Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 4s (2025)

Comparison Between Different Types of Latin Hypercube Designs Applied to Kriging Models

Zainab Abdulateef Rasheed[1], Younus Al-Taweel[2], Shaymaa M. Younus[3]

[1,2,3] Department of Mathematics, College of Education for Pure Science, University of Mosul, Iraq.
Corresponding author: Zainab Abdulateef Rasheed
zainab.abdulateef@uomosul.edu.iq[1], younus.altaweel@uomosul.edu.iq[2], shaymaa.mohammed@uomosul.edu.iq [3]

Article History:

Received: *14-10-2024*

Revised: *21-11-2024*

Accepted: *02-12-2024*

Abstract:

Numerous scientific fields employ Kriging models (KMs) to study how physical systems behave. The computer code's response is interpreted by the KM as Gaussian process. A well-designed is an essential part of the KMs. Our study involves the design and analysis that is used in constructing KMs for computer experiments. Evaluating how design affects prediction accuracy is what we are interested in. Latin hypercube designs (LHDs) are commonly used for building KMs. We investigate many types of LHDs and assess their performances. We built many KMs based on different types of LHDs and the performance of each model is compared. The comparison is performed through some measures that test the KM performance. We apply KMs to the Robot Arm function as a real computer code.

Keywords: Kriging models, design and analysis, Latin hypercube design, computer code.

1. Introduction

Computer experiments are used more frequently in recent years to replace physical tests that are thought to be too expensive, impractical, or impracticable. The creation of a computer code (CC) is a prerequisite for conducting a computer experiment. Running the CC with different input values is called a computer experiment. Large numbers of input variables and a lot of time are normally required for computer experiments. To quickly produce many results and lower the computational costs, Kriging models (KMs), are required. These models are frequently fitted using only a few data points, and they should quantify uncertainty, which measures the model's prediction accuracy. If a KM is chosen carefully, it can even be more beneficial than the underlying physical system because it removes noise. For computer experiments that have deterministic outcomes, it would be advantageous for the GP model to interpolate the observations.

The KM has already been used to model data from computer simulations in many scientific disciplines. Consequently, [1] proposed a KM as a metamodel for the CC. Numerous scientific fields have previously employed the KM to model data from computer simulations. A few samples of recent works are shown here, in engineering, [2] was able to achieve approximations by using a KM on CC for power cylinders. [3] created a KM to load CC for wind turbines. In climate science, [4] obtained simulated data on soil moisture using a CC and a KM. In computer experiments, [5]

compared tow different methods for building KM. [6] proposed a few methods for KM validation. [7] introduced nonseparable dependent covariance matrix to generate multivariate KM for multi-output CC. [8] suggested a few measures for multivariate KM validation.

A computer experiment could take several hours or more to complete on one run. Consequently, it may not be possible to use a CC directly for the design of computer experiments. As such, when drawing conclusions based on a CC, selecting a suitable set of input points is crucial. Design for computer experiments is the term used in the literature to describe this quickly expanding topic. Space-filling have become increasingly popular for computer experiments. Numerous distinct design types, including Latin hypercube designs (LHD), designs based on random sequences fall within the category of space-filling designs. The majority of these designs exhibit uniform space-filling characteristics, meaning that the points are evenly distributed throughout the space.

In recent years, LHDs have become very popular in building KMs for computer experiments. Therefore, several types of LHDs have been suggested. This essay aims to investigate the performance of several types of LHDs. We first build many KMs, each of which is built based on a particular type of LHDs, then we compare between the predictions of the KMs through some suggested measures.

This paper is listed as follows. Section 2 reviews constructing KMs to obtain predictions of the CC. Section 3 discusses the effect of design and experiments for computer experiments and presents several types of LHDs. In section 4, several validating measures for KMs are proposed. In Section 5, several KMs that are built using different types of LHDs are applied on a real CC and the results are compared. In Section 6, we give the conclusion of our results.

