Transactions of the ASME, Journal of Mechanical Design, Vol. 125, pp. 210-220, June 2003.

# Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points

*G. Gary Wang, Assistant Professor*
*Dept. of Mech. and Indus. Engineering*
*University of Manitoba*
*Winnipeg, MB, Canada R3T 5V6*
*Tel: 204-4749463 Fax: 204-2757507*
*Email: gary_wang@umanitoba.ca*

## *Abstract*

This paper addresses the difficulty of the previously developed Adaptive Response Surface Method (ARSM) for high-dimensional design problems. The ARSM was developed to search for the global design optimum for computation-intensive design problems. This method utilizes Central Composite Design (CCD), which results in an exponentially increasing number of required design experiments. In addition, the ARSM generates a complete new set of CCD samples in a gradually reduced design space. These two factors greatly undermine the efficiency of the ARSM. In this work, Latin Hypercube Design (LHD) is utilized to generate saturated design experiments. Because of the use of LHD, historical design experiments can be inherited in later iterations. As a result, ARSM only requires a limited number of design experiments even for high-dimensional design problems. The improved ARSM is tested using a group of standard test problems and then applied to an engineering design problem. In both testing and design application, significant improvement in the efficiency of ARSM is realized. The improved ARSM demonstrates strong potential to be a practical global optimization tool for computation-intensive design problems. Inheriting LHD samples, as a general sampling strategy, can be integrated into other approximation-based design optimization methodologies.

***Keywords:*** Response Surface Method, Latin Hypercube Design, Computation-intensive Design, Design Optimization, Global Optimization

### *Introduction*

As today's engineering design problems become more complex, many sophisticated analysis and simulation software tools are used for design study. These analysis and simulation processes are usually computation-intensive, which take considerable time and bring high computation cost to the product design. To optimize a design, conventional direct search or gradient-based optimization methods iterate those analysis and simulation processes sequentially to search for the optimum. In contrast, Response Surface Method (RSM) plans a group of design alternatives and performs the design analysis and simulation simultaneously on these design alternatives. Then an approximation model, called a response surface, is constructed. Consequently, design optimization does not involve those computation-intensive processes but rather an explicitly formulated and usually simple response surface. The total computation cost can therefore be reduced. RSM also has other engineering advantages such as supporting parallel computation, accommodating both continuous and integer design variables, and estimating parameter sensitivities [1].

### *Literature Review*

With the aim to reduce the computation cost for computation-intensive design problems, current RSM related research is roughly along three directions. The first direction is on the identification and development of a group of design alternatives for approximation, or experimental designs, that can provide maximum information with minimum number of design experiments. Among various groups of developed experimental designs, the Central Composite Design (CCD) has been widely used. However, the number of points in CCD increases exponentially with the number of design variables, which proves to be inefficient for high-dimensional design problems. Though there are small composite designs (SCD) that consist of a faction of CCD points, the SCD has significant difficulty in estimating linear and interaction coefficients [2]. Chen [3] designed a nonstandard CCD method and positive results have been achieved. Besides CCD, alphabetical optimal designs, especially D-optimal designs, are also widely used [4-9]. Myers and Montgomery [2] identified the pitfalls of the D-optimality designs, which have only model-dependent D-efficiency and do not address prediction variance. Moreover, for second-order models, the D-criterion often does not allow any (or many) center runs. This often leaves large variance in the design center. As computer experiments involve

mostly systematic error rather than random errors as in physical experiments, Sacks *et al*. [10] stated that in the presence of systematic rather than random error, a good experimental design tends to fill the design space rather than to concentrate on the boundary. They also stated that standard designs, e.g. CCD and D-optimality designs, can be inefficient or even inappropriate for deterministic computer codes. Simpson *et al.* [11] confirmed that a consensus among researchers is that experimental designs for deterministic computer analyses should be space filling. Space filling methods include orthogonal arrays [12,13] and various Latin Hypercube Designs [14-18].

The second direction of RSM related research is on developing better response surface models. Besides the commonly used polynomial functions, Sacks *et al.* [10,19] a stochastic model, called Kriging, to treat the deterministic computer response as a realization of a random function with respect to the actual system response. Neural networks have also been applied in generating the response surfaces for system approximation. Other types of models include Rational Basis Functions (RBF), Multivariate Adaptive Regression Splines (MARS), and inductive learning. A combination of polynomial functions and artificial neural networks has also been archived in [20]. So far there is no conclusion about which model is definitely superior to the others. However, much more insight on various models has been gained recently [11,21-25]. Among various models, Kriging and second-order polynomials are the most intensively studied. The general consensus is that the Kriging model is more accurate for nonlinear problems but difficult to obtain and use. On the contrary, the polynomial model is easy to construct, clear on parameter sensitivity, and cheap to work with but is less accurate than the Kriging model [23,25].

The third direction of research aims at developing methods that can iteratively improve the accuracy of the response surface modeling [26]. The early work of Box and Draper [27] proposed a method to gradually refine the response surface to better capture the real function. Chen [3] intended to develop some heuristics to lead the surface refinement to a smaller design space. The *variable-complexity response surface modeling* method uses analyses of varying fidelity to reduce the design space to the region of interest. This method has been successfully used in various structural designs [7]. Wujek and Renaud [29,30] compared a number of move-

limit strategies that all focus on controlling the function approximation in a more "meaningful" design space.

Besides the three direct RSM-related research streams, RSM has also been intensively used in robust design and multidisciplinary design [3,31-34], to name a few).

The author and his colleagues have developed a so-called Adaptive Response Surface Method (ARSM), which was in the third direction of RSM-related research [35]. At each iteration, ARSM discards portions of the design space that correspond to function values larger than a given threshold value. Thus, the design space reduces gradually to the neighborhood of the global design optimum. Such a space reduction method is characterized by its systematic consideration of the interrelationship between design variables. One major limitation of ARSM is that the number of required design experiments increases exponentially with the number of design variables. In addition, at each iteration, ARSM requires a completely new set of CCD points, i.e., design experiments at previous iterations cannot be inherited. These two limitations are largely imposed by the use of CCD in the ARSM. This work identifies the Latin Hypercube Design (LHD) as a substitute of CCD and modifies the process of ARSM to overcome these two limitations.

### *Latin Hypercube Designs*

Latin Hypercube Sampling, or Latin Hypercube Design (LHD), was first introduced by McKay *et al.* [14]. This method was found to be more accurate than random sampling and stratified sampling to estimate the means, variances and distribution functions of an output. Moreover, it ensures that each of the input variables has all portions of its range represented. It can cope with many input variables and is computationally cheap to generate. LHD was developed in the statistics community [14-18]. It began to gain attention in engineering design since its first use in computer experiments by Sacks *et al*. [10,19].

In practice, Latin Hypercube Design samples can be obtained as follows. The range of each design input variable is divided into $n$ intervals, and one observation on the input variable is made in each interval using random sampling with each interval. Thus, there are $n$ observations

on each of the $d$ input variables. One of the observations on $x_1$ is randomly selected (each observation is equally likely to be selected), matched with a randomly selected observation on $x_2$, and so on through $x_d$. These collectively constitute a design alternative $X_1$. One of the remaining observations on $x_1$ is then matched at random with one of the remaining observations on $x_2$, and so on, to get $X_2$. A similar procedure is followed for $X_3$, …, $X_n$, which exhausts all observations and results in $n$ LHD sample points [15].

Mathematically, suppose that ($F_1$, …, $F_d$) are distribution functions of the independent input variables ($x_1$, …, $x_d$) and $x_{ij}$ is the $i$th value of the $j$th variable $x_j$ for $i=1, …, n, j=1, …, d$. Define $P=(p_{ij})$ to be an $n\times d$ matrix, where each column of $P$ is an independent random permutation of ($1, …, n$). Moreover, let $r_{ij}$ be $n\times d$ values of uniform [0, 1] random variables independent of $P$. Then the design sites $x_{ij}$ of a LHD sample are defined by

$$x_{ij} = F^{-1}[\frac{1}{n}(p_{ij} - r_{ij})] \quad (1)$$

In Eq. (1), $p_{i1},\cdots,p_{id}$ determine in which 'cell' $x_{ij}$ is located, and $r_{i1},\cdots,r_{id}$ determine where in the cell $x_{ij}$ is located [17]. In this work, all the distribution functions, $F$, are assumed to be uniform.

