2017

Second-order nearly orthogonal Latin hypercubes for exploring stochastic simulations

MacCalman, AD

MacCalman, Alex D., H. Vieira, and T. Lucas. "Second-order nearly orthogonal Latin hypercubes for exploring stochastic simulations." Journal of Simulation 11, no. 2 (2017): 137-150.

http://hdl.handle.net/10945/57854

Journal of Simulation (2017) 11, 137–150

www.palgrave.com/journals

Second-order nearly orthogonal Latin hypercubes for exploring stochastic simulations

AD MacCalman[1]*, H Vieira[2] and T Lucas[3]

[1]United States Military Academy, West Point, NY, USA; [2]Technological Institute of Aeronautics, São José dos Campos, Brazil; and [3]Naval Postgraduate School, Monterey, USA

This paper presents new Latin hypercube designs with minimal correlations between all main, quadratic, and two-way interaction effects for a full second-order model. These new designs facilitate exploratory analysis of stochastic simulation models in which there is considerable *a priori* uncertainty about the forms of the responses. We focus on understanding the underlying complexities of simulated systems by exploring the input variables' effects on the behavior of simulation responses. These new designs allow us to determine the driving factors, detect interactions between input variables, identify points of diminishing or increasing rates of return, and find thresholds or change points in localized areas. Our proposed designs enable analysts to fit many diverse metamodels to multiple outputs with a single set of runs. Creating these designs is computationally intensive; therefore, several have been cataloged and made available online to experimenters.

Journal of Simulation (2017) **11**(2), 137–150. doi: 10.1057/jos.2016.8; published online 3 June 2016

Keywords: simulation; design of experiments; genetic algorithms; Latin hypercube; nearly orthogonal designs; space-filling designs

1. Introduction

The field of statistical design of experiments (DOE) has applications in all areas of research. Scientists use DOE to help them understand how the world works through observation in the areas of behavioral, social, and natural sciences, engineering, medicine, finance; manufacturing transportation, and many others. DOE allows us to efficiently learn about and characterize the complex nature of our world. As computers become progressively more powerful and affordable, they have become an increasingly valuable instrument for experimentation (Lucas *et al*, 2015).

Computer simulations provide important insights in the areas mentioned above when physical experimentation is not possible or cost effective. Simulations are simplified representations of reality, programmed on a computer, which are regularly used to find optimal settings, make predictions, develop an understanding of a particular simulation model or system, and discover robust decisions or policies (see Sacks *et al* (1989) and Kleijnen *et al* (2005)). Simulation is a common technique for studying prospective systems or scenarios. There are countless discrete-event simulations that are highly stochastic and likely involve dozens (or even many more) input factors (Law, 2007). One of the largest users of simulation is the United States Department of Defense (DoD), which has invested many billions of dollars in modeling and simulation. DoD uses simulations to help test war plans, decide what equipment to acquire, train personnel, study doctrine, and potential operational concepts, and much more (see www.msco.mil and National Research Council (2006)). Indeed, the United States Navy lists over 900 models and simulations in its online modeling repository (nmso.navy.mil/NavyMSRR.aspx), many of which are stochastic. Simulation's utility is not limited to defense—stochastic simulation is widely used to model prospective systems in industry, academia, scientific research, and elsewhere. Simulation assists decision makers in understanding and reasoning about extremely complex systems and processes (Bankes, 1993).

A common objective when analyzing these simulations is to identify the factors (ie, input variables) that significantly affect the response (ie, output variable) and, for those that do, determine the nature of the relationship. Several statistical methods can fit a functional model of the simulation response, such as parametric polynomial response surface approximation, neural networks, frequency domain methods, and Gaussian Processes (GP). Such a model is commonly referred to as a metamodel (model of a model) or a surrogate model. Barton (1998) said that the most common way to quantify the relationship between a stochastic simulation's input factors and a response is to fit a parametric polynomial function using the linear regression modeling method. These polynomial functions have readily interpretable parameter coefficients that indicate the rate of change of a given factor while all other factors are held constant. The magnitude and sign of the main-effect coefficients, along with their higher-order-term coefficients, express the nature of the input factor's effect on the response. These coefficients provide insight into which factors matter, whether they increase or decrease the response, whether there are synergistic effects between two or more factors, or if a factor yields diminishing or increasing rates

**Correspondence: AD MacCalman, United States Military Academy, Building 742, West Point, NY 10996, USA.*
E-mail: alex.maccalman@gmail.com

of return. Other simulation objectives may include finding optimal settings and making predictions. For an in-depth review of the classic and modern designs used for simulation experiments, see Kleijnen (2005) and Kleijnen (2008a).

Another metamodeling technique is the GP model. GPs are often used for predicting the response of deterministic simulations. The GP constructs predictions that are a weighted combination of the observed values depending on an underlying correlation structure. GP models have primarily been used to predict deterministic simulations, but more recently have been applied to stochastic simulations. Beers and Kleijnen (2003) demonstrate how GP performs better than the polynomial model at predicting random simulation output for a given test case. They also point out that the polynomial model is an attractive statistical method when looking for explanations, screening factors, or validating simulation models. Applying GP to stochastic simulations is an ongoing area of research (Kleijnen, 2008b; Ankenman *et al*, 2010). The type of design and functional model we use is determined by the goal of the simulation study. Since our goal is to understand behavior, we often use polynomial models to approximate the functional form of the input/output relationship over a range of inputs because they can describe the nature of the simulation complexities with parametric coefficients that are easier to interpret; in this paper, we propose new designs that meet this goal, while simultaneously exploring the entire design region to find interesting behavior.

A good metamodel is one that makes parsimonious use of the input factors and is simple to understand (Sanchez, 2008). The most common polynomial model used to describe response surfaces is the second-order model, which includes main effect, quadratic, and two-way interaction terms. These second-order models provide a rich variety of functional forms that can represent surfaces with global or local maximums and minimums, rising or stationary ridges, and saddles (Myers *et al*, 2009). The terms within the second-order model provide valuable insights on important interactions (eg, synergies) and trends such as diminishing or increasing rates of change. Having the ability to quantify interactions and non-linear trends can provide valuable insights to the systems analyst. Siggelkow (2002) studied the implications of misperceiving interactions between system activities and concludes that uncertainty about activities that complement each other are more costly than uncertainties about activities that substitute each other.

The metamodels we can fit—and hence the insights that we can glean—depend critically on the design. For example, we cannot identify a quadratic response for a quantitative input variable from a two-level design. In such a case, inferences based on the implicit assumption of linearity may be erroneous. The error that occurs when our assumed model is wrong (typically due to underfitting) is known as ‘model bias’. We desire designs less susceptible to model bias and with the ability to detect it when it occurs. Thus, we prefer designs that allow us to fit a breadth of metamodels. Another difficulty that experimenters face while fitting polynomial metamodels is correlations among the inputs. When two inputs are highly correlated, it is difficult (or impossible) to distinguish their effects on the response. Orthogonal designs overcome this problem, and are thus desired. An additional challenge occurs when there might be localized effects, such as a threshold or changepoint. To increase the odds of detecting localized effects, we favor designs that sample throughout the experimental region. Such designs are called *space-filling*. This paper presents a genetic algorithm (GA) for generating designs that allow experimenters to fit a full second-order polynomial with nearly uncorrelated coefficient estimates between all terms, while adequately exploring the interior of the design region and allowing higher-order terms (ie, above quadratic) on a modest number of factors.