2. Constructing Kriging Models

The CC is considered to be unknown, and a Gaussian process can be used as a prior distribution to represent it. As a result, the KMs is a distribution that can be utilized to produce CC predictions and measure their degree of uncertainty.

$$\hat{y} = \sum_{i=1}^{p} \beta_i \eta_i(\mathbf{x}) + z(\mathbf{x}) \quad (1)$$

with $z(\mathbf{x})$ is thought to be a Gaussian process

$$z(\mathbf{x}) \sim GP(0, \sigma^2 R(\mathbf{x_i} - \mathbf{x_j})) \quad (2)$$

where and $\boldsymbol{\beta}$ are the coefficient parameters, σ^2 is the overall variance. We use here the Matérn covariance function:

$$R_{\nu=5/2}(\mathbf{x}_i - \mathbf{x}_j) = \prod_{i=1}^{P} \left(\frac{1+\sqrt{5}|x_i - x_j|}{\theta_i} + \frac{5|x_i - x_j|^2}{3\theta_i^2} \right) \exp\left(-\frac{\sqrt{5}|x_i - x_j|}{\theta_i} \right) \quad (3)$$

where $\theta_i \geq 0$ is the correlation parameter that determines the prediction's smoothness and $\nu = \frac{5}{2}$ manages the existence of derivatives.

We consider the output $y = \eta(\mathbf{x})$ evaluated at $\mathbf{x}$. Suppose $\mathbf{X} = (\mathbf{x_1}, \dots, \mathbf{x_n})$ is a set of input and $\mathbf{y} = (y_1 = \eta(\mathbf{x_1}), \dots, y_n = \eta(\mathbf{x_n}))$ are the outputs. The linear predictor at a new input $\mathbf{x}^*$ is

$\hat{y}(\mathbf{x}^*) = \lambda^T(\mathbf{x}^*)\mathrm{y}.$ (4)

The prediction's mean squared error (MSE) is

$\mathrm{MSE}\big(\hat{y}(\mathbf{x}^*)\big) = E[\lambda^T(\mathbf{x}^*)\mathbf{y} - y(\mathbf{x}^*)]^2$ (5)

with

$\boldsymbol{E[\lambda^T(x)^*\mathbf{y}] = E[\mathbf{y}(x^*)]}$ **(6)**

where $y(\mathbf{x}^*) = \eta(\mathbf{x}^*)$**.**

From equation (4), the best predictor minimizes MSE in equation (5).

Now, we suppose $\mathbf{f}(\mathbf{x}^*) = \{\eta_1(\mathbf{x}^*), \dots, y_n = \eta_k(\mathbf{x}^*)\}^T$. Here, we use universal KMs. Also, suppose $\boldsymbol{F} = [\mathbf{f}(\mathbf{x_1}), \dots, \mathbf{f}(\mathbf{x}_n)]^{\boldsymbol{T}}$ are regression functions evaluated at $\mathbf{x}_1, \dots, \mathbf{x}_n$. Moreover, consider $\mathbf{R}$ is the correlation matrix for. Thus, conditional on, $\mathbf{y}$, the output prediction is a normal $N\big(\hat{y}(\mathbf{x}^*), {\sigma_{y^*}}^2\big)$ where

$\hat{y}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}^*)^T\widehat{\boldsymbol{\beta}} + \mathbf{r}^T\mathbf{R^{-1}}(\mathbf{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}})$ (7)

${\sigma_{y^*}}^2 = \hat{\sigma}^2[1 - \mathbf{r}^T\mathbf{R^{-1}}\mathbf{r} + \mathbf{u}^T(\mathbf{F}^T\mathbf{R^{-1}}F)^{-1}\boldsymbol{u}]$ (8)

where

$\hat{\beta} = (\boldsymbol{F^T R^{-1} F})^{-1}\boldsymbol{F^T R^{-1}}\mathbf{y}$ (9)

$\boldsymbol{r} = \{R(\mathbf{x}^*, \mathbf{x_1}), \dots \dots R(\mathbf{x}^*, \mathbf{x_n})\}^{\boldsymbol{T}}$

$\boldsymbol{u} = \boldsymbol{F^T R^{-1} r} - \mathbf{y}$

3. Efficient designs for computer experiments

Building a KM to predict CC outputs at unknown inputs is one of the main goals of computer experiments. Accordingly, the prediction error should serve as the basis for proper optimality criteria when selecting a design in computer experiments. A good design must generate dense points in the entire design space and produce dense points in the projection of each variable. LHDs are constructed with projections that are uniformly distributed throughout the points of each input. Regretfully, it's possible that these designs don't fit the hypercube completely. A number of optimality criteria are presented to help choose a good design from a type of LHDs. The majority of these criteria rely on a measure that measures the degree of dispersion of point's set.