Latin Hypercube Design has a few distinctive features that are desirable for ARSM:

1. LHD provides more information within a design space. As illustrated before, this feature is desired for the approximation of computer experiments that bear mainly the system error rather than the random error. The other widely used space-filling sampling method is Orthogonal Array (OA). OA can generate a sample with better space-filling property than LHD. However, the generation of an OA sample is more complicated than LHD [12,13]. In addition, OA demands strict level classification for each variable, which might bring difficulty in real design. In real design, not all combinations of variable level lead to realistic design solutions, and some may cause the crash of the analysis or simulation, which is not uncommon in finite element analysis. In that case, the engineers must manually adjust variables to an appropriate number, deviating from one of the defined levels. Thus the property of OA might be undermined. The same situation might happen to LHD. But LHD allows the variable adjustment to a certain degree without undermining the property of a LHD sample, as long as the new point still falls into the same variable interval as before.

Moreover, in this work, LHD sample points will be inherited between design iterations to increase the efficiency of ARSM, but how to inherit an OA is a new challenge. Because the flexibility, ease of construction, and good space-filling property of LHD, this work will use LHD instead of OA as the sampling method.

2. LHD provides uniform random sampling. They treat every design variable as equally important and ensures uniformly distributed sampling in a given design space. Thus, no interrelationship between design variables is pre-assumed in LHD.
3. The size of a LHD sample is controllable. The sample size is determined by the designer, who is usually constrained by the budget, time, or other conditions. When the number of points is equal to the number of model coefficients, we refer to this group of designs as being "saturated". A saturated group of samples represents the minimum requirement to fit a second-order model. LHD provides the capability to generate a saturated design because the sample size is controllable.

Besides the above-listed three features, LHD allows the design inheritance, which will be discussed in more detail later. For various optimal LHDs, (e.g. minimax, minimum integrated mean square error, maximum entropy, and orthogonal), there is no consensus on which criterion is better. In addition, the identification of any set of optimal LHD sample is an optimization problem and thus requires extra computation. Thus, this work uses only the simplest LHD. Further study may help to identify a type of optimal LHD that can justify the extra computational effort.

### *A Brief Overview of ARSM*

ARSM is developed for design optimization involving computation-intensive processes for the evaluation of objective and constraint functions. A standard non-linear optimization problem is usually formulated as

$$\textit{Minimize} \qquad f(\vec{x}) \qquad \vec{x} = [x_1, \cdots, x_n] \tag{2}$$

$$\textit{subject to} \qquad h_j(\vec{x}) = 0, \quad (j = 1, \cdots, J) \tag{3}$$

$$g_k(\vec{x}) \le 0, \quad (k = 1, \cdots, K) \tag{4}$$

$$x_{L,\,i} \le x_i \le x_{U,\,i} \quad (i = 1, \cdots, n) \tag{5}$$

The lower and upper bounds $x_{L,i}$ and $x_{U,i}$ in Eq. (5) are initial variable bounds imposed by the designer. To reduce the number of computation-intensive processes, a response surface model, which is also called a surrogate, is built for either the objective or constraint function. The design optimization is thus based on surrogates of objectives and constraints. The optimization problem therefore becomes

$$\textit{Minimize} \qquad \tilde{f}(\vec{x}) \qquad \vec{x} = [x_1, \ \cdots, \ x_n] \tag{6}$$

$$\textit{subject to} \qquad \tilde{h}_j(\vec{x}) = 0, \quad (j = 1, \cdots, J) \tag{7}$$

$$\tilde{g}_k(\vec{x}) \le 0, \quad (k = 1, \cdots, K) \tag{8}$$

$$x_{L,\,i} \le x_i \le x_{U,\,i} \qquad (i = 1, \ \cdots, \ n) \tag{9}$$

where the tilde symbol means surrogates for corresponding functions in Eqs. (2) - (4). ARSM uses the model defined by Eqs. (6) - (9) for design optimization.

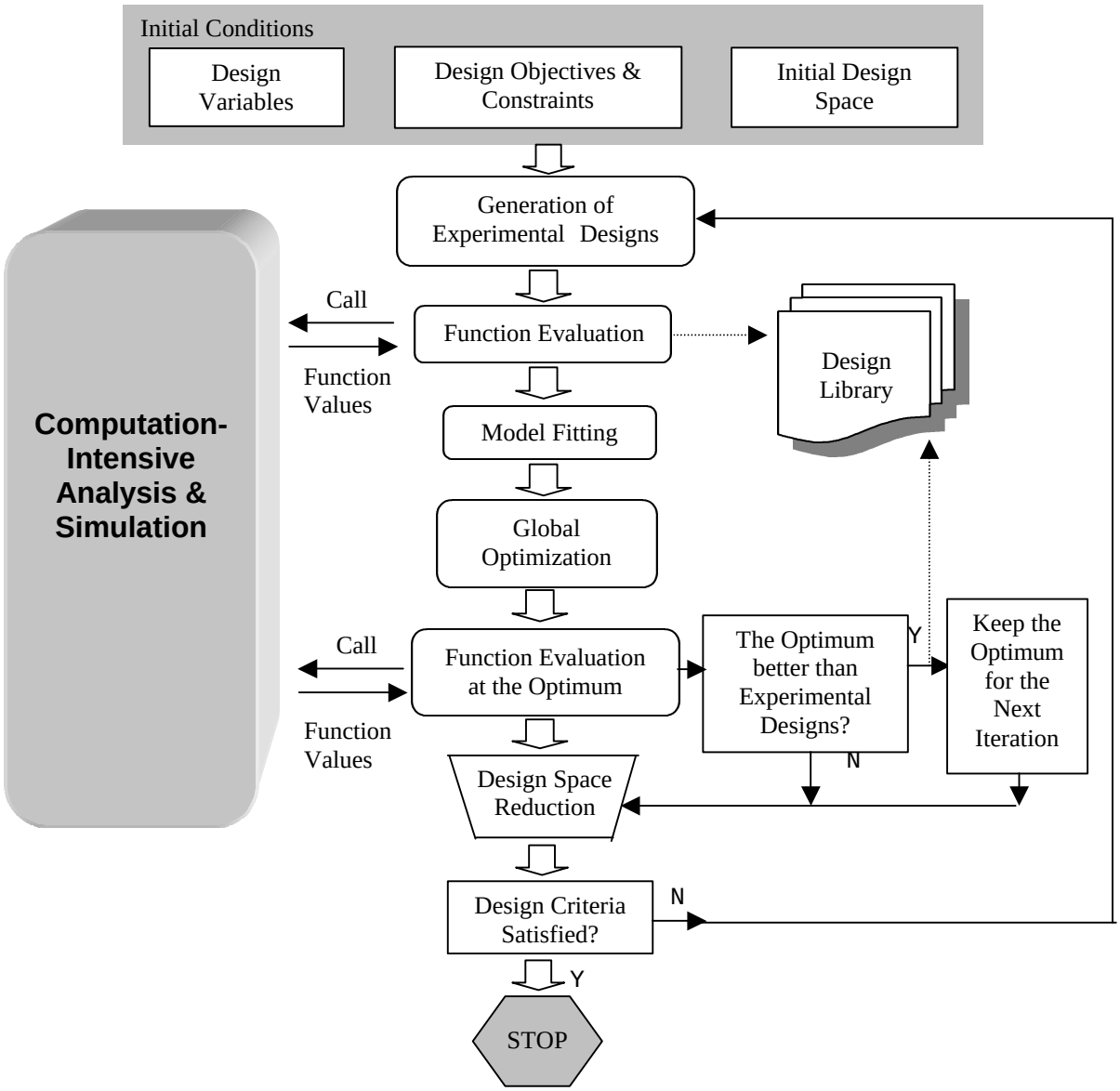



**Figure 1 The flowchart of ARSM.**

ARSM is based on the Response Surface Method, in which the design space is systematically reduced according to a given threshold output function value. Assume we are minimizing a design objective with respect to a number of design variables. As shown in Figure 1, given design variables, objectives, constraints, and the initial design space, experimental designs (points in the design space) are generated according to a formal planning strategy, *e.g.* CCD. Values of the objective function are evaluated through the computation intensive analysis and simulation processes. The quadratic response surface model, or surrogate, is fitted to the data using the least square method. When constraints are considered, a global optimization algorithm

is used to find the optimum. Following this step, the value of the actual objective function at the optimum of the surrogate is calculated through an evaluation of the computation-intensive objective function. If the value of the actual objective function at the surrogate optimum is better than the values at all other experimental designs, the point is added to the set of experimental designs for the following iteration because the point represents a promising search direction. All experimental designs and the accepted model optimum are recorded in a design library. Then a threshold value of the objective function is chosen. The design space that leads to objective function values larger than the threshold is then discarded. In ARSM, the second highest value of the objective function in the set of experimental designs is chosen as the threshold. If this second highest value cannot help to reduce the design space, the next highest value of the design function will be used, and so on. The optimization process will terminate if a satisfactory design emerges in the design library or the difference between the lower limit and the upper limit of each design variable is smaller than a given small number.