There are numerous applications of GAs that construct computer-generated designs. Some of these applications focus on optimizing the alphabetical criterion that find good coefficient estimates or predicted response estimates for a specified model. Techniques to construct D-optimal designs using GAs have been shown to outperform other traditional optimal procedures that require the search space to fit a particular structure (Heredia-Langner *et al*, 2003). GAs were used to find optimal designs that are robust across a specified number of different models (Heredia-Langner *et al*, 2004). GAs have constructed designs involving mixture and process factors that include control and noise factors (Goldfarb *et al*, 2005). Other GAs have focused on optimizing space-filling metrics. Morris and Mitchell (1995) used simulating annealing to find designs with a distance metric for the fitness function. Jin *et al* (2005) developed an enhanced stochastic evolutionary algorithm with efficient methods for evaluating optimality criteria. Joseph and Hung (2008) proposed a modification of the simulating annealing algorithm by using a ‘smart swap’ method, rather than randomly swapping design points. In addition, they used a weighted average of a distance metric and the average column correlation among the main effect terms as their fitness function. Moon *et al* (2011) developed algorithms for generating maximin Latin hypercube and orthogonal designs that show improvements to the existing algorithms under a variety of criteria. These aforementioned algorithms construct designs that optimize a number of different criteria. Our algorithm has a different purpose; it constructs space-filling designs that minimize the correlations between all terms in a full quadratic model.

The article is organized into six sections. Section 2 provides general background and explains the benefits of flexible designs for uncertain response surfaces. Section 3 explains the GA we use to construct our proposed designs. Section 4 compares the performance of our designs against eight other commonly used designs for complex responses. Section 5 catalogs the designs, while the final section summarizes the results.

2. Background

This section describes the experimental setting, discusses how parametric polynomial metamodels and non-parametric partition trees are used to characterize complex simulation behavior, reviews some of the common designs used to estimate second-order models, defines Latin hypercubes, and explains the metrics we use to assess our designs.

2.1. The experimental setting

We consider experimental settings involving a stochastic, computer-based simulation in which users specify the input values and analyze the outputs. The *design matrix* is the complete specification of input settings for each factor over a set of runs. We assume the design is specified before the experiments being conducted and that we will not perform sequential designs. The simulation being investigated contains k continuous variables that we wish to vary in n computational experiments over a k dimensional hyperrectangle. We denote the $n \times k$ design matrix as $\boldsymbol{X}$, where row i of $\boldsymbol{X}$ corresponds to the ith experimental run, and column j represents the jth input variable. Thus, X_i^j is the value the experimenter sets for factor j in run i. We further denote the jth column of $\boldsymbol{X}$ as X^j and the ith row as X_i. Finally, let Y_i be an outcome generated by the ith experiment and $\boldsymbol{Y}$ be the vector formed by all Y_i.

2.2. Characterizing complex behavior with polynomial metamodels and partition trees

If we assume the response surface is a linear combination of the input factors, then the metamodel has the following form:

$$Y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \epsilon, \qquad (1)$$

where β_0 is the intercept term, and β_j is the coefficient of the X^j term and represents a factor's rate of change or effect on the output response Y when all other factors are held constant. We will refer to (1) as a main-effects metamodel. During an exploration study of a simulation, these β_j coefficients provide the insights that help describe the response behavior. The error term ϵ represents other sources of variation not accounted for by the factors. This could be because of a lack of fit or the noise induced by pseudorandom variables in the simulation.

Complicated simulations are rarely well represented by a main-effects response surface. A quadratic second-order model is often used to model real-world problems and it has the following form:

$$Y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \sum_{j=1}^{k} \beta_{jj} (X^j)^2 + \sum_{i=1}^{k-1} \sum_{j>i}^{k} \beta_{ij} X^i X^j + \epsilon, \qquad (2)$$

where $(X^j)^2$ is the quadratic term for the jth input factor, β_{jj} is its coefficient, X^iX^j is the two-way interaction between the ith and jth input factors, and β_{ij} is the coefficient of X^iX^j. We will refer to (2) as a second-order or full quadratic metamodel. The quadratic term's coefficient describes a non-linear relationship that may indicate a factor's diminishing or increasing rate of change on the response. The coefficient of an interaction term reveals a factor effect's dependence on the setting or level of another factor. In order to estimate higher-order models, we must expand the design matrix to include additional columns that represent the higher-order terms. Moreover, the design must be such that all of the parameters are estimable.

Least squares estimation is the most common method to estimate the β coefficients. The stability of these estimates depends on the correlations among input factors within the design matrix (Ryan, 2007). Therefore, we desire designs that have minimal correlation among all terms in our metamodel. The number of terms needed for a full second-order model increases according to the following expression:

$$p = 1 + 2k + k\left(\frac{k-1}{2}\right), \qquad (3)$$

where p is the number of terms and k is the number of factors. However, in practice, most fits require far fewer terms. Using an algorithm to search for a design that minimizes the maximum absolute pairwise correlation for a full quadratic model is significantly more difficult than for a main effects only model or a main effects with the quadratic terms added to it; the algorithm must evaluate each pairwise comparison For example, there are as many $\begin{pmatrix} p \\ 2 \end{pmatrix}$ pairwise correlation comparisons for a 15-factor full quadratic model as there are for a 135-factor main effects only model (9045 comparisons).

A second-order metamodel approximates a smooth, non-linear response surface, but cannot account for a three-way interaction, a cubic term, a discontinuous step function that may exist in the output, or many other relationships. A three-way interaction can describe the synergistic effect of three factors, while a cubic term can indicate the presence of an inflection point. Step functions, or thresholds, are common when dealing with complicated response surfaces. Identifying the presence of a step function can lead to important insights when analyzing a system. For example, during the test and evaluation of the maximum allowable weight of a cargo parachute, the rate of decent may increase linearly or non-linearly as the weight increases up until a weight threshold. Once we exceed this threshold, the parachute will collapse and increase or step up the rate of decent by a significant amount. Identifying the weight threshold for a cargo parachute is therefore critical for those involved with its use. For an example of threshold detection using a Latin hypercube design in software testing, see Cioppa and Lucas (2007). In order to account for an existing threshold, we can include a step function in the metamodel with the following form:

$$Y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \sum_{j=1}^{k} \beta_{jj} (X^j)^2 + \sum_{i=1}^{k-1} \sum_{j>i}^{k} \beta_{ij} X^i X^j + \beta_s I(X^s > threshold) + \epsilon, \qquad (4)$$

where $I(a)$ is an indicator function that has a value of 1 if a is true, and 0 otherwise. For example, if the value of X^s exceeds the threshold, the response will step up in the amount of β_s. Space-filling designs are particularly useful for identifying the presence of step functions.

Partition trees are an excellent way to identify thresholds. For continuous variables, a partition tree finds the optimal split in a data set where the distance between the two group means is the greatest (Sall, 2007). Each split occurs at a factor level that separates the data into two groups, one below and one above the split level. Figure 1 shows a second-order polynomial with a step function and a partition tree that finds where the discontinuous step function occurs.