Replications at a specific input set must be avoided because the CC is deterministic and does not offer any more insight into the response. Therefore, space-filling LHDs are good for computer experiments because they facilitate the fitting of precise KMs. Naturally, when the main goal is to use a KM to simulate a CC, we would prefer designs to be space-filling. For this reason, the predictor from a KM is actually an interpolator. The untested point's position in relation to the observed points in the design space is used to calculate the prediction error. Hence, if the untested site is situated in a sub-region that receives little observation, the prediction accuracy will decline. Recently, several types of designs have been suggested to be an improvement of the LHDs. In the following subsections, we review the LHD and some designs that are based on the LHD.

3.1. Latin Hypercube Design (LHD)

The LHD was proposed by [9] and it divides the interval of the variable into n subintervals with equal probability, it is seen as a sort of stratified sample. LHD is easy to produce and the points are distributed uniformly throughout the input region for each variable. Therefore, the LHD is frequently used in the literature on computer experiments.

Suppose we need to generate n points $\mathbf{x_1}, \dots, \mathbf{x_n}$ with $\mathbf{x}_i = x_{i1}, \dots, x_{ip}$. Suppose Π is a matrix of dimension $n \times p$ with columns that are independent random permutations of $\{1, \dots, n\}$. Assume that U_{jk} is evenly distributed, identical, and independent random sample on (0,1) and independent of Π. Therfore, an LHD is generated by

$$\boldsymbol{X}_{jk} = \frac{\Pi_{jk} - U_{jk}}{n} \tag{10}$$

3.2. Maximin Latin Hypercube Design (MLHD)

The MLHD was described by [10] as a design based on determining how far apart the design points are from one another. The MLHD measures how evenly the points are distributed and depends on distance measurements between points. Suppose $\mathbf{X} \subset \chi$ be a design, that has the points $\mathbf{x_1}, \dots, \mathbf{x_n}$ and suppose d is a metric. The shortest distance between design points is first calculated. Then, the MLHD can provided by

$$\boldsymbol{X} = \max_{\mathbf{X} \subset \Omega} \min_{(x,x') \in \mathbf{X}} dis(x_i, x_j) \tag{11}$$

Consequently, the design points are ensured to be uniformly distributed throughout the input space using the maximin LHD [11].

3.3. Improved Latin Hypercube Design (ILHD)

The ILHD is a technique, suggested by [12], that aims at enhancing the distribution of sample points in a multi-dimensional space through specific mathematical methods. This approach focuses on improving the distribution of points within the hypercube by reducing the variance between the distances between points, thereby enhancing estimation accuracy.

This algorithm of the ILHD is predicated on the idea that selecting the best sample one after the other will result in the best set overall. The above assumption is not entirely true when the set is confined since the choices are not independent and each choice influences the quality of subsequent choices. Luckily, until the final few selections, the result set is largely unchanged.

By distributing points as projected onto a two-dimensional face of the hypercube, ILHD develops the premise that the points must have equal volume representation. A low coefficient of variation for the minimum distance between sample points was considered to be well-distributed. A decrease in the confidence interval size indicates that the quality is improved by the additional constraint. Applying the restriction to the hypercube's surfaces and edge would reduce uncertainty, therefore it stands to reason that if it could also be applied to the hypercube's volume, the advantage would increase. Hypercube volume distributions are ignored by LHD in the same way that it ignores hypercube surface distributions. As a result, ILHD will preserve a far more organized sample while preserving the system's inherent stochasticity for error estimates.

Furthermore, each ideal sample equals hypervolume representation necessitates that the minimal distance between them equal an optimal distance as required by the ILHD. Equation (12), where n is bins number and m is dimensions number, determines the optimal value.

$$d_{opt} = \frac{n}{\sqrt[m]{n}} \quad (12)$$

Finding the edge length of a hypercube of a given size involves taking the mth root of the volume per point, which is a mathematically imperfect leap from volume per point to distance. Thus, the design of ILHD aims to ensure an even distribution of points across all dimensions of the hypercube, contributing to reduced variance and improved accuracy of results in various applications.

3.4. Genetic Latin Hypercube Design (GLHD)

The GLHD works on a wide range of optimization problems. The GLHD was suggested by [13] and the description of the general GLHD is given by the following steps:

1. Generate the initial population, a population of random LHD with size N, where N must be even.
2. The best N/2 LHD are selected as "survivors," with the remaining LHD being discarded.
3. N offspring of the N/2 survivors are created in this stage. First, from the entire set of children, the best LH is assigned to numbers 1 and N/2 + 1. The kth (for k [2,N/2]) is then paired with the best LH. The best and the kth will produce a pair of offspring. Having the best LH and swapping out a random column with the matching column in the kth LH yields the first child. The child that comes out on top among the others is awarded the number k. To get the second child, take the kth LH and swap out a random column with the matching column in the best LH. Among the children, the resulting child is assigned number N/2+k.
4. All of the prior generation's offspring—aside from the first child, who is the best LH—are mutated. If the random number created for each column falls below a predetermined p, two randomly selected items in the column are switched. The random number is generated according to the uniform distribution for each column.

For more details about the GLHD see [13] and [14].

3.5. Sliced Latin Hypercube Design (SLHD)

[15] proposed SLHD the sliced Latin hypercube design. It divides each slice into slices of smaller LHDs. One appealing aspect of the SLHD is that every design slice achieves maximal homogeneity in every one-dimensional projection.

Assume that we wish to build an SLHD with n points and q slices, where a LHD makes up each slice. The q LHDs are written as $\mathbf{X_1}, \ldots, \mathbf{X_q}$. For positive integers m and t, suppose $n = mt$. Let $Z_n = \{1, \ldots, n\}$ and then divide it into m blocks, $\mathbf{b_1}, \ldots, \mathbf{b_m}$, where

$$\boldsymbol{b_i} = \{a \in Z_n | \lceil a/t \rceil = i\} \quad (13)$$

where $a \in R, \lceil a \rceil$ is the smallest integer no less than a. Then, a matrix M_n is needed to be generated as follows:

1. For $i = 1, \ldots, m$, the elements in $\mathbf{b_1}, \ldots, \mathbf{b_m}$ are permuted independently.

2. M_n is a matrix of dimension $m \times t$ with rows $\mathbf{b_1}, \ldots, \mathbf{b_m}$. For $j = 1, \ldots, t$, use a permutation that is applied independently from one column to the next to shuffle the elements in the jth column of M_n at random.

To produce an SLHD, we first use the two previously mentioned processes to individually create q sliced permutation matrices, $M_n^1, \ldots, M_n^q$. After that, for $c = 1, \ldots, t$, we create a matrix of dimension $m \times q$, M^c, by letting its jth column to be the cth column in M_n^j, $j = 1, \ldots, q$. To obtain the matrix M, we finally join $M^{(1)}, \ldots, M^{(t)}$, row by row.

$$M = \bigcup_{c=1}^{t} M^{(c)}. \tag{14}$$

Thus, $\boldsymbol{X} = x_{ik}$, a $n \times q$ SLHD with q slices and $M = m_{ik}$,, can be constructed as follows:

$$x_{ik} = \frac{m_{ik} - u_{ik}}{n} \text{ for } i = 1,2, \ldots \ldots \ldots n \ , k = 1, \ldots \ldots, q \tag{15}$$

where m_{ik} and u_{ik} are mutually independent and u_{ik} and m_{ik} are independent random variables that have a uniform distribution on [0,1).

3.6. Maximin Sliced Latin hypercube design (MSLHD)

[16] proposed the MSLHD to be an improvement over the SLHD. The MSLHD works by dividing the Latin square into parts (slices) to achieve an optimal distribution of elements within each slice while preserving the statistical properties and features of the samples. Initially, a classical LHD is formed as a matrix, with each row of the matrix containing one level for each design factor. This is followed by dividing the matrix into a fixed and predefined number of slices. Using random distribution, the distribution of points within each slice is optimized to ensure an even distribution and balance across all variables. Finally, one of the optimization algorithms or numerical methods is used to achieve an optimal solution by minimizing overlap between slices and ensuring the optimal distribution of elements within each slice.

For positive integers p, n, t and m, using t slices and m points in each of the p factors, [16] provides a straightforward and understandable way to create an $n-$run SLHD. There are only two steps in this construction method:

1. For each of the t slices, independently create t tiny LHDs $\mathrm{X}_1, \ldots, \mathrm{X}_q$, each of which has m points in p factors. To create a $n \times p$ matrix $X = \bigcup_i^t X_i$, stack the factor levels, denoted by $1, \ldots, m$.
2. Replace each of t entries in l $(l = 1, \ldots, m)$ in every of X column independently with a random permutation λ_l of the $[(1-t)t + 1, \ldots, l\ t]$ elements.

A total of $((m!)^t\ (t!)^m)^p$ potential SLHDs can be created for any given value of m, t, and p. The best SLHD among them is identified to be that maximizes an optimal criterion of the specific design.

[16] suggested a single objective function for the maximin distance criterion of the MSLHD. It is given by

$$\phi_{Mm}(X) = \frac{1}{2}\left((\phi_r(X) + \frac{1}{t}\sum_{i=1}^{t} \phi_r\left(X^i\right)\right) \tag{16}$$

where

$$\phi_r(X) = \left(\frac{2}{n(n-1)} \sum_{1 \le i \le j \le n} \frac{1}{d(x_i,x_j)^r}\right)^{\frac{1}{r}} \quad (17)$$

and

$$d(x_i, x_j) = \left\{\sum_{k=1}^{p} |x_{ik} - x_{jk}|^q\right\}^{\frac{1}{q}} \quad (18)$$

Equation (16) minimizes the $\phi_r(X_i)$ for every slice in addition to the $\phi_r(X)$ for each design point. The MSLHD is defined as be that minimizes the $\phi_{Mm}(X)$ in equation (16). [16] provide a general technique to find the best SLHDs using an exchange algorithm. Assume $p_0 = 0.5$ and that we wish to minimize a predefined metric $\phi(X)$ for the SLHD. The approach begins by selecting SLHD randomly and proceeds to analyze a series of new designs that are produced by perturbing the previous one:

1. Put X as the current SLHD. Create a z random variate that follows Uniform(0, 1).
2. If z less than or equal to p_0, choose a slice in X at random, then switch two randomly selected elements inside a randomly selected column of X. Indicate the new SLHD as X_{try} and proceed to Step (4).
3. In the case where z is bigger than p_0, select an X column at random and swap two elements randomly from a chosen $(\Pi_l, l, \dots, m)$ for this column. Proceed to Step (4) after designating the resultant SLHD as X_{try}.
4. Replace the current value X with X_{try} if X_{try} is smaller than X; if not, replace X with X_{try} with probability $\Pi = \exp\left\{\frac{X_{try} - X}{t}\right\}$, where t is a predefined parameter.
5. Until certain prerequisites for convergence are satisfied, repeat steps (1) through (4).
6. The ideal design is identified as the MSLHD that the algorithm finds to have the least $\phi(X)$ value.

4. Validating Kriging models

The KMs are built using specific assumptions. Measures to evaluate KM performance and ascertain whether the underlying assumptions were valid must therefore be used. We use two distinct measures to validate KMs in this study. The first measure is root relative squared error (RRSE) given by

$$\text{RRSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\bar{y} - y_i)^2}} \quad (19)$$

where $\bar{y}$ is the mean $\boldsymbol{y}$ [17]. The difference between expected and actual CC values is shown by the RRSE. The KM predictions will be reasonable as RRSE $\to 0$. The Mahalanobis distance (MD) is the second measure and it is given by

$$\text{MD} = \left(\mathbf{y}(\mathbf{x}^*) - \hat{\mathbf{y}}(\mathbf{x}^*)\right)^T \left(\sigma_{y^*}{}^2\right)^{-1} \left(\mathbf{y}(\mathbf{x}^*) - \hat{\mathbf{y}}(\mathbf{x}^*)\right) \quad (20)$$

The correlation between the CC outputs is taken into account by the MD. If the MD value is too high, a conflict between the KM and the CC will be displayed. Conditional on t $\mathbf{y}$, the MD distribution under KM assumptions is F distribution with m and $n - p + 1$ degrees of freedom [18].

5. Application

In this section, we apply KMs that are based on different types of LHDs to the Robot Arm (RM) function as an illustration of CC. [4] utilized the RM function as an example of applying quasi-regression techniques. Four segments make up the RM position, which is modeled by the RM function. For $i = 1, \dots, 4$, each segment with a length L_i and it is positioned at the angle ϑ_i, while the shoulder is fixed at the origin. RM 's function is provided by