## Identification of the Reduced Design Space

A second-order response surface model is usually formulated as in Eq. (10).

$$y = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^{\,2} + \sum_{i<j} \sum_{j=1}^{n} b_{ij} x_i x_j \tag{10}$$

where $b_i$, $b_{ii}$, and $b_{\mathrm{ij}}$ represent regression coefficients; $x_i, (i = 1 \cdots n)$ are design variables and $y$ is the response. Assume the threshold for the objective function is $y_{0,}$ the range of design variable, $x_k$ $(\mathrm{k} = 1, 2, \cdots, \mathrm{n})$, can be obtained by optimizing $x_k$ with respect to all other design variables. Rearranging the response surface model given in Eq. (10) gives a quadratic function in $x_k$, with respect to other design variables in the coefficients as in Eq. (11),

$$b_{kk} x_k^{\,2} + (b_k + \sum_{i=1,<k}^{n} b_{ik} x_i + \sum_{i>k}^{n} b_{ki} x_i) x_k + [\sum_{i=1,\neq k}^{n} (b_{ii} x_i^{\,2} + b_i x_i) + \sum_{i<j,\neq k}^{n} \sum_{j=1,\neq k}^{n} b_{ij} x_i x_j + b_0 - y_0] = 0 \tag{11}$$

which can be written as

$$a x_k^2 + b x_k + c = 0,$$

where,

$$a = b_{kk}$$

$$b = b_k + \sum_{i=1,<k}^{n} b_{ik} x_i + \sum_{i>k}^{n} b_{ki}\, x_i$$

$$c = \sum_{i=1,\neq k}^{n} (b_{ii} x_i^{\,2} + b_i x_i) + \sum_{i<j,\neq k}^{n} \sum_{j=1,\neq k}^{n} b_{ij} x_i x_j + b_0 - y_0 \qquad (12)$$

The two solutions of $x_k$, $x_{k,1}$ and $x_{k,2}$, are given by

$$\left.\begin{matrix} x_{k,1} \\ x_{k,2} \end{matrix}\right\} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \qquad (a \neq 0) \qquad (13)$$

The reduced range of $x_k$ is identified by finding the minimum and maximum of $x_k$ with respect to all other design variables within their bounds, $x_i \in [x_{l,i}, x_{u,i}]$. Two subsidiary optimization problems are formulated as:

For the lower limit of $x_k$,

*Minimize* $\min\{x_{k,1}, x_{k,2}\}$ (14)

*subject to*

$$x_{l,i} \leq x_i \leq x_{u,i} \quad (i = 1, \cdots, k-1, k+1, \cdots, n) \qquad (15)$$

where $x_{l,i}$ and $x_{u,i}$ are, respectively, the lower and upper limits of $x_i$ from the previous model.

For the upper limit of $x_k$,

*Minimize* $-\max\{x_{k,1}, x_{k,2}\}$ (16)

*subject to*

$$x_{l,i} \leq x_i \leq x_{u,i} \quad (i = 1, \cdots, k-1, k+1, \cdots, n) \qquad (17)$$

These optimization problems involve non-linear objective functions that have no guaranteed unimodality. The *simulated annealing* global optimization method is thus applied for their solutions [36]. Though there are *2n* subsidiary optimization processes involved in the space reduction process, the time for each optimization is usually less than one tenth of a second on a UNIX workstation. Compared to the possible hours of computation time for each computation-intensive process, the total time for space reduction is negligible.

### Limitations of ARSM

As discussed in [35], the required number of design experiments, and thus the computation-intensive processes, increases exponentially with the number of design variables. Therefore the current ARSM is only efficient for small-scale design problems. This limitation is imposed by the use of CCD. The other limitation is that in a reduced design space, a completely new set of CCD points is generated in order to maintain the structure of a CCD sample. Because CCD points are defined by the design space alone, the design points are different for a different design space. In ARSM, a dilemma is whether to maintain the structure of CCD, i.e., generating a complete new set of points according to the reduced design space, or to inherit previous design experiments. ARSM chose the former. In this work, due to the use of Latin Hypercube Design, the above two limitations are overcome. That is, the number of design experiments increases at a much reduced rate for high-dimensional design problems, and the design experiments can be inherited due to the random nature of the Latin Hypercube Design.

### *ARSM Using Inherited LHD Points*

This work addresses the improvements over the ARSM archived in [35]. For clarity, the modified ARSM is referred as the improved ARSM and the one developed before is referred as the previous ARSM for the later sections of the paper. Figure 2 shows the flowchart of the improved ARSM, which is similar to that of the previous ARSM shown in Figure 1. For the ease of comparison, features unique to the improved ARSM are boldfaced. As we can see from the figure, the main stream of data flow remains the same. The difference lies on the generations of experimental designs. At the beginning of the optimization process, saturated LHD points are generated instead of the CCD points. The model optimum is added to the design library and inherited as long as it falls into the reduced design space. Existing design experiments that fall into the reduced design space will also be inherited for the next iteration. If the total number of inherited points is less than the number of model coefficients or the inherited points do not form a LHD sample, new LHD points will be generated. The new set of design experiments will then be used to fit a new response surface and the optimization process continues. The termination criteria remain the same as those of the previous ARSM.