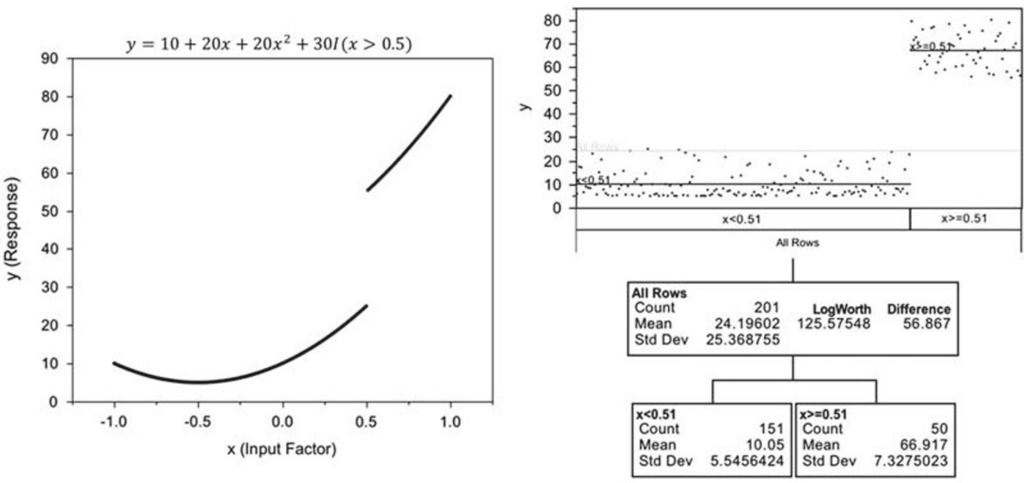
Partition trees, when used in conjunction with regression analysis, can be powerful tools for finding metamodels involving step functions. If we find a split that explains the data's variability with a partition tree, then we can create a new factor (an indicator variable) that has value 0 if x is less than the threshold identified by the partition tree, and 1 otherwise. The indicator variable becomes a term in the metamodel that may be significant and help explain some of the variance. In order to take full advantage of partition trees, we must use space-filling designs to efficiently explore the interior of the design region in order to find interesting thresholds.

2.3. Common designs used to estimate second-order models

The most frequently used design for the full second-order model is the Central Composite Design (CCD) (Myers *et al*, 2009). The CCD has numerous applications in response surface methodology (Box and Draper, 1987). The full factorial CDD has 2^k corner points, $2k$ axial points that normally extend beyond the corner points, and a selected number of center points. For the calculations in this paper, we assume only one center point is used. The Faced CCD (FCCD) has the axial points positioned at the face of the design region so that the corner points are not collapsed inside the established factor ranges. The FCCD is used when the experimental region cannot extend beyond the corner points. Another popular classic second-order design is the Box-Behnken Design (BBH) (Box and Behnken, 1960). Recently Jones and Nachtsheim (2011) developed an algorithm to generate large three-level designs (upto 30 factors are available online) that 'provides estimates of main effects that are unbiased by any second order effect'. Moreover, they quantify the correlations between quadratic and interaction effects.

Computer-generated optimal designs are often used when traditional designs are not applicable. For example, when there is a non-rectangular experimental region that has factor constraints, qualitative factors, or if we want to fit a non-standard model that excludes a subset of quadratics or interactions (Myers *et al*, 2009). The computer typically generates a design by using a point exchange algorithm that maximizes a specified criterion for a given model, usually first, second, or higher order. Furthermore, a specified form of the covariance matrix is often assumed. The types of criterion used are known as the alphabetic optimality criteria, which typically minimize some function of the covariance matrix of the coefficient estimates. For example, a D-optimal design minimizes the determinant of the covariance matrix, while the I-Optimal design minimizes the average prediction variance, both for a prespecified model (usually a main-effects model with constant variance). For a detailed discussion of optimal design criteria, see Atkinson *et al* (2007). It is worth noting that despite the 'optimal' that appears in these design names, many are derived heuristically and may not, in fact, be best at achieving their stated optimality criteria.

Computer-generated optimal designs can work very well when we know the model form and error structure; perhaps because we are experimenting over a small experimental region where simple models suffice. Because we are focused on understanding stochastic simulations over broad experimental regions where the response mean and variance can be much more complex, we cannot assume the model form and therefore



All Rows			
Count	201	LogWorth	Difference
Mean	24.19602	125.57548	56.867
Std Dev	25.368755		

x<0.51	
Count	151
Mean	10.05
Std Dev	5.5456424

x>=0.51	
Count	50
Mean	66.917
Std Dev	7.3275023

Figure 1 The chart on the left shows a quadratic polynomial with a step function, *I(a)*, that represents an indicator function with a value of 1 if *a* is true and 0 otherwise. The chart on the right shows the split in the data where the difference in the response mean for each group is the greatest.

do not prefer to use so-called optimal designs for exploratory analysis.

2.4. Latin Hypercubes and other space-filling designs

The traditional second-order designs mentioned thus far typically experiment only at the corners, faces, and center of the design space. While not constrained to, the optimal designs often generate such designs. Indeed, for the examples later in this paper, all of the JMP generated optimal designs for a full second-order model only sample at the corners, faces, and center of the design space. Space-filling designs are better suited for identifying unknown response surfaces where multiple complex forms and localized effects are possible (Myers *et al*, 2009). Some of the popular space-filling designs include the sphere-packing, uniform, maximum entropy, and maximin distance Latin hypercube designs. Sphere-packing designs maximize the minimum distance between pairs of design points (Johnson *et al*, 1990). Uniform designs scatter the design points as uniformly as possible (Fang, 1980). The maximum entropy designs maximize the information contained in the distribution of a data set (Shewry and Wynn, 1987). The maximum distance Latin hypercube designs attempt to maximize the minimum distance between design points while requiring even spacing for each factor level (Morris and Mitchell, 1995). All of these designs are available in the SAS Institute's JMP software (see www.jmp.com). While the LHS notation is generally used more broadly in Latin hypercube sampling, in this paper we are referring to the maximin distance Latin hypercubes JMP creates.

As our designs are based on the Latin hypercube family, we detail how they are created. McKay *et al* (1979) first proposed the Latin hypercube design and described it as follows: For each input variable X^j, 'all portions of its distribution [are] represented by input values [by dividing its range into] n strata of equal marginal probability with $1/n$, and [sampling] once from each stratum' (McKay *et al*, 1979, p. 56). Following Koehler and Owen's (1996) notation, the ith element in the jth column, X_i^j, is determined by $X_i^j = F_j^{-1}((\pi_j(i) - U_{ij})/n)$, for $i = 1,\ldots,n$ and $j = 1,\ldots,k$, where $\pi_j(1),\ldots,\pi_j(n)$ is one of the $n!$ possible random permutations of $1,\ldots,n$ in which all $n!$ permutations are equally likely. F_j, for $j = 1,\ldots k$, are continuous and invertible distribution functions. U_{ij}, for $i = 1,\ldots,n$ and $j = 1,\ldots,k$ are independent and identically distributed uniform [0,1] random variables.

Many analysts choose F_j to be a uniform distribution and take a fixed value in each stratum (eg, the median). In this situation, the design points all fall on a lattice in k-space. In such a case, creating a Latin hypercube corresponds to independently generating k random permutations of the first n natural numbers and appropriately scaling the columns to cover the factors' ranges. This uniform spacing guarantees that for each factor j, assuming its scaled range is $[a,b]$, $\forall x \in [a, b]$, $\max_{i=1,\ldots,n} \left|x - X_i^j\right| \leqslant (b-a)/(2(n-1))$. Therefore, if a response to a factor has a sharp threshold, these designs will closely bracket it. Moreover, with an LH, at the extreme, an analyst could fit an $n-1$ degree polynomial to a single input variable.

There are several reasons why we choose to base our designs on the Latin hypercube family. LHs have been extensively used within the computer experiments literature (Santner *et al*, 2003). Santner *et al* (2003) believe that LHs are popular because they are easily obtainable, for example, they are available in many simulation software packages, and they have no projection redundancy when their design points are projected onto any single dimension. Furthermore, they have few restrictions on n and k. Beers and Kleijnen (2004) indicate that the Latin hypercube design is the simplest and most popular design used for the GP model. Challenor (2013) uses Latin hypercubes to validate GP models. In addition, the resultant output data allow analysts to fit many different diverse models to multiple univariate response outputs from a single experimental set (Sanchez *et al*, 2012).