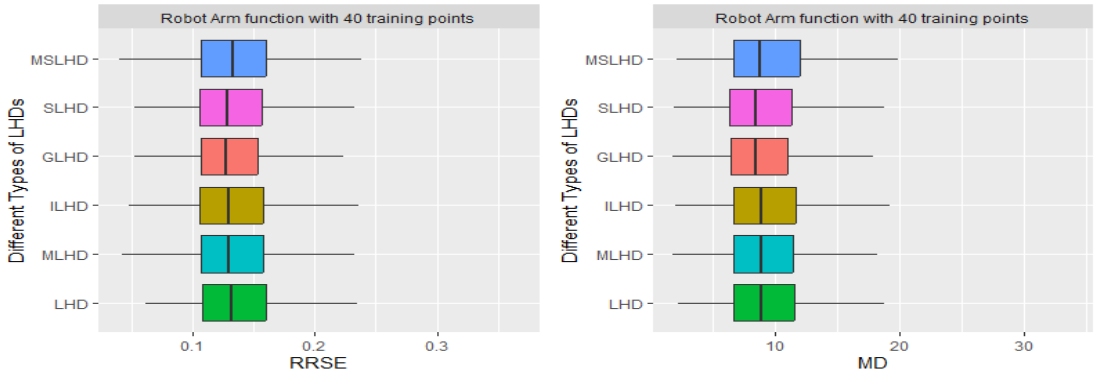
$$\eta(\mathbf{x}) = (u^2 + v^2)^{\frac{1}{2}} \qquad \textbf{(21)}$$

where

$$u = \sum_{i=1}^{4} L_i \cos\left(\sum_{j=1}^{i} \vartheta_j\right) \qquad and \quad v = \sum_{i=1}^{4} L_i \sin\left(\sum_{j=1}^{i} \vartheta_j\right)$$

The response of the RM is the distance between the RM 's origin and the RM 's end. There are eight input variables the in RM function: $\mathbf{x} = (\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, L_1, L_2, L_3, L_4)$ with ranges $\vartheta_i \in [0, 2\pi]$ for $i = 1, \dots, 4$. The four angles of the arm segments are $\vartheta_1, \vartheta_2, \vartheta_3$ and ϑ_4 with lengths L_1, L_2, L_3 and L_4 where $L_i \in [0,1]$ for $i = 1, \dots, 4$.

We considered different combinations of sample sizes for training points, $n = 5p, 10p, 15p$. For the test points, we used $m = 2p$. For training and test inputs, each of these combinations was duplicated 1000 times. Then, the RM outputs were obtained. The θ_i in equation (3) were estimated by the maximum estimated method (MLE) and then Conditional KMs were built. We then calculated the RMSE, equation (19), and the MD, equation (20). Figure 1 shows box-plots of 1000 RRSE and MD values of the KMs that are built using different types of LHDs with different $n = 5p = 40$ training points and a different $m = 2p = 16$ test points in every replication.



Box-plots of 1000 RRSE and MD values for KMs of the RM that built using different types of LHDs with 40 trainig points and 16 test points.

We can see from the left panel of Figure 1 that box-plots of the RRSE values for KMs. It can be seen that the RRSE values are small for KMs of the all types of LHDs. Therefore, the performance of all types of LHDs seems to be good. The right panel of Figure 1 displays box-plots of 1000 MD values for KMs. It can be seen that the values of the MD are similar for all the KMs that built on different types of LHDs. The MD values approach to their expected value 16. Hence, to see the best performance of the different types of LHDs, Table 1 shows the mean of the 1000 values of the RRSE and MD for all types of LHDs.

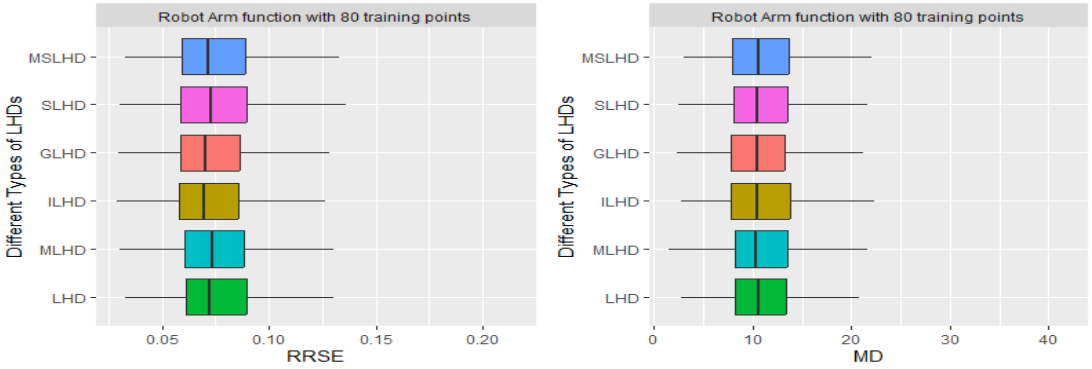
Table 1: The mean of the RRSE and MD values of the KMs for all types of LHDs with $n = 40$

	LHD	MLHD	ILHD	GLHD	SLHD	MSLHD
RRSE	0.141	0.138	0.156	0.125	0.146	**0.109**
MD	9.639	9.982	**10.197**	8.928	9.483	7.826

The best values are in bold font.

It can see be seen from Table 1that the MSLHD has the smallest RRSE mean value whereas ILHD has best MD mean value.

We increased the training points to be $n = 80$. Figure 2 presents box-plots the 1000 RRSE and MD values for KMs that are built using different types of LHDs with different $n = 5p = 80$ design points and a different set of $m = 2p = 16$ test points in each of the 1000 replication.



Box-plots of the 1000 RRSE and MD values for KMs of RM that built using different types of LHDs with 80 trainig points and 16 test points.

We can see from theleft panel of Figure 2 that the RRSE values for KMs are very small of the all types of LHDs. The performance of all types of LHDs has been improved. The right panel of Figure 2 presents box-plots of 1000 MD values for KMs. It can be seen that the values of the MD are also similar for KMs of the all types of LHDs. They are now more closer to their expected value 16. Table 2 shows the mean of the 1000 values of the RRSE for all types of LHDs.

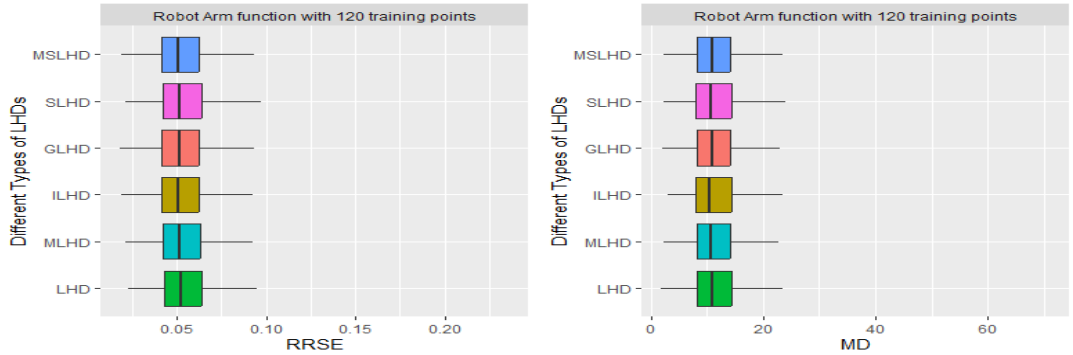
Table 2: The mean of the RRSE and MD values of the KMs for all types of LHDs with $n = 80$

	LHD	MLHD	ILHD	GLHD	SLHD	MSLHD
RRSE	0.081	0.076	0.086	0.070	0.082	**0.058**
MD	11.467	10.916	**11.753**	11.109	11.725	10.005

The best values are in bold font.

We can see from Table 2 that the MSLHD has also the smallest RRSE mean value and the ILHD has also best MD mean value.

We also increased the training points to be $n = 120$. Figure 3 presents box-plots of 1000 RRSE and MD values for KMs that built using different types of LHDs with different $n = 5p = 120$ design points and a different set of $m = 2p = 16$ test points in each of the 1000 replication.



Box-plots of 1000 RRSE and MD values for KMs of RM that built using different types of LHDs with 120 trainig points and 16 test points.

Now, we can see from left panel of Figure 3 that the RRSE values are now more smaller for KMs of the all types of LHDs. The performance of all types of LHDs are now very good as the values of RRSE approach to zero. The right panel of Figure 3 displays box-plots MD values for KMs. It can be seen that the values of the MD are also similar for KMs of the all types of LHDs. They are now more closer to their expected value 16. Table 3 shows the mean of the 1000 values of the RRSE for all types of LHDs.

Table 3: The mean of the RRSE and MD values of the KMs for all types of LHDs with $n = 120$

	LHD	MLHD	ILHD	GLHD	SLHD	MSLHD
RRSE	0.058	0.054	0.063	0.050	0.060	**0.042**
MD	12.346	11.200	**12.544**	11.522	12.042	10.888

The best values are in bold font.

Table 3 shows that the MSLHD also has the smallest RRSE mean value and the ILHD also has the best MD mean value.

We conclude that the RRSE values for KMs of all types of LHDs have become more smaller by adding more training points. In addition, based on RRSE values, the behavior of all KMs all types of LHDs are a quite similar, with a minor outperformance to MSLHD. In the other hand, the MD values for KMs of all types of LHDs have been improved and become more closer to their expected value by increasing the design points. Moreover, the performance of all KMs all types of LHDs are a also quite similar, with a minor outperformance to ILHD. This is maybe because the ILHD aims to reduced variance which results in having smaller MD values as the MD take into account the uncertainty of the KMs predictions.

6. Conclusion

In this study, we examined Kriging models that are used widely as statistical techniques for estimating and evaluating complex computer codes. The design of computer experiments is a crucial component in constructing Kriging models as it may affects the accuracy of predictions. Latin hypercube designs have become popular for building Kriging models. Thus, in this work, we evaluated the performance of many types of Latin hypercube designs by comparing the predictions of many Kriging models that are built using different types of Latin hypercube designs. This comparison is achieved via some measures that test the KM performance. We apply Kriging models

on a real example. The root relative squared error results of applying Kriging models on a real example shows that the performance of all types of Latin hypercube designs are a quite similar. The Maximin Sliced Latin hypercube design performs slightly better. However, the results of the Mahalanobis distance shows that the Improved Latin Hypercube Design is slightly outperformed other Latin hypercube designs as it reduces the variance in of Kriging models predictions.

References

[1] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, vol. 4, no. 4, pp. 409–423, 1989.

[2] R. Li and A. Sudjianto, "Analysis of computer experiments using penalized likelihood in Gaussian Kriging models," *Technometrics*, vol. 47, no. 2, pp. 111–120, 2005.

[3] I. Abdallah, C. Lataniotis, and B. Sudret, "Parametric hierarchical kriging for multi-fidelity aero-servo-elastic simulators—Application to extreme loads on wind turbines," *Probabilistic Engineering Mechanics*, vol. 55, pp. 67–77, 2019.

[4] J. P. Fentanes *et al.*, "Quasi-regression," *Journal of complexity*, vol. 17, no. 4, pp. 588–607, 2001.

[5] Y. H. Al-Taweel and N. Sadeek, "A comparison of different methods for building Bayesian kriging models," *Pakistan Journal of Statistics and Operation Research*, pp. 73–82, 2020.

[6] R. W. Al-Naser and Y. Al-Taweel, "Using Kriging models for approximating computer models and quantifying their uncertainty," in *AIP Conference Proceedings*, AIP Publishing, 2023.

[7] J. P. Kleijnen and E. Mehdad, "Multivariate versus univariate Kriging metamodels for multi-response simulation models," *European Journal of Operational Research*, vol. 236, no. 2, pp. 573–582, 2014.

[8] Y. Al-Taweel, "Uncertainty quantification of multivariate Gaussian process regression for approximating multivariate computer codes," *TWMS Journal of Applied and Engineering Mathematics*, 2024.

[9] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[10] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of statistical planning and inference*, vol. 43, no. 3, pp. 381–402, 1995.

[11] K.-T. Fang, R. Li, and A. Sudjianto, *Design and modeling for computer experiments*. CRC Press, 2010.

[12] B. Beachkofski and R. Grandhi, "Improved distributed hypercube sampling," in *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2002, p. 1274.

[13] T. W. Simpson, *A concept exploration method for product family design*. Georgia Institute of Technology, 1998.

[14] M. Liefvendahl and R. Stocki, "A study on algorithms for optimization of Latin hypercubes," *Journal of statistical planning and inference*, vol. 136, no. 9, pp. 3231–3247, 2006.

[15] P. Z. Qian, "Sliced Latin hypercube designs," *Journal of the American Statistical Association*, vol. 107, no. 497, pp. 393–399, 2012.

[16] S. Ba, W. R. Myers, and W. A. Brenneman, "Optimal sliced Latin hypercube designs," *Technometrics*, vol. 57, no. 4, pp. 479–487, 2015.

[17] A. M. Overstall and D. C. Woods, "Multivariate emulation of computer simulators: model selection and diagnostics with application to a humanitarian relief model," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 2016.

[18] L. S. Bastos and A. O'Hagan, "Diagnostics for Gaussian process emulators," *Technometrics*, vol. 51, no. 4, pp. 425–438, 2009.