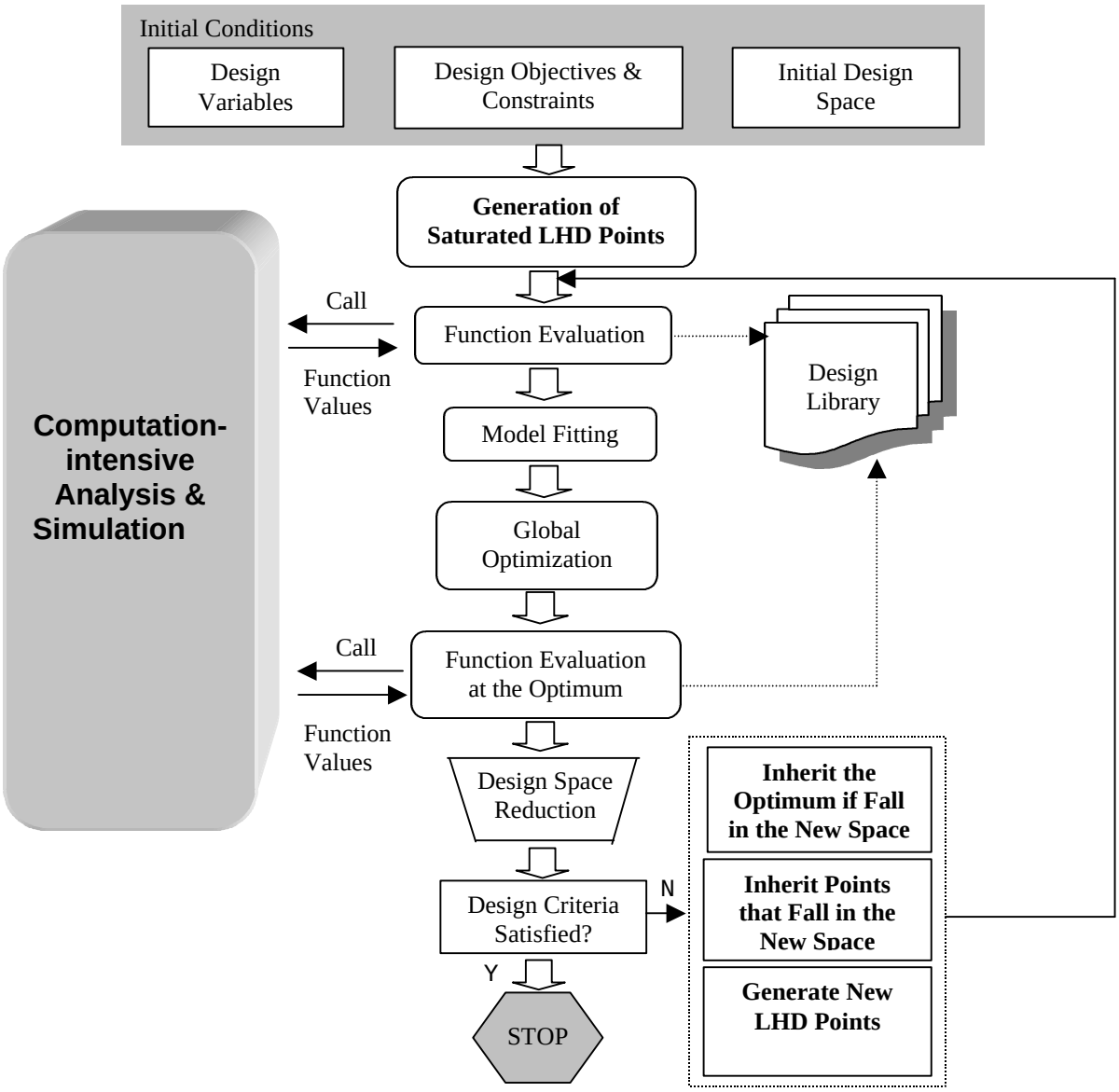



**Figure 2 The flowchart of the improved ARSM.**

As the ARSM iteratively constructs surrogates, a new set of experimental designs is formed at each iteration. In this work, some of the design points from the previous iteration will be inherited as long as they fall into the new reduced design space. To generate additional points to form a new sampling set with the inherited ones, two methods have been tested in this work. Method I is to simply generate new LHD points so that the total number of points will be equal to the number of coefficients. The new set of points generated will then become a combination of the inherited points and the new LHD points. Such a combination, however, usually does not form a LHD sample. To maintain the uniform distribution property of a LHD sample, the second

method was introduced. Method II will identify the positions of the inherited points in the new design space, find the intervals of each variable that are not represented by the inherited points, generate new LHD points, and then fill in those underrepresented intervals for each design variable.

To illustrate the second method, let us look an example with two variables $x_1$ and $x_2$. Assume at the second iteration of ARSM, three points fall in the new design space as shown in Figure 3. For two design variables, the number of coefficients for the second polynomial function defined by Eq. (10) is six. Therefore, six intervals for each variable are assumed. Those intervals represented by the inherited points are shaded in Figure 3.

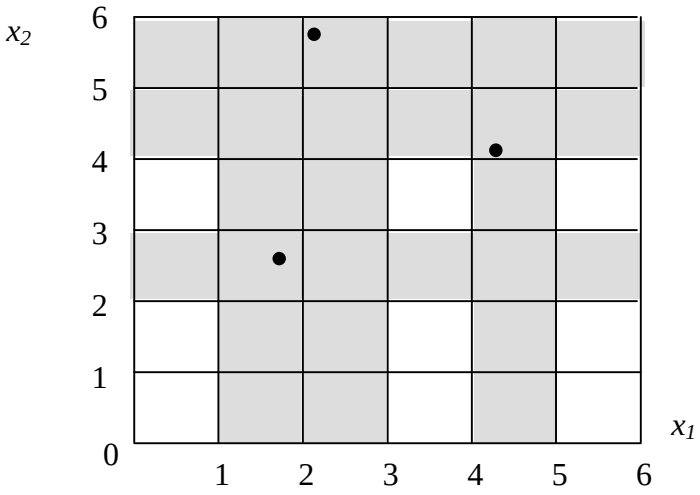



**Figure 3 Inherited design points and underrepresented variable intervals in the new space.**

In this example, if shaded areas are removed from the figure, the underrepresented variable intervals will form a blank space with three intervals for each variable. That indicates three LHD points are needed to fill in the new space to form a LHD sample. Thus, three LHD points are generated independently as shown in Figure 4.

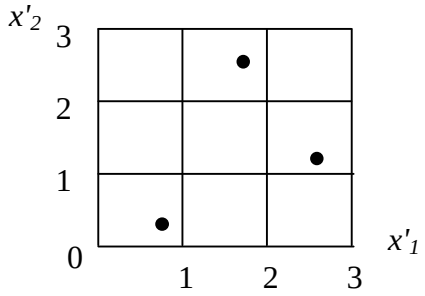



**Figure 4 A new Latin Hypercube sample of three points.**

Through a search computer program, the underrepresented intervals and their positions can be identified. Then, the new LHD sample is mapped to the un-shaded intervals in the new design space. As shown in Figure 5, Point P is one of the three new points generated according to LHD. The position of P is at the third interval of $x'_1$ and the second interval of $x'_2$. To map the point P to the real design space, the third underrepresented interval of $x_1$ and the second underrepresented interval of $x_2$ are identified. Then the relative position of P in its interval is directly mapped to that of Point $P_m$ in the real design space. Other points can be mapped in a same manner. As one can see, the combination of inherited points, represented by a black dot, with the mapped new LHD points, represented by a black square, form a Latin Hypercube sample with all the intervals of variables are represented.

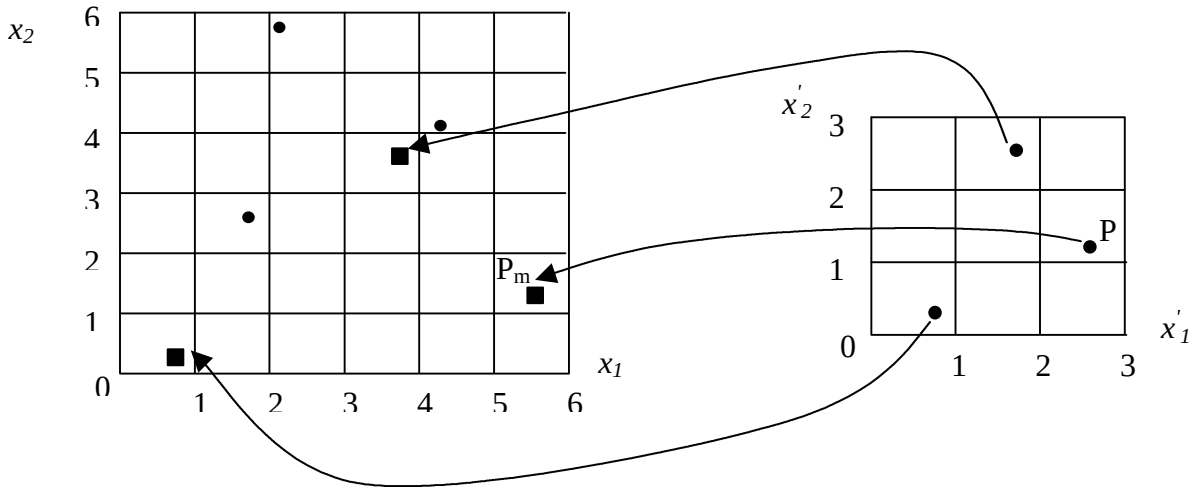



**Figure 5 Mapping the new LHD sample to the design space.**

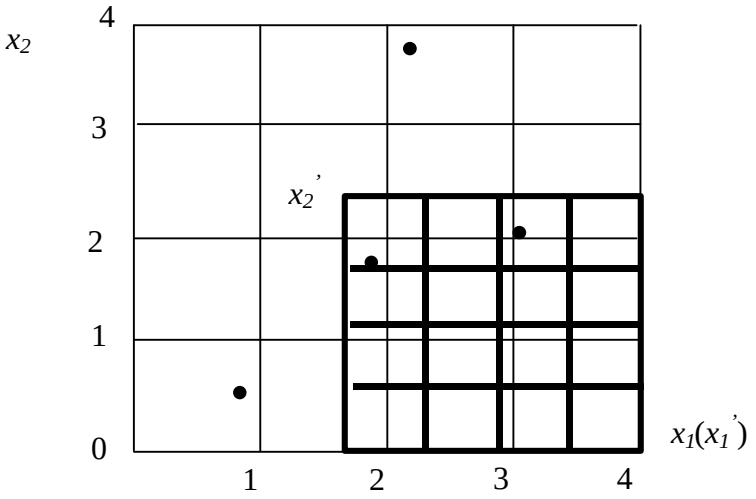



**Figure 6 Two inherited points falling in the same variable interval.**

It is not unusual that two or more inherited points might fall into the same variable interval in the new design space because of the re-defining of intervals in every new space. As one can see in Figure 6, assuming the four dots represent four LHD points in the original space and the new space is represented by the box with thick lines, there are two points fall into the reduced space. These two, however, fall into the same interval for $x_2'$ in the new space. As a result, the number of underrepresented intervals for each variable is not equal. Instead of deleting one of the points, which is a piece of valuable information obtained through computation-intensive processes, this work decides to generate points to represent all the intervals for the variable with the largest number of underrepresented intervals. For the case shown in Figure 6, the number of new points should be three. When mapping points to the real design space, a random number is generated as the new position of the to-be-mapped point along the variables with less number of underrepresented intervals. For example, in Figure 7, assuming four points represented by black dots are inherited points, the position of Point P along the $x_1$ direction should be in the fourth underrepresented interval, which is between 5 and 6 in the real design space. However, since for variable $x_2$ the number of underrepresented intervals is only three, all the intervals of $x_2$ can be represented after replacing the other three new points. In this case, a random number between 0 and 6 is generated, which indicates the position along the $x_2$ position for Point P. In Figure 7 the fourth interval is randomly chosen and P is mapped to Point $P_m$. Notice in this case, the resultant sample has seven points instead of six.

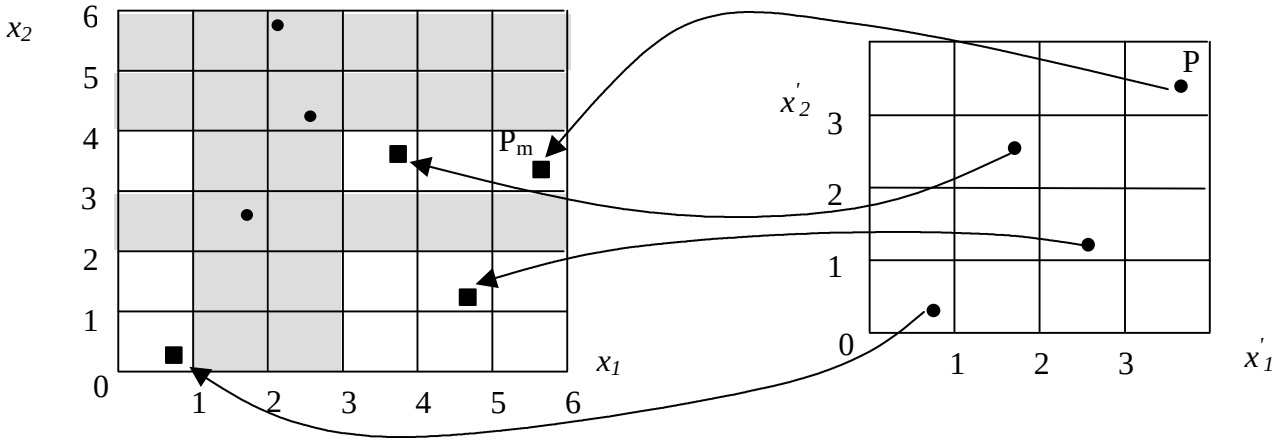
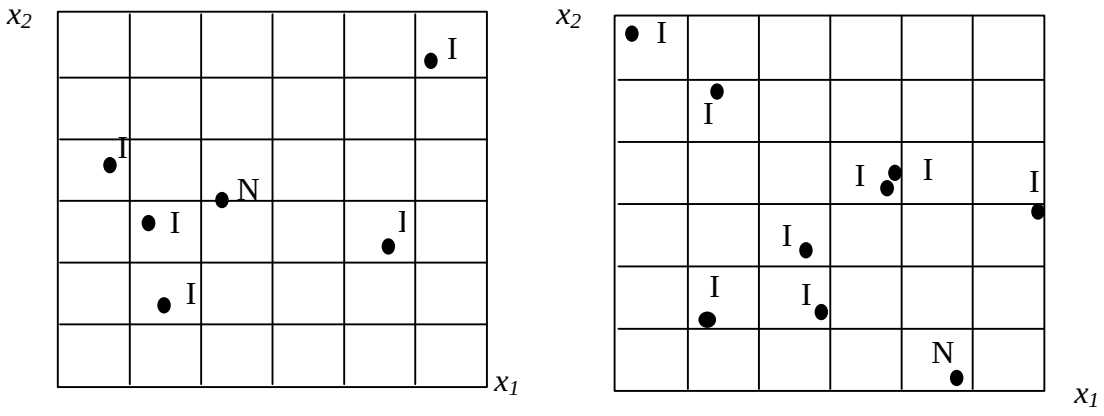



**Figure 7 Mapping when the number of underrepresented intervals for each variable is different.**

The above-illustrated process can be easily extended to multiple variables and automated through computer programming.

The inheritance Method I, which generates random points without maintaining a LHD sample after the first iteration, can always keep the number of sample points at a minimum level, such as a saturated design. This is because the inheritance Method I always generates the right new random LHD points to ensure the total number of inherited designs is equal to the required number of saturated designs. Though a LHD sample is not maintained by using Method I, a random sample can be formed at each iteration. Because the inherited LHD points come from essentially a controlled random sampling method, they are still random in nature even in a reduced design space. When combined with the new random LHD points, the inherited random points form a statistically random sample. Nevertheless, the new sample is less uniformly distributed than a LHD sample. The inheritance Method II, which maintains a LHD sample with a slight redundancy at each iteration, can ensure a set of well-distributed sample points at the expense of some extra points. For one of the test problems to be discussed in a later section, the Goldstein and Price function, the sampling points at iteration 8 for both methods are shown in Figure 8. In the figure, the letter 'I' indicates inherited points from the previous iteration; the letter 'N' indicates a new point generated at the current iteration. As one can see in Figure 8 (a), the inheritance Method I generates a random point regardless of the representation of variable intervals. As a result, there are underrepresented intervals for both variables $x_1$ and $x_2$. In Figure 8 (b), the inheritance Method II places the new point in a position that ensures all the intervals of variables have representative points.




**(a) The sample generated by Method I.** **(b) The sample generated by Method II.**

**Figure 8 Samples generated by two inheritance methods at the 8th iteration for Goldstein and Price function.**

The consequence of the two inheritance methods to the optimization results will be compared in later sections.

### *Improvements to ARSM*

By using Latin Hypercube Design, ARSM has been improved from three aspects. These aspects are described in detail in the following subsections.

#### Inherited Designs versus A New Set of CCD Points

The previous ARSM generates a complete new set of CCD points at each iteration to maintain the structure of a CCD sample. This method is not economical as the design experiments are not inherited from the previous iterations. Given the LHD method, such an inheritance becomes possible. Also because of the inheritance, more design iterations become affordable, and potentially, a more accurate optimum can be identified.

#### Inheritance of the Last Model Optimum

In the previous ARSM, the last optimum obtained from optimizing the response surface is kept for the next iteration only if it leads to a lower actual objective function value than the function values of all the other design points. The reason is that at each iteration the CCD points are at the boundaries and only one is at the center. Adding an extra point in the space implies that more emphasis is put to the region around the point against other areas of the design space. As a result, the fitted response surface model will be biased and the design optimization will be led toward that particular region. Therefore, unless the model optimum is really the best solution so far, inheriting this point will likely lead the search astray. In the improved ARSM, LHD points are scattered evenly or randomly within the design space. The inheritance of the last model optimum, even though it is not the best solution so far, will only add one more random point in the space and will not bias the model fitting or lead the search astray. Thus in the improved ARSM, the last optimum was always kept for the next iteration, as long as it falls in the reduced design space. The inheritance of the last optimum ensures that all of the information obtained through computation-intensive processes is utilized to improve the model approximation for the ensuing design iterations.

### Controllable Number of Experimental Designs

To be able to construct a second-order response surface model as in Eq. (10), at least $(n+1)(n+2)/2$ number of points (function evaluations) are required to obtain the equal number of coefficients. For computation-intensive design problems, the objective is often to reduce the number of these computation-intensive processes, which are function evaluations at each design iteration. Since the size a LHD sample is determined by the designer, one can choose saturated LHD points to fit a second-order model. On the other hand, a CCD sample consists of a $2^n$ factorial design with $2$ levels for each of $n$ variables, $2n$ axial points, and a certain number of central points. The number of CCD points is a function of the number of design variables, regardless of the number of model coefficients. If the number of design variables is $n$, then the total number of points in the CCD will be at least $2^n+2n+1$. If assuming the number of design variables is $n=10$, then the number of saturated LHD points is 66; the number of CCD points is 1045. With the increasing number of design variables, a saturated LHD sample includes much fewer design experiments than a CCD sample. The improved ARSM also represents the theoretically minimum computation cost for a second-order response surface model due to the saturated designs. Therefore, the improved ARSM has much greater efficiency than the previous ARSM in terms of the total number of computation-intensive processes. The improved ARSM is thus more suitable for high dimensional design problems.

In application, some terms of this second-order polynomial defined by Eq. (10) can be eliminated based on previous experience or other related studies. Thus, the number of required points can be further reduced. This work always assumes that the model defined in Eq. (10) is the final model and no terms can be eliminated. This assumption is made for descriptive convenience rather than a constraint because one can eliminate some terms if possible and use the modified model as a substitute for the model in Eq. (10).

In summary, ARSM can be improved in efficiency from the above-described aspects. The testing of the improved ARSM against a number of well-known optimization problems and the application of the ARSM to a design problem further confirms the improvements.

### *Testing the ARSM*

The improved ARSM has been tested with a number of widely accepted test problems for global optimization algorithms [37-41]. The test problems are listed below where *n* represents the number of variables.

- Goldstein and Price function (GP), *n* = 2.

$$f(x_{1,}x_2) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]* \\ [30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$$

$$x_{1,2} \in [-2\ 2] \tag{18}$$

- Six-hump camel-back function (SC), *n* = 2.

$$f_{sc}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,\ x_{1,2} \in [-2,\ 2] \tag{19}$$

- Branin function (BR), *n* = 2.

$$f_B(x) = (x_2 - \frac{5.1}{4p^{\ 2}}x_1^2 + \frac{5}{p}x_1 - 6)^2 + 10(1-\frac{1}{8p})\cos x_1 + 10,\ x_1 \in [-5,\ 10],\ x_2 \in [0,\ 15] \tag{20}$$

- Generalized polynomial function (GF), *n* = 2.

$$f_{SB}(x) = u_1^2 + u_2^2 + u_3^2 \tag{21}$$

$$u_i = c_i - x_1(1-x_2^i),\ i = 1,\ 2,\ 3 \qquad x_{1,2} \in [-5\ 5] \tag{22}$$

$$c_1 = 1.5,\ c_2 = 2.25,\ c_3 = 2.625 \tag{23}$$

- Rastrigin function (RS), *n* = 2.

$$f_{RS}(x) = x_1^2 + x_2^2 - \cos 18x_1 - \cos 18x_2,\ x_{1,2} \in [-1,\ 1] \tag{24}$$

- Geometric container function (GC), *n = 3*.

$$\textit{Minimize}\ \ f_{GC}(x) = 0.2/x_1x_2x_3 + 4/x_1 + 3/x_3,\ x_i \in [0,\ 5] \tag{25}$$

*subject to*

$$\begin{aligned} g_1(x) &= -x_1 \le 0 \\ g_2(x) &= -x_2 \le 0 \\ g_3(x) &= -x_3 \le 0 \\ g_4(x) &= 2x_1x_3 + x_1x_2 - 10 \le 0 \end{aligned} \tag{26}$$

- Hartman function (HN), *n* = 6.

$$f_{HN}(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2],\ x_i \in [0, 1],\ i = 1, \cdots, n \tag{27}$$

where,

| $i$ | $a_{ij},\ j = 1, \cdots, 6$ | | | | | | $c_i$ |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 3 | 17 | 3.5 | 1.7 | 8 | 1 |
| 2 | .05 | 10 | 17 | 0.1 | 8 | 14 | 1.2 |
| 3 | 3 | 3.5 | 1.7 | 10 | 17 | 8 | 3 |
| 4 | 17 | 8 | .05 | 10 | 0.1 | 14 | 3.2 |

| $i$ | $p_{ij},\ j = 1, \cdots, 6$ | | | | | |
|---|---|---|---|---|---|---|
| 1 | .1312 | .1696 | .5569 | .0124 | .8283 | .5886 |
| 2 | .2329 | .4135 | .8307 | .3736 | .1004 | .9991 |
| 3 | .2348 | .1451 | .3522 | .2883 | .3047 | .6650 |
| 4 | .4047 | .8828 | .8732 | .5743 | .1091 | .0381 |

The test results are compared with those obtained by the previous ARSM [35]. The improved ARSM with two inheritance methods are also compared and tabulated in Table 1. The third column of Table 1 lists the number of local minima of each test function within its given design space; the fourth column lists the analytical global optimum for each test problem.

**Table 1 Summary of test results on the improved ARSM**

| Test Func. | # of Var. | # of Local Min. | Global Optimum Obtained | | | | Number of Function Evaluations | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Anal. Solu. | Prev. ARSM | *Impr. ARSM Method I* | *Impr. ARSM Method II* | *Prev. ARSM* | *Impr. ARSM Method I* | *Impr. ARSM Method II* |
| GP | 2 | 4 | 3.000 | 3.210 | 3.250 | 3.000 | 70 | 30 | 77 |
| SC | 2 | 6 | -1.032 | -0.866 | -1.026 | -1.029 | 100 | 39 | 44 |
| BR | 2 | 3 | 0.398 | 2.099 | 0.417 | 0.398 | 50 | 15 | 36 |
| GF | 2 | ≥5 | 0.000 | 0.609 | 0.444 | 0.082 | 144 | 29 | 46 |
| RS | 2 | 50 | -2.000 | -2.000 | -1.417 | -1.854 | 9 | 17 | 60 |
| GC | 3 | 1 | 3.362 | 5.307 | 3.413 | 3.397 | 76 | 64 | 38 |
| HN | 6 | ≥3 | -3.320 | -3.320 | -2.652 | -2.456 | 1248 | 158 | 105 |

From Table 1, one can see that first the improved ARSM, either with the inheritance Method I or Method II, can converge to a near-global optimum. Second, in general the improved ARSM needs significantly fewer function evaluations to achieve the same magnitude of accuracy as the

previous ARSM. For functions SC, BR, GF, and GC, the improved ARSM has achieved better accuracy with much less number of function evaluations. For functions GP, the improved ARSM with Method I needs fewer function evaluations at a loss of little accuracy compared with the previous ARSM, while the ARSM with inheritance Method II has reached the global optimum with a few more function evaluations. It is to be noted that for a large number of variables, e.g. $n = 6$ in HN, the improved ARSM reduced the required function evaluations by an order of magnitude. For function RS, the previous ARSM performs better than the improved ARSM because the symmetric structure of the RS function coincidentally corresponds to the symmetric structure of CCD. Therefore, in general the improved ARSM is more efficient than the previous ARSM in terms of the number of computation-intensive function evaluations.

When comparing the improved ARSM with two inheritance methods, one can see from Table 1 that the improved ARSM with Method II demonstrates higher potential to reach the real global optimum. For functions GP, BR, SC, GF, and GC, the ARSM with Method II identifies the real optimum with negligible error. The improved accuracy of the ARSM with Method II over Method I indicates the quality of the response surface is dependent on the sample quality. This observation is confirmed by [23]. The number of iterations required by the ARSM with Method II is generally more than that of the ARSM with Method I. This is largely due to the need to ensure that the sample of designs at each iteration is a LHD sample for the ARSM with Method II.

It is to be noted that the test problems listed in this work are special problems selected to test global optimization algorithms. Their objective function values can vary significantly and involve many local minima in the design space. For example, the GP function rises up to higher than $10^9$ and has four local minima in the region [-2 2]. Those test problems are difficult to solve for local-optimization methods such as conventional direct search or gradient-based methods. ARSM was developed as a global optimization algorithm for computation-intensive design problems. Thus, the improved ARSM was compared with the previous ARSM on the premise of being a global optimization method. Even though for a two-variable optimization problem, a local-optimization method might converge with a small number of function evaluations, the obtained optimum will be a local minimum. Some deterministic global optimization methods do

not require gradient information and explicitly expressed objective functions such as Jones' DIRECT [42] algorithm. These methods may perform as comparatively well as the improved ARSM, but these methods are sequential in nature and thus lack the engineering advantages of ARSM such as the parallel computation. The following section will show how the improved ARSM is applied to a design problem.

### *A Design Example*

The capability of the improved ARSM in solving real engineering design problem is demonstrated with the aid of a simple beam design problem involving four design variables. This problem is modified from the original problem recorded in [43]. The objective of this design problem is to minimize the vertical deflection of an I-beam (Figure 9) that will simultaneously satisfy the cross-section area and stress constraints under given loads. Various parameter values for the problem are:

- Allowable bending stress of the beam = 6 kN/cm$^2$.
- Young's Modulus of Elasticity (E) = 2x10$^4$ kN/cm$^2$.
- Maximal bending forces P = 600 kN and Q = 50 kN.
- Length of the beam (L) = 200 cm

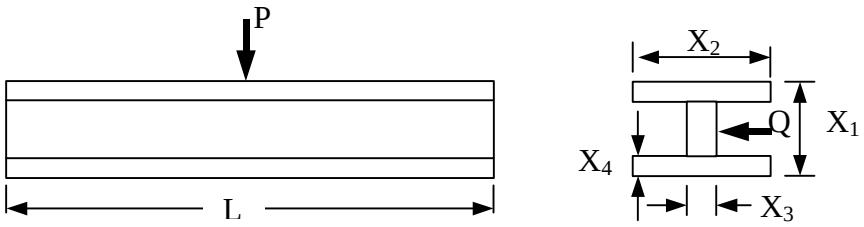



**Figure 9 A Beam Design Problem.**

The optimization problem can be formulized as below:

Minimize the Vertical Deflection f(x) = PL$^3$ / 48EI

$$f(x) = \frac{5000}{\frac{1}{12}x_3(x_1 - 2x_4)^3 + \frac{1}{6}x_2x_4^3 + 2x_2x_4(\frac{x_1 - x_4}{2})^2} \tag{28}$$

Subject to:

Cross Section Area less than 300 cm$^2$

$$g_1(x) = 2x_2x_4 + x_3(x_1 - 2x_4) \le 300 \quad (29)$$

Stress Constraint

$$g_2(x) = \frac{180000x_1}{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]} + \frac{15000x_2}{(x_1 - 2x_4)x_3^3 + 2x_4x_2^3} \le 6 \quad (30)$$

Initial Design Space

$$10 \le x_1 \le 80, \quad 10 \le x_2 \le 50, \quad 0.9 \le x_3 \le 5, \quad 0.9 \le x_4 \le 5 \quad (31)$$

The problem formulated above is a simple non-linear constrained problem. It could be solved with a variety of optimization methods. Now let us assume the objective function defined by Eq. (28) is a computation-intensive function and thus the reduction of the number of function evaluations is our concern. The improved ARSM with Method II is then applied to solve the problem. The solution is compared with that obtained by the previous ARSM and by the '*constr*' function in Matlab 5.3™ (Table 2). The '*constr*' function provides an efficient local optimization method; different starting points may lead to different local optimum. Nine different starting points are picked randomly for the above problem. Two local minima are found at

x1 = 80, x2 = 34.28, x3 = 0.9, x4 = 3.42 with *f(x) = 0.0134*

x1 = 80, x2 = 50, x3 = 0.9, x4 = 2.32 with *f(x) = 0.0131*

The number of objective function evaluations varies with the starting points. The average number of function evaluations from the nine runs is 69 evaluations with the range from 32 to 101.

**Table 2 Comparison of the Beam Design with Various Methods.**

| Methods | Number of Func. Eval. | Number of Design Iterations | Optimal Design [$x_1$, $x_2$, $x_3$, $x_4$] | Objective Value | Cross-section Area | Stress |
|---|---|---|---|---|---|---|
| Improved ARSM | 29 | 15 | [79.99 48.42 0.90 2.40] | 0.0131 | 299.97 | 4.48 |
| Previous ARSM | 125 | 8 | [80.00 37.05 1.71 2.31] | 0.0157 | 299.98 | 6.12 |
| Matlab™ | 69 | N/A | [80.00 50.00 0.90 2.32] | 0.0131 | 300 | 4.43 |

From the beam design problem one can see that the solutions obtained through three different methods all achieved a solution that satisfies the constraints. The improved ARSM reaches the best solution as the same as that obtained with Matlab™ by trying various starting points (possibly the global optimum). As a global optimization strategy that is independent on the starting point, the number of function evaluations required by the improved ARSM is even much smaller than that by Matlab™. The improved ARSM outperforms the previous ARSM in terms of both the minimum objective function value and the efficiency, i.e., the number of objective function evaluations. It is also observed that for the improved ARSM method, a few new points are added at each iteration, since many previous design points can be inherited. As a result, the optimization result improves slowly and thus needs more design iterations than the previous ARSM method, for which a complete new set of points are generated. As shown in the example, the improved ARSM method requires 15 iterations to reach the optimum, whereas the previous ARSM method only needs 8 iterations. If the maximum amount of parallel computation or, the minimum optimization time, is desired, the increase of number of design iterations may be of concern.

### *Discussion*

The work aims to improve the efficiency of the Adaptive Response Surface Method (ARSM) by virtue of the Latin Hypercube Design. From the test problems and the design example, one can see that the improved ARSM is more efficient that the previous ARSM, for either inheritance method. For the ARSM with Method II, the algorithm reached very high accuracy with fewer function evaluations than that of the previous ARSM.

## Applicability of the ARSM to Various Functions

In (Wang *et al*. 2001), it is mentioned that ARSM works well for overall-convex functions. It should be clarified that ARSM works better in general than the conventional response surface method (RSM) for non-concave functions instead of overall-convex functions. That is to say, no matter how complicated the function is, as long as there is a convex section in the design space, ARSM has high potential to identify that section and locate the optimum. This observation is taken from the testing of highly complicated functions — mostly non-convex — as described in the *Testing the ARSM* section. For pure concave functions, ARSM performs the same as the conventional RSM, because the optimum is at the boundary and the design space cannot be

reduced. The improved ARSM includes a small utility function to identify concave functions. This utility function makes ARSM applicable to all types of functions, even though ARSM bears no advantages over conventional RSM for pure concave functions.

### Constrained Optimization Problems

The ARSM solves constrained optimization problems as defined in Eqs. (6) - (9). In that model, constrains can be given explicitly or be approximated by surrogates. If constraints are computationally expensive, they can be approximated at each design iteration, similar to the objective function. If constrains are explicitly given or are of first-order or second-order for which accurate surrogates are easy to obtain, the optimum found from ARSM by using the model defined in Eqs. (6) - (9) is guaranteed to satisfy the constraints described in Eqs. (3) - (5). If the constraints are highly nonlinear and computation intensive, the satisfaction of constraints depends on the accuracy of the surrogate in the neighborhood of the obtained optimum. In general, ARSM works very well if the constraints are active in the neighborhood of the unconstrained global optimum, where ARSM yields very accurate surrogates. If the constraints are active in the region far from the neighborhood of the unconstrained global optimum, ARSM might only give a mediocre solution as the errors may come from the inaccurate fitting for both the objective and constraint functions. ARSM, however, bears advantages over the conventional RSM as the chance of finding the constrained optimum is still higher than the latter because of the smaller design space in ARSM. Though explicitly given constraints are assumed for the constrained GC function and the design problem, the achieved constrained global optimum and the high efficiency have testified the capability of ARSM for constrained optimization problems. Further research is needed to improve the space reduction strategy to consider the objective as well as constraint functions.

Though significant improvements have been made on ARSM, the selection of the threshold value (cutting plane) is still *ad hoc*. Another observation of the improved ARSM is that if the cutting is too conservative, most previous design points can then be inherited. Thus, few new points are added for the next iteration. If only one or two points are added to the design group at each iteration, the optimization result improves slowly. Therefore, an appropriate trade-off between the aggressiveness of the cutting and the process speed is to be developed. From the

author's experience, a 5~10% reduction on either bound of any of the variables indicates that an appropriate threshold was chosen and such a reduction should be accepted for the next iteration.

### *Conclusion*

The improved ARSM, using the Latin Hypercube Design (LHD) instead of the Central Composite Designs (CCD), significantly increases the efficiency of the previous ARSM and enables ARSM be used for high-dimensional problems. For a second-order response model, a saturated experimental design becomes possible. The nature of LHD also makes the design inheritance possible in ARSM, which further improves the efficiency of ARSM. From the testing and the design example, the improved ARSM demonstrates greatly enhanced efficiency over the previous ARSM. With the improved ARSM, a global design solution can be obtained with very modest computation cost for computation-intensive design problems. Though limitations exist, ARSM at the current development stage demonstrates strong potential to be a global optimization tool for design problems involving computation-intensive function evaluations. The method of inheriting Latin Hypercube Design points could be integrated to other move-limits methods in which the design space is varied. It can also be used for step-by-step sampling in a same design space to gradually improve the approximation accuracy. As a general sampling method, it might find more applications in approximation-based design optimization.

### *Acknowledgement*

The author extends special thanks to Prof. R. Haftka at the University of Florida for his early comments on D-optimal designs and Latin Hypercube Designs. The author also thanks Prof. W. Chen at the University of Illinois at Chicago, Prof. J. Renaud at the University of Notre Dame, and all the reviewers for their value feedback. Comments and source codes from Prof. J. S. Park at Chonnam National University are also very much appreciated. In addition, research funding from the Natural Science and Engineering Research Council (NSERC) of Canada is gratefully acknowledged.

### *References*

1. Haftka, R., Scott, E. P., and Cruz, J. R., 1998, “Optimization and Experiments: A Survey,” *Applied Mechanics Review*, Vol. 51, No. 7, pp. 435-448.

2. Myers, R. H., and Montgomery, D. C., 1995, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley and Sons, Inc., Toronto.

3. Chen, W., 1995, *A Robust Concept Exploration Method for Configuring Complex System*, Ph.D. Thesis, Georgia Institute of Technology.

4. Mitchell, T. J., 1974, "An Algorithm for the Construction of "D-Optimal" Experimental Designs," *Technometrics*, Vol. 16, No. 2, pp. 203-210.

5. Bernado, M. C., Buck, R., and Liu, L., 1992, "Integrated Circuit Design Optimization Using a Sequential Strategy," *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 3, pp. 361-372.

6. Sell, J. W., and Lepeniotis, S. S., 1992, "Thermoplastic Polymer Formulations: An Approach through Experimental Design," *Advances in Polymer Technology*, Vol. 11, No.3, pp. 193-202.

7. Giunta, A. A., Balabanov, V., Haim, D., Grossman, B., Mason, W. H., Watson, L. T., and Haftka, R. T., 1997, "Multidisciplinary Optimization of a Supersonic Transport Using Design of Experiments theory and Response Surface Modeling," *Aeronautical Journal*, 101(1008), pp. 347-356.

8. Unal, R., Lepsch, R. A., Engelund, W., and Stanley, D. O., 1996, "Approximation Model Building and Multidisciplinary Optimization Using Response Surface Methods," *AIAA-96-4044-CP*, pp. 592-598.

9. Unal, R., Lepsch, R. A., and McMillin, M. L., 1998, "Response Surface Model Building and Multidisciplinary Optimization Using D-Optimal Designs," *AIAA-98-4759*, pp. 405-411.

10. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., 1989a, "Design and Analysis of Computer Experiments," *Statistical Science*, Vol. 4, NO. 4, pp. 409-435.

11. Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F., 1998, "Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization," *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization*, St. Louis, MO, September 2-4, AIAA, **1** (381-391) AIAA-98-4755.

12. Taguchi, G., Yokoyama, Y., and Wu, Y., 1993, *Taguchi Methods: Design of Experiments*, American Supplier Institute, Allen Park, Michigan.

13. Owen, A., 1992, "Orthogonal Arrays for Computer Experiments, Integration, and Visualization," *Statistica Sinica*, No. 2, pp. 439-452.

14. McKay, M. D., Bechman, R. J., and Conover, W. J., 1979, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," Technometrics, Vol. 21, No. 2, May 1979, pp. 239-245.

15. Iman, R. L., and Conover, W. J., 1980, "Small Sensitivity Analysis Techniques for Computer Models with an Application to Risk Assessment," *Communication Statistics - Theory and Methods*, A9 (17), pp. 1749-1842.

16. Tang, B., 1993, "Orthogonal Array-based Latin Hypercubes," *Journal of American Statistical Association*, Vol. 88, No. 424, Theory and Methods, pp. 1392-1397.

17. Park, J. S., 1994, "Optimal Latin-hypercube Designs for Computer Experiments," *Journal of Statistical Planning Inference*, Vol. 39, pp. 95-111.

*18.* Ye, K. Q., Li, William, and Sudianto, A., 2000, "Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs," *Journal of Statistical Planning and Inferences*, Vol. 90, pp. 145-159.

19. Sacks, J., Schiller S. B., and Welch, W. J., 1989b, "Designs for Computer Experiments," *Technometrics*, February 1989, Vol. 31, No. 1, pp. 41-47.

20. Varaarajan, S., Chen, W., and Pelka, C. J., 2000, "Robust Concept Exploration of Propulsion Systems with Enhanced Model Approximation Capabilities," *Engineering Optimization*, Vol. 32, No. 3, pp. 309-334.

21. Giunta, A. A., and Watson, L. T., 1998, "A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models," *AIAA-98-4758, American Institute of Aeronautics and Astronautics, Inc.*, pp. 392-401.

22. Koch, P. N., Simpson, T. W., Allen, J. K., and Mistree, F., 1999, "Statistical Approximations for Multidisciplinary Design Optimization: The Problem of Size," *Journal of Aircraft*, Vol. 36, No. 1, pp. 275-286.

23. Jin, R., Chen, W., and Simpson, T., 2000, "Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria," *8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 6-8, 2000.

24. Lin, Y., Krishnapur, K., Allen, J. K., and Mistree, F., 2000, "Robust Concept Exploration in Engineering Design: Metamodeling Techniques and Goal Formulations," *Proceedings of the 2000 ASME Design Engineering Technical Conferences*, DETC2000/DAC-14283, September 10-14, 2000, Baltimore, Maryland.

25. Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K., 2001, "Metamodels for Computer-based Engineering Design: Survey and Recommendations," *Engineering with Computers*, No. 17, pp. 129-150.

26. Dennis, J. E., and Torczon, V., 1996, "Managing Approximation Models in Optimization," in Alexandrov, N. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State of the Art*, Society for Industrial and Applied Mathematics, Philadelphia.

27. Box, G. E. P., and Draper, N. R., 1969, *Evolutionary Operation: A Statistical Method for Process Management*, John Wiley & Sons, Inc., New York.

28. Giunta, A. A., Balbanov, V., Kaufmann, M., Burgee, S., Grossman, B., Haftka, R. T., Mason, W. H., Watson, L. T., 1996, "Variable Complexity Response Surface Design of an HSCT Configuration," *Multidisciplinary Design Optimization: State of the Art*, Alexandrov, N. and Hussaini, M. Y., editors, Society for Industrial and Applied Mathematics, Philadelphia.

29. Wujek, B. A. and Renaud, J. E., 1998a, "New Adaptive Move-Limit Management Strategy for Approximate Optimization, Part 1," *AIAA Journal*, Vol. 36, No. 10, pp. 1911-1921.

30. Wujek, B. A. and Renaud, J. E., 1998b, "New Adaptive Move-Limit Management Strategy for Approximate Optimization, Part 2," *AIAA Journal*, Vol. 36, No. 10, pp. 1922-1934.

31. Renaud, J. E., and Gabriele, G. A., 1993, "Improved Coordination in Non-hierarchical System Optimization," *AIAA Journal*, Vol. 31, pp. 2367-2373.

32. Renaud, J. E., and Gabriele, G. A., 1994, "Approximation in Non-hierarchical System Optimization," *AIAA Journal*, Vol. 32, pp. 198-205.

33. Chen, W., Allen, J. K., Tsui, K. L., and Mistree, F., 1996, "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors," *Journal of Mechanical Design, Transactions of the ASME*, Vol. 118, pp. 478-485.

34. Korngold, J. C., and Gabriele, G. A., 1997, "Multidisciplinary Analysis and Optimization of Discrete Problems Using Response Surface Methods," *Journal of Mechanical Design*, Vol. 119, pp. 427-433.

35. Wang, G., Dong, Z., and Aitchison, P., 2001, "Adaptive Response Surface Method — A Global Optimization Scheme for Computation-intensive Design Problems," *Journal of Engineering Optimization*, Vol. 33, No. 6, pp. 707-734.

36. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-680.

37. Dixon, L., and Szegö, G., 1978, "The Global Optimization Problem: An Introduction", *Toward Global Optimization 2*, L. Dixon and G. Szegö (ed.), North-Holland, New York, pp. 1-15.

38. Hock, W., and Schittkowski, K., 1981, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer-Verlag, Berlin, Heidelberg, New York.

39. Pardalos, P. M., and Rosen, J. B., 1987, *Constrained Global Optimization: Algorithms and Applications*, Lecture Notes in Economics and Mathematical Systems, Vol. 268, Springer-Verlag, Berlin, Heidelberg, New York.

40. Schittkowski, K., 1987, *More Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 282, Springer-Verlag, Berlin, Heidelberg, New York.

41. Törn, A., and Zilinskas, A., 1987, *Global Optimization*, Lecture Notes in Computer Science, Vol. 350, Springer, Berlin, Heidelberg, New York.

42. Jones, D. R., Perttuen, C. D., and Stuckman, B. E., 1993, "Lipschitzian Optimization Without the Lipschitz Constant," *Journal of Optimization Theory and Application*, Vol. 79, No. 1, pp. 157-181.

43. Gold, S., Krishnamurty, S., 1997, "Trade-offs in Robust Engineering Design," *Proceedings of the 1997 ASME Design Engineering Technical Conferences*, DETC97/DAC3757, September 14-17, Saramento, California.