Unfortunately, a given Latin hypercube need not have good correlation or space-filling properties. To address this, a number of researchers have developed algorithms to reduce or eliminate correlations among columns of a Latin hypercube and improve on their space-filling properties (Florian, 1992; Owen, 1994; Ye, 1998; Steinberg and Lin, 2006; Cioppa and Lucas, 2007; Joseph and Hung, 2008; Pang *et al*, 2009; Sun *et al*, 2009; Moon *et al*, 2011; Vieira, *et al*, 2011; Hernandez *et al*, 2012; Vieira, *et al*, 2013; Yang *et al*, 2013). A Latin hypercube with no correlation between any of its input variables is an orthogonal Latin hypercube (OLH). A nearly orthogonal Latin hypercube (NOLH) is defined as a Latin hypercube with a maximum absolute pairwise correlation no greater than 0.05 between any two input variables (Hernandez *et al*, 2012). Although this criterion is somewhat arbitrary, designs meeting it suffer minimal adverse multicollinearity effects.

Sun *et al* (2009) indicate that when second-order effects are present, a Latin hypercube must satisfy the following two properties: (a) each main effect term in a design must be orthogonal to each other; and (b) each main effect term in a design must be orthogonal to each of the quadratics and two-way interaction terms. Yang *et al* (2013) develop sliced Latin hypercubes with these properties in situations in which the number of factors is a power of 2. However, there is a third property that the above do not consider: (c) each quadratic term is orthogonal to other quadratic term, each two-way interaction term is orthogonal to each other, and each quadratic term is orthogonal to each two-way interaction term. In order to properly analyze a full second-order model, a design must have all three properties. Creating space-filling designs with properties (a), (b), and (c) is much more harder than if we limited the properties to just (a) and (b); this is due to the quadratic growth of the number two-way interaction terms as k increases. The many research efforts mentioned above create Latin hypercubes that satisfy properties (a) and also sometimes (b)—with some relaxing the orthogonal requirement to be nearly orthogonal. Currently, there are no orthogonal or nearly NOLH designs in the literature that satisfy properties (a), (b), and (c) simultaneously; the designs proposed in this paper are nearly NOLH designs with all three properties.

2.5. Design assessment metrics

Our algorithm focuses on creating LH designs that minimize the pairwise correlations for the coefficient estimates a full second-order model (ie, Equation (2)). Thus, we need to control the correlations among all pairs of columns in the second-order regression matrix, which we will denote as $\mathbf{Z}$. The first k columns in $\mathbf{Z}$ are the design matrix $\boldsymbol{X}$, for the main effects terms. The quadratic terms in $\mathbf{Z}$ require an additional k columns that are the squares of the columns in the design matrix. Finally, the last $k(k-1)/2$ columns consist of the element-by-element product of the columns of $\boldsymbol{X}$, thereby enabling the estimation of the two-way interactions. Therefore, $\mathbf{Z}$ is the $n\times(2k+k(k-1)/2)$ regression matrix needed to estimate the βs in Equation (2)—with the intercept estimated by augmenting $\mathbf{Z}$ with a column of ones. The correlation coefficient between any two vector columns, Z^i and $\mathbf{Z}^j$, in regression matrix $\mathbf{Z}$ is:

$$\rho_{ij}=\frac{\sum_{b=1}^{n}\left[\left(Z_b^i-\bar{z}^i\right)\left(Z_b^j-\bar{z}^j\right)\right]}{\sqrt{\sum_{b=1}^{n}\left(Z_b^i-\bar{z}^i\right)^2\sum_{b=1}^{n}\left(Z_b^j-\bar{z}^j\right)^2}}, \tag{5}$$

where $\bar{z}^i$ and $\bar{z}^j$ are the mean of the ith and jth columns in $\mathbf{Z}$. Ideally, we would like a design in which $\rho_{ij}=0$ for $i=1,\ldots,2k+k(k-1)/2$ and $j=1,\ldots,2k+k(k-1)/2$, with $j\neq i$. We quantify the degree of non-orthogonality by calculating the maximum absolute pairwise *(map)* correlation between the columns of regression matrix $\mathbf{Z}$:

$$\rho_{map(Z)}=\max\left\{\left|\rho_{ij}\right|,\ \forall(i\neq j)\right\}. \tag{6}$$

A design with a $\rho_{map(Z)}$ near zero will minimize confounding factors and result in nearly independent and more precise coefficient estimates in the second-order regression metamodel, as well as enhance the performance of partition tree analysis (Kim and Loh, 2003). Other authors (eg, Owen (1994) and Joseph and Hung (2008)) minimize the sum (or average) of the squares of the pairwise correlations (for a first-order model). We prefer to minimize $\rho_{map(Z)}$ because it bounds the worst-case correlation. A design can have low average correlation, but a few unacceptable values—especially when there are a large number of pairwise correlations, as is the case when fitting second-order models to a model with numerous factors.

Low correlation, even orthogonality, does not guarantee good space-filling. The modified L_2 discrepancy (ML_2) is a space-filling measure often used to assess how well a design covers the entire design region; the smaller the value, the better a design's space-filling property (Hickernell, 1998). The ML_2 is a modified version of the L_∞ discrepancy. Fang *et al* (2000, p. 238) state that discrepancy is a measure of uniformity and the L_∞ 'is probably the most commonly used measurement for discrepancy … and has been universally accepted in quasi-Monte-Carlo methods and number theoretic methods'. The L_∞ is equivalent to the Kolmogorov-Smirnov statistic in goodness of fit testing between the empirical distribution function defined by the n design points and the uniform distribution function over the experimental region. Because the L_∞ discrepancy is too computationally expensive for high dimensions, this article uses the ML_2 (with the designs normalized to [0,1] in each dimension) to assess a design's space-filling property. The ML_2 metric is calculated using the following expression:

$$ML_2=\left(\frac{4}{3}\right)^k-\frac{2^{1-k}}{n}\sum_{d=1}^{n}\prod_{i=1}^{k}\left(3-x_{di}^2\right)+\frac{1}{n^2}\sum_{d=1}^{n}\sum_{j=1}^{n}\prod_{i=1}^{k}\left[2-\max\left(x_{di},\,x_{ji}\right)\right]. \tag{7}$$

A design with a smaller ML_2 is preferred. By its construction, if a design has a low ML_2, then it tends to have good space-filling properties. In Section 4, we use $\rho_{map(Z)}$ and ML_2 to compare our second-order NOLH designs with the eight traditional, optimal, and space-filling designs mentioned in Sections 2.3 and 2.4.

3. Genetic algorithm (GA)

Our goal is to minimize Equation (6) while using a Latin hypercube design. The algorithm creates designs that minimize the maximum absolute pairwise correlation of the columns in $\mathbf{Z}$ while sampling once within each of n equally spaced strata for each of the k factors. Solving this is challenging since the objective function is non-linear and not differentiable everywhere.

To construct the second-order NOLH algorithm, we utilize the principles of GAs (Holland, 1975). GAs are different from traditional optimization methods. They are meta-heuristics that do not guarantee an optimal solution, but they often find attractive solutions in complicated environments where linear and non-linear mathematical programming cannot (Michalewicz and Fogel, 2010). In our domain, an optimal solution is a Latin hypercube design that has an orthogonal regression matrix, $\mathbf{Z}$, which includes the quadratics and two-way interactions. Unfortunately, these designs have not yet been found or proven to exist for an arbitrary number of design points, n. Because our goal is to find good (nearly orthogonal) designs and not necessarily optimal ones, we choose to use GAs to find attractive design solutions with a minimal $\rho_{map(Z)}$ and good space-filling properties.

A GA uses random selection as a guide to finding better-performing solutions from a population of candidate solutions. The algorithm iteratively generates new populations using attractive solution characteristics from the previous generation; the intent is to evolve solutions that perform better with each new generation. The user measures performance using a predefined fitness value that has a functional form with no limitations. Our algorithm starts with one randomly generated, centered design column and a population of randomly generated, centered candidate columns. We center a column (around zero) by subtracting its mean from each of the entries. If we did not center the columns then they would be highly correlated with their quadratic term. Our GA solution is that column which, when added to the existing design, results in the lowest $\rho_{map(Z)}$. The solution for the other GAs mentioned in Section 2 is a complete design for a given n and k, where the number of possible solutions is $(n!)^k$. Because our GA solution is only one column that is added to the design incrementally, we reduce the search space significantly by limiting the number of solutions to $(n!)$.

Within a GA, an operator is a set of instructions that performs actions on solutions to evolve them into better performers (Goldberg, 1989). Our algorithm uses two types of operators to modify the structure of a given column. The swap operator swaps or exchanges a pair of values within the jth column at random positions. The jiggle operator adds a small value to the element in a randomly selected row of the column, while subtracting the same amount from a different randomly selected row. We define the term jiggle as a slight perturbation to two of the values within a column. The jiggle operator does not perturb the lowest and highest values in X^j, so that the desired experimental ranges do not change. The perturbation amount is selected from a uniform distribution. The upper and lower bounds of the uniform distribution ensure that the values remain in their original interval. For example, if the upper and lower bounds are set to $+/-0.5$, then the jiggle operator can never perturb a value set to 2 to be greater than 2.5 or less than 1.5. This preserves the idea that a Latin hypercube samples once in each interval of the range. These bounds on the jiggle operation also preserve the design's space-filling properties. In addition, subtracting the same amount from one value that we add to another preserves the column's mean.

In order to create a new population of column solutions, the algorithm selects attractive columns from the old population and creates new columns by modifying the selected column's structure using the swap or jiggle operators. Each column in the old population has a probability for selection that is proportional to its performance. In this way, the algorithm allows any candidate to be selected, but places a higher selection probability on candidates that perform well. A column's performance is measured by its fitness value. The fitness value is defined as the complement of the maximum absolute pairwise correlation $(1-\rho_{map(Z)})$, so that the higher the fitness value, the higher the selection probability. In order to increase the chance of selecting the columns with a higher fitness, we redefine the fitness value by using a linear ranking defined as:

$$fitness(t) = \text{Min} + [\text{Max} - \text{Min}][kstar - t]/]n - 1], \tag{8}$$

where *fitness(t)* is the redefined fitness value of the tth column; Max and Min are the maximum and minimum of the original fitness values from the old population $(1-\rho_{map(Z)})$, respectively; *kstar* is the number of columns in the old population; and t is the rank-ordered index of the original fitness values (Vieira, 2008).

The search for a candidate column solution with the lowest $\rho_{map(Z)}$, is highly dependent on the randomly generated, initial population. As the number of generations increase, the best-performing column converges to a local solution. In order to increase the chance of finding a low $\rho_{map(Z)}$,the algorithm performs a limited number of exploration trials, each with its own initial population and a predefined number of generations. The algorithm then exploits the population with the best-performing column solution by continuing for a set number of additional generations. Each of the exploration and exploitation generations utilizes the swap operator only. After all the swap generations are complete, the best-performing column is appended onto $\boldsymbol{X}$ and the algorithm searches for the next column. Once $\boldsymbol{X}$ contains the designed number of k columns, the algorithm then performs a set number of generations using the jiggle operator on each column in $\boldsymbol{X}$.

The choice of n and k depends on the experimental conditions. The experimenter chooses k based on the objectives of a study. A large k implies the need for a larger n. Because of experimental constraints imposed by time and resources, a design may need to be as small as possible for a given k. To find the smallest n for a given k, we performed several iterations of the algorithm by bracketing n within a chosen range until we found the lowest $\rho_{map(Z)}$. For ease of use, we cataloged designs with the smallest n that obeys our definition of nearly orthogonality ($\rho_{map(Z)} < 0.05$) for k from 3 to 12. Now that we defined the solutions, operators, selection probabilities, and fitness values of our GA, we present the algorithm steps used to find the second-order NOLH designs for a given n and k:

Step 1: Start with an $n \times 1$ design matrix, $\boldsymbol{X}$ with n design points and one randomly generated Latin hypercube column (ie, a random permutation of the first n natural numbers). Center the column at zero; thus, the extreme values in the column are $\pm(n-1)/2$.

Step 2: Perform a set number of exploration trials, defined as *numTrials*. Each trial has its own initial population of candidate columns with a size defined as *popSize* and a set number of generations defined as *numExploreGen*. For each generation, calculate the candidate column's fitness within the population. Copy a portion of the best-performing columns into the next population. Select a column randomly, using a cumulative distribution function based on each column's relative fitness. Create a pool of new columns by performing the swap operator on the selected column a random number of times. Add the best-performing column from the pool into the next population. The number of swap operations performed is random and depends on the number of design points, n; this number is drawn from a uniform $[1,s]$ distribution, where $s = \lfloor swapPortion \times n \rfloor$ and *swapPortion* is set to a value between 0 and 1. The size of each pool is defined as *poolSize*. Continue to create new columns in this same manner; the generation is complete when the next population is full of newly created columns.

Step 3: Select the population among the exploration trials that contains the best-performing column. Exploit the selected population by performing additional swap generations (as described in Step 2). The number of exploitation swap generations is defined as *numExploitGen*.

Step 4: At the completion of the exploitation swap generations, add the best-performing column to the design matrix, $\boldsymbol{X}$. Repeat Steps 2–4 until the designated number of columns, k, is in $\boldsymbol{X}$.

Step 5: Select a column, X^j, in $\boldsymbol{X}$. Create an initial population of new columns by performing the jiggle operator on the X^j. The jiggle operator creates new columns by adding a small amount, ω, where $\omega = U - 0.5$, to a randomly selected $X^j_{i_1}$, while subtracting the same amount to another randomly selected $X^j_{i_2}$; U is a uniform [0,1] random variable and the rows with values $\pm\ (n-1)/2$ are excluded from consideration. In order to preserve the design's space-filling property, the algorithm bounds the values to be within $+/-$ a distance from the original value before any jiggle operation. We define θ^j_i to be the original value from X^j_i before Step 5. The bounded distance is defined as *halfWidth* and is the maximum distance a value may be perturbed away from θ^j_i. Setting *halfWidth* $\leqslant 0.5$ will ensure the values remain within each of the n equally spaced strata in X^j_i. Setting *halfWidth* too high will degrade the design's space-filling property, but improve the search for lower correlations. Ensure that the pair of new values, $X^j_i + \omega$ or $X^j_i - \omega$, is within the range $[X^j_i - \omega < \theta^j_i < X^j_i + \omega]$. Perform the jiggle operation a random number of times; the number of times is drawn from a uniform $[1,g]$ distribution, where $g = \lfloor jigglePortion \times n \rfloor$ and *jigglePortion* is set to a value between 0 and 1. Continue to create new columns in this same manner until the size of the population is equal to *popSize*.

Step 6: Perform a set number of jiggle generations, defined as *numJigGen*, on the selected column, X^j. For each generation, calculate each candidate column's fitness within the population; copy a portion of the best-performing columns into the next population. Select a column randomly using a cumulative distribution function based on each column's relative fitness. Create a pool of new columns by performing the jiggle operator on the selected column a random number of times (as described in Step 5). Add the best-performing column from the pool into the next population. A jiggle generation is complete when the next population contains *popSize* new columns. After *numJigGen* amount of generations, if the best-performing column improves the design's ρ_{map}, add it to the $\boldsymbol{X}$. If not, add the originally removed X^j back to $\boldsymbol{X}$. Repeat Steps 5 and 6 for each column in $\boldsymbol{X}$.

Step 7: Repeat Steps 5 and 6 a designated number of times, defined as *jigglePasses*.

GAs can take a while to solve. The algorithm's time depends highly on n and k as well as the algorithm parameters (denoted in italics in the above steps). To find the appropriate algorithm parameter settings, we performed many thousands of experiments on the algorithm using designs created by our GA. After extensive experiments and parameter tuning we developed a unique GA that efficiently constructs our designs. Then, given the most efficient algorithm parameters we found, by systematic, brute force, trial and error, the minimum n that produced a second-order NOLH for k from 3 to 12 (MacCalman, 2013). These cataloged designs are available for download at harvest.nps.edu.

Liefvendahl and Stocki (2006) studied algorithms for constructing optimal Latin hypercubes by comparing the efficiency of the column-pairwise (CP) algorithm and the GA for optimized Latin hypercubes. The CP algorithm iteratively interchanges two elements in a column, while the GAs they evaluated use the traditional crossover and mutation operators; an operator is a set of instructions that performs actions on solutions to evolve them into better performers (Goldberg, 1989). Both of the evaluated algorithms operate directly on all columns of a Latin hypercube design. Goldberg (1989) states that every GA developed for a particular purpose should have a unique scheme, operators, and fitness function that will improve the search for better-performing solutions. Our proposed GA uses elements of both the CP and the traditional GAs mentioned in Liefvendahl and Stocki (2006) while having a different scheme, operator, and fitness function. Our GA scheme operates incrementally on one column at a time instead of the entire design. By constructing the design columns sequentially, we reduce the search space from $(n!)^k$ to $(n!)$ in order to improve the search for better-performing solutions.

For a more detailed description of the algorithm and an in-depth review of its performance (eg, relationships between n, k, and $\rho_{map(Z)}$), variability, and timing, see MacCalman (2013). All told, many thousands of hours were spent finding these designs within the DoD High-Performance Computing Modernization Program at the Navy DoD Supercomputing Resource Center (DSRC), Stennis Space Center and the US Air Force Research Laboratory DSRC, Wright-Patterson Air Force Base. We implemented the algorithm using Java$^{\text{TM}}$ 2 and made it available for download at harvest.nps.edu under the Software downloads link. Thus, users can develop their own custom designs based on their own needs. For example, one may not need a $\rho_{map(Z)} < 0.05$, and thus require a smaller n.

4. Design comparisons

This section compares our new second-order NOLH with the FCCD, BBH, D-Optimal, I-Optimal, and the following four space-filling designs: maximin distance Latin hypercube (referred in this paper as the LHS), Sphere-Packing (Sphere Pack), Maximum Entropy (Max Entropy), and Uniform. We used JMP$^{\text{TM}}$ 9.0 software to create each of the alternative designs for our comparison. Because the software uses a stochastic algorithm to create all the designs except for the FCCD and BBH, we instantiated the algorithm 100 times for the Sphere Pack, Uniform, and LHS designs, and 30 times for the D-Optimal, I-Optimal, and Max Entropy designs, and selected the one with the lowest $\rho_{map(Z)}$ to use in the comparisons below.

Figures 2, 3, and 4 compare the designs with the second-order NOLH using color correlation plots, scatter plot matrices, and a chart showing ML_2 *versus* $\rho_{map(Z)}$. In order to make a valid comparison, each design has 4 factors and 25 design points, and each factor's range is scaled from – 1 to 1.

Figure 2 indicates that the second-order NOLH has the lowest $\rho_{map(Z)}$ over all the terms. The other designs may fit accurate polynomial metamodels, but that may be due to chance if the terms in the true model coincide with regression matrix columns

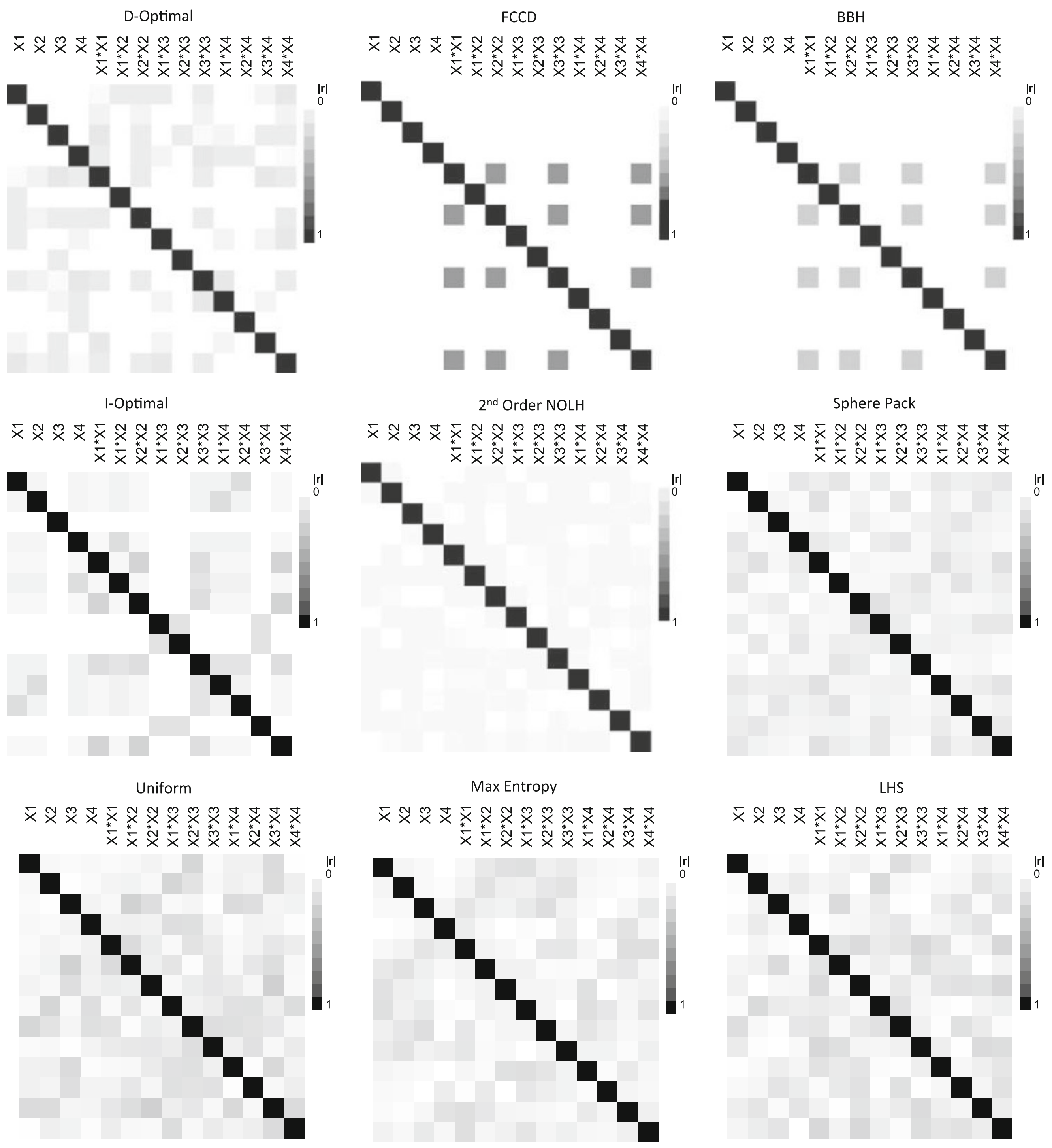


Figure 2 Color Correlation Plots. Darker-shaded colors indicate higher correlations (black represents a correlation of 1.0 and white represents a correlation of 0.0). Each plot shows designs with 4 factors and 25 design points for all second-order terms.

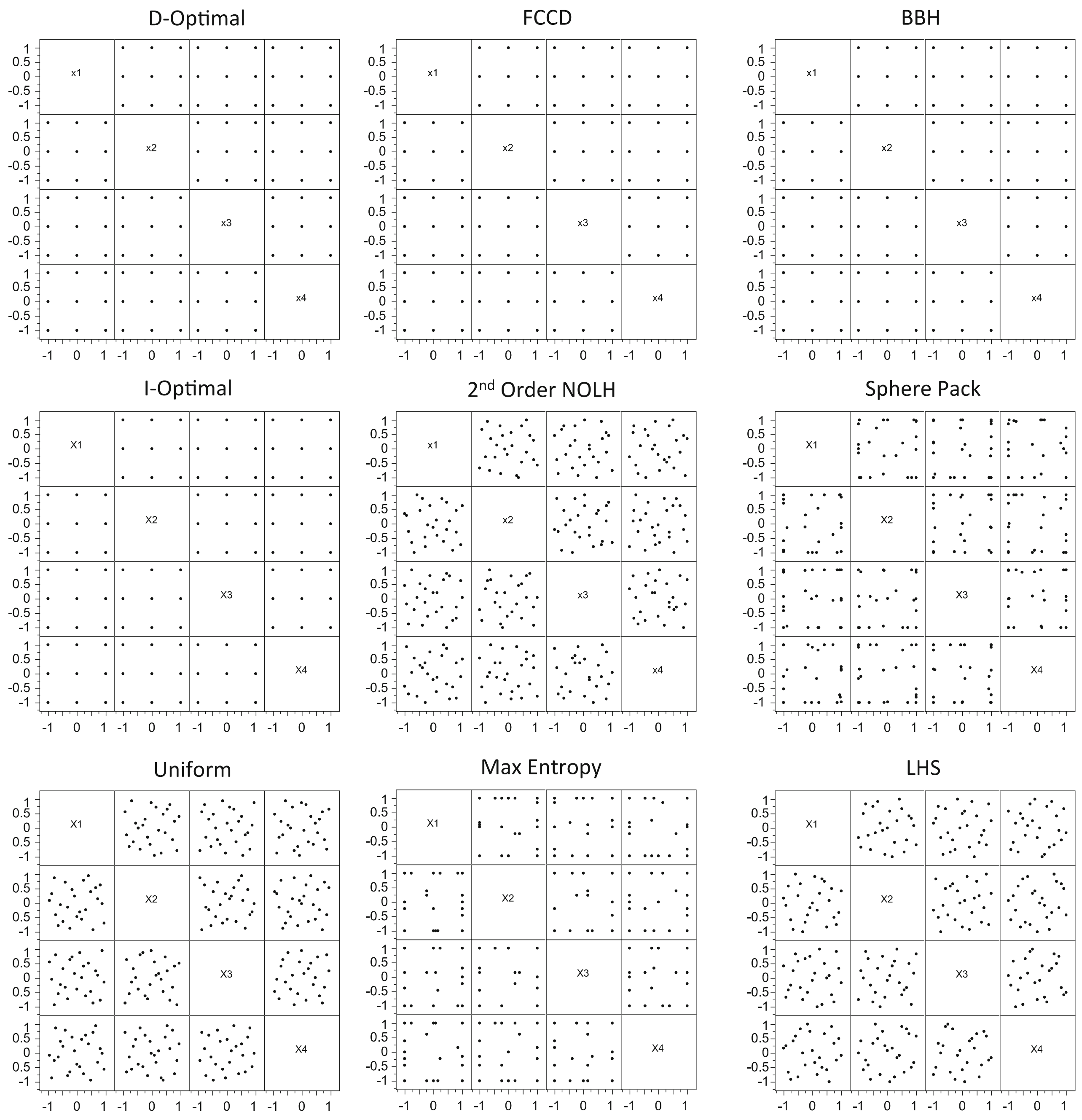


Figure 3 Design Scatter Plots. The chart shows the designs' two-dimensional projections of the 4-factor, 25-point design space. The FCCD, BBH, D-Optimal, and I-Optimal designs have points overlaid on top of each other because they only sample at the corners, faces, and center.

that have low correlation. The second-order NOLH has a $\rho_{map(Z)}$ of 0.032 and, therefore, guarantees that no term in the second-order model is confounded more than minimally with another term.

Figure 3 provides a visual perspective of each design's space-filling characteristic. By their construction, the FCCD and BBH designs only sample at the corners, faces, and center of the design space. While not constrained to this, the I-Optimal and D-Optimal designs JMP generates also have this feature. As a result, these four designs cannot fit metamodels containing polynomials of order greater than two because their third, fourth, or higher-order regression matrices have columns that are linearly dependent. Space-filling designs provide information about all portions of the design space by sampling throughout the region, which makes them well-suited to fit a variety of models (Santner *et al*, 2003).

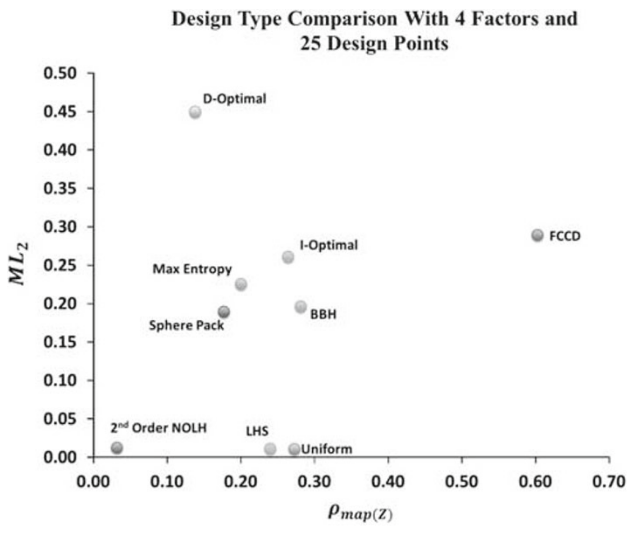


Figure 4 The ML_2 *versus* $\rho_{map(Z)}$ for each of the nine designs.

Figure 4 shows a plot of the ML_2 metric *versus* the $\rho_{map(Z)}$ for each of the nine designs. The second-order NOLH's $\rho_{map(Z)}$ dominates all other designs for the second-order regression matrix. In terms of space-filling, the second-order NOLH has an ML_2 very close to the LHS and Uniform design.

The recently developed Latin hypercube designs from Yang *et al* (2013) that guarantee orthogonality between 'all main effects and any main effect and any quadratic or two-factor interaction effect'. Thus, a comparison with them is warranted. These designs use powers of two, and the closest available variant is the $k=4$ and $n=32$ OLH. With respect to correlation for a full second-order model, $\rho_{map(Z)}=0.877$, as there are substantial correlations among quadratic term effects. The space-filling is quite good, with an ML_2 of 0.009. While this is better than all the designs in Figure 4, it has the advantage of using 32 points rather than 25.

In addition to the ML_2, we also compared all designs using the distance metric developed by Morris and Mitchell (1995), and also used in Joseph and Hung (2008) and Hernandez *et al* (2012). Since the results were similar to what the ML_2 measure revealed, an additional figure provides no insight.

5. Cataloged designs

Figure 5 compares a sample of the computer-generated optimal and space-filling designs created in JMP^{TM} 9.0 with the second-order NOLH for upto 12 factors with the same number of design points. We calculated each design's $\rho_{map(Z)}$ using a matrix that includes all second-order terms. JMP^{TM} 9.0 uses different random seeds for each of the optimal and space-filling design creation and, therefore, has a different $\rho_{map(Z)}$ or ML_2 results for each generation. Thus, the designs shown are only a single instantiation that may not have the best $\rho_{map(Z)}$ or ML_2. However, the variability in these measures is low, so the general qualitative pattern holds across different realizations. For example, the average $\rho_{map(Z)}$ of 100 LHS designs with 4 factors and 25 design points generated using JMP^{TM} 9.0 is 0.35 with a standard deviation of 0.053; the average ML_2 result for the same design is 0.011 with a standard deviation of 0.0006.

We can see from Figure 5 that the second-order NOLH designs have the lowest $\rho_{map(Z)}$, with excellent space-filling properties based on its ML_2 performance. This makes the second-order NOLH design very competitive against the other five design types; the second-order NOLH designs are available online at harvest.nps.edu under the Software downloads link to the left. Table 1 summarizes the performance metrics for the 10 designs.

As we noted in the introduction, there are many applications of stochastic simulations within DoD. There are well over 100 theses from the Naval Postgraduate School that have used simulations to examine complex military problems (see, harvest.nps.edu). These problems include systems design decisions, operational and tactical insights from force-on-force scenarios, resource allocation, system life-cycle management, and verification and validation of simulation models. Nearly all of these theses utilize a DOE using a NOLH design for exploratory analysis. After fitting metamodels with a polynomial form, examining the output landscape using partition trees and other data mining methods, the results of the exploratory analysis reveal that input factors impact one or more responses. The key insights are often interesting interactions between factors, thresholds that occur in local areas of the design region, and factors that yield diminishing or increasing rates of return. In the future, these critical insights will no longer be confounded due to high correlations. Our proposed second-order NOLH designs addresses this confounding problem by guaranteeing minimal correlations among all terms in a full second-order model while simultaneously allowing us to fit higher order models if desired and explore the interior of the design space. See MacCalman (2013) for an applied problem that uses the second-order NOLH design in a system design application.

6. Conclusion

In order to understand the complex nature of our world, we must be able to detect the driving factors during experiments and understand how they impact the results. Computer simulations and DOE enable us to model our world by simultaneosly exploring numerous factors that may affect the complex nature of the simulation response. Because we are concerned with understanding complex stochastic simulation responses, we focus on fitting polynomials with constant coefficients that can reveal the significant factors and describe their synergistic and non-linear effects on the response. The simulation analyst needs flexible space-filling experimental designs that can fit a wide variety of high-order polynomial models with a single experiment. In addition, because we never know the true form of the response surface, analysts need designs that minimize *a priori*

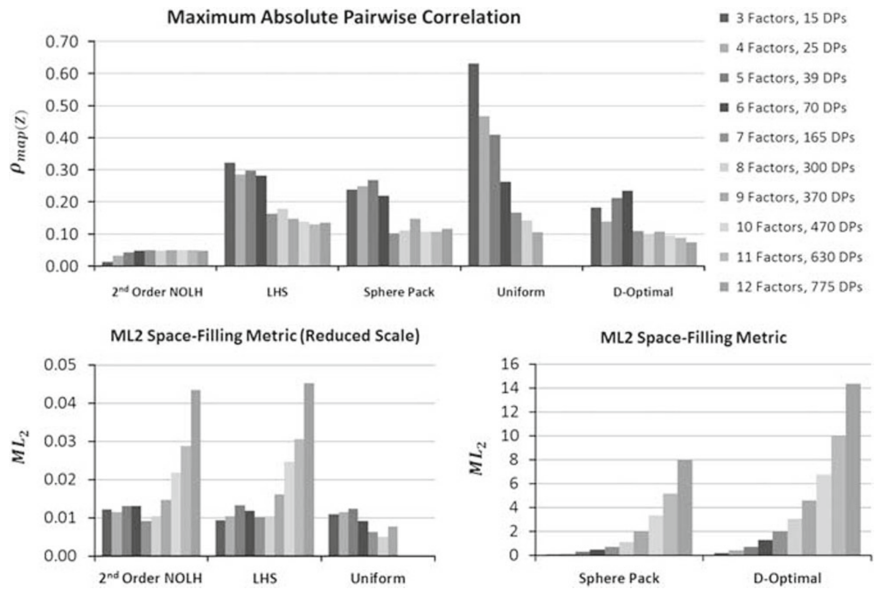


Figure 5 Design comparisons of the second order $\rho_{map(Z)}$ and ML_2 for a sample of design types with the same number of design points. The 10-, 11-, and 12-factor uniform designs are not listed due to the time required to construct them. The ML_2 charts are split into two because of the differences in scale among the designs.

Table 1 Cataloged second-order NOLH Designs

Number of factors	*Number of Design Points*	$\rho_{map(z)}$	ML_2
3	15	0.012	0.012
4	25	0.032	0.011
5	39	0.043	0.013
6	70	0.048	0.013
7	165	0.049	0.009
8	300	0.048	0.010
9	370	0.050	0.015
10	470	0.050	0.022
11	630	0.050	0.029
12	775	0.048	0.044

model assumptions in order to analyze multiple responses that potentially have many different forms.

In this paper, we presented a new GA that constructs second-order NOLH designs with minimal correlations between all main, quadratic, and two-way interaction terms. The designs proposed in this paper can fit a wide variety of polynomial models that will facilitate a better understanding of the simulation response. In addition, because our designs are from the LH family, they are excellent candidates for the GP model for prediction purposes due to their space-filling characteristics—however, that is beyond the scope of this paper. The key advantage of our designs is threefold. First, they can fit the most commonly used polynomial metamodel with guaranteed minimal correlations; second, with suitable caution, they can fit higher-order models to a handful of factors; and third, they are space-filling allowing us to take full advantage of partition trees to find interesting behavior in local areas of the experimental region.

The designs presented in this paper only considered continuous factors. Our ongoing research involves constructing designs that minimize the correlation between all second-order terms for a mix of continuous, discrete, and categorical factors. The infinite combinations of discrete and categorical levels require a need for a custom design creator capable of generating second-order NOLH designs for a mix of factor types often encountered during simulation studies. Moreover, we note the opportunity to research the performance of our designs for prediction purposes using the GP model.

Acknowledgements—The authors would like to thank our colleagues at the Naval Postgraduate School for their insightful comments. In addition, the authors are grateful to the referees, whose feedback improved the content and clarity of this paper. The research was made possible by a grant from the Office of Naval Research (N0001412WX20823).

References

Ankenman B, Nelson B and Staum J (2010). Stochastic kriging for simulation metamodeling. *Operations Research* **58**(2): 371–382.

Atkinson AC, Donev AN and Tobias RD (2007). *Optimum Experimental Designs, with SAS*. Oxford: Oxford University Press: .

Bankes S (1993). Exploratory modeling for policy analysis. *Operations Research* **41**(3): 435–449.

Barton RR (1998). Simulation metamodels. In: Medeiros DJ, Watson EF, Carson JS, and Manivannan, MS (eds). *Proceedings of the 1998 Winter Simulation Conference*, Vol. 1, IEEE. Piscataