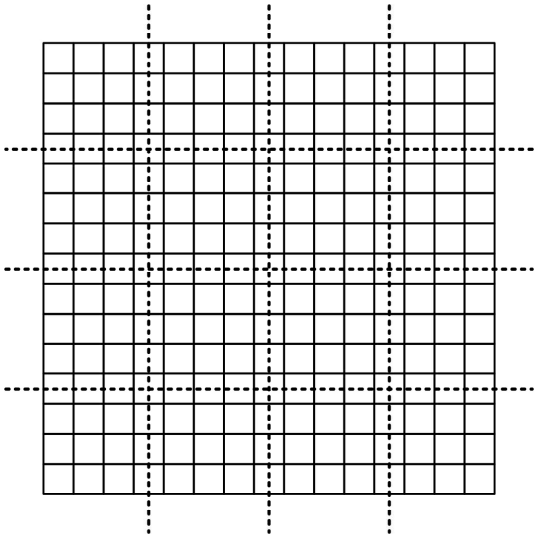


Figure 4: 16×2 Latin Hypercube divided mesh.

Finally, it should be done the proper placement of the seed into each of the blocks of the divided mesh. This can be done starting by properly scaling the "seed design" and then placing it at the origin. After that, 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 relating this two process. If the block shifting is performing on the rows, the seed-origin shift must be performed on the columns, and vice-versa. Figure 5 illustrates how this process is applied.

The greater advantage of this approach is that there is no calculation to be performed. All operations can be viewed as translations of a N-points block in a N-dimensional hypercube. Preliminary studies have been conducted that show promising results in 2 and 3-dimenisonal cases.

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, resultant from the optimization. The initial Latin Hypercube is a random design with good one-dimensional projective property (in other words, there is only one point for each level), but not so good space-filling property, on the other hand, in the Optimal Latin Hypercube the projective property is maintained while the space filling property is improved.

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.

At this point, it can be illustrated what is the comprimising solution when adopting the present approach for Optimal Latin Hypercube generation. Table 4 shows a comparison among three different strategies. Both strategies that uses Genetic Algorithms are described in [1]. Basically, they differ from the present method since they are based on

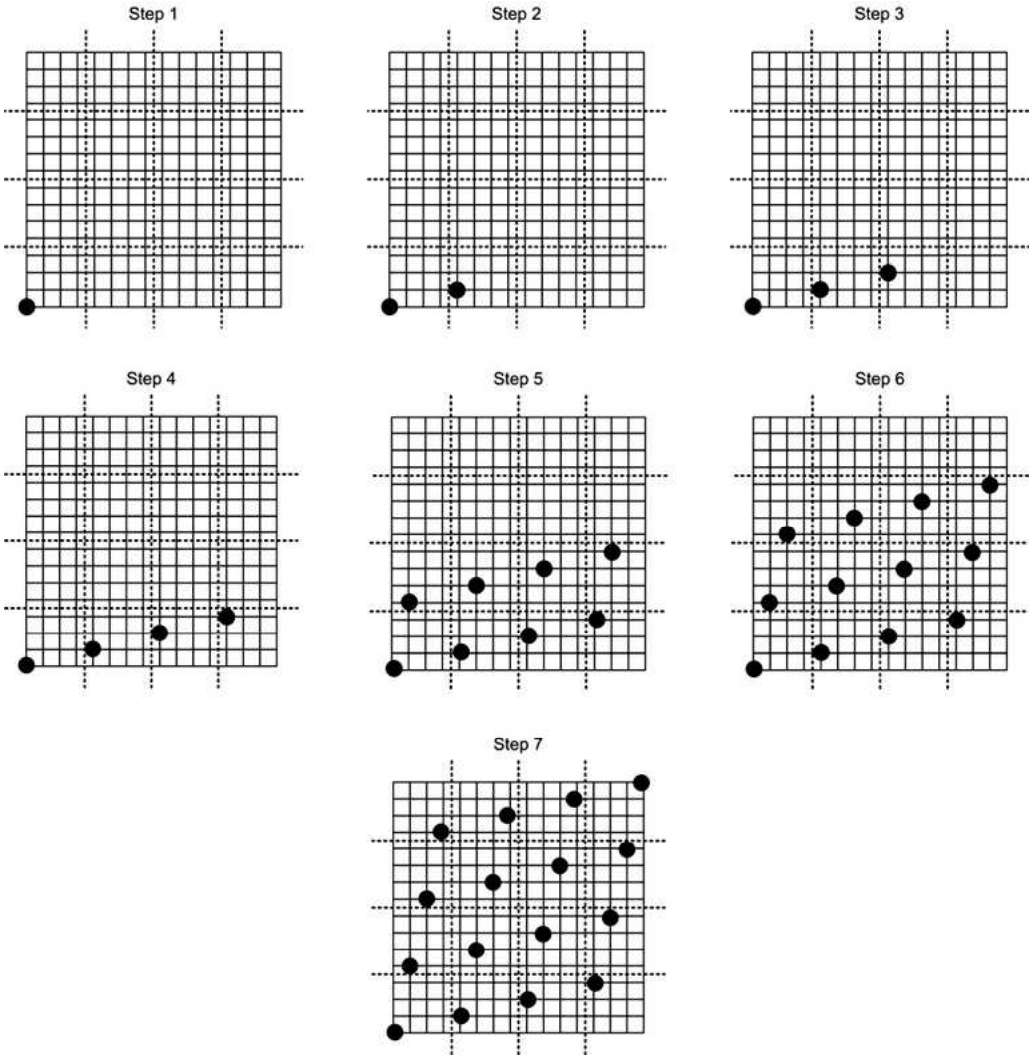
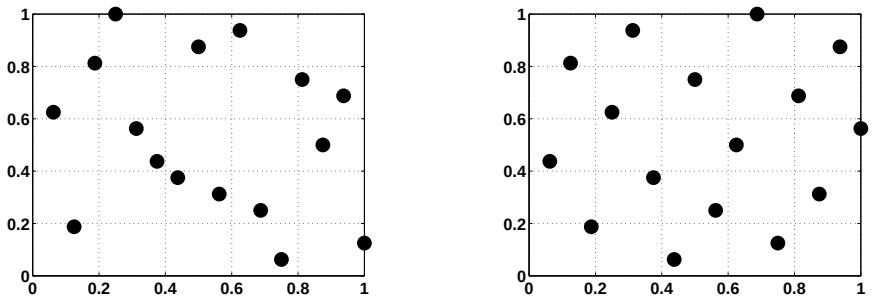


Figure 5: 16×2 Structured Latin Hypercube creation process.



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

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

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

two different codifications for Genetic Algorithm and instead of using the ϕ_p-criterion, they use the potential energy U-criterion. This criterion is analogous to the potential energy of the system of material points and can be expressed by:

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

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

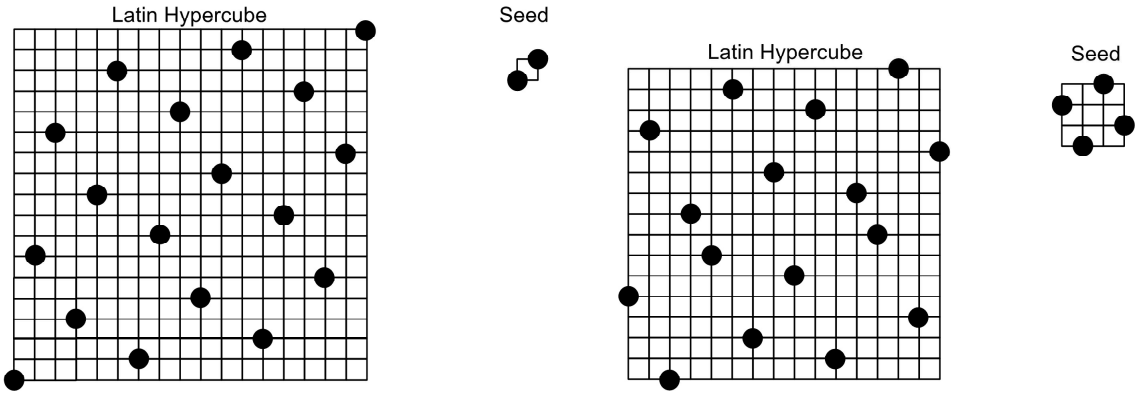
Comparing both the number of function evaluations and the value for U-criterion, it is easy to see that the current approach is a compromise between computational cost and the final design. The advantage is clear specially when comparing the number of function evaluations for large designs. In general,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

Table 4: Nuber 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 [1])**		**Permutation Genetic Algorithm (data from [1]**		**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)	1,996840	(0.7270)	143,000	(0.7670)
120×5	59,802,200	(0.8003)	1,998,540	(0.7930)	475,200	(0.8167)

6.2 Structured Latin Hypercube Designs

Figure 7 illustrates two more examples in which the present methodology was applied.



(a) 18×2Latin Hypercube (b) 16×2 Latin Hypercube

Figure 7: Examples of structured Latin Hypercube designs.

One of the appeals of this approach is to save time spent by the Optimal Latin Hypercube generator algorithm. When using this structured Latin Hypercube design, one can substitute the initial guess for the Optimal Latin Hypercube generator algorithm or even the final design. To have an idea about how efficient can be the proposed method, Tab. 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 the maximum number of 100 iterations. The three initial guesses used were: (a) a structured design, (b) a random initial design, and (c) the worst (diagonal) initial design (as shown in Fig. 1(b)).

At this point, it is important to notice that:

1. For all cases the structured design presents their initial and final values for the ϕ_p-criterion very close one to the other.

2. Considering that all of these cases stopped by the number of iterations without improvements, the structured design allows a faster convergence of the optimization task.

This is a sign that the structured design is, at least, near to a local minima. Considering the computational time for the optimization of the initial guess and the final value of the ϕ_p-criterion for all cases, one using the structured design can decide to skip the optimization step as a compromising solution between the time and a good design.

7 Conclusions

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

It was also described an empirical approach to create structured Latin Hypercube designs. This approach is based on the hope that a simple "seed design", with few points, can be managed to build a complete design. The technique has as main advantage the fact that it does not need either any type of optimality criterion evaluation or search for

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

the solution. All the building process is based only on scaling and translations of the "seed design".

References

[1] 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.

[2] R. Jinb, 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.

[3] 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.

[4] 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.

[5] 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*, 21:239245, 1979.

[6] 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.

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

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

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

[10] K. Q. Ye, W. Li, and A. Sudjianto. Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs. *Journal of Statistical Planning and Inference*, 90:145159, 2000.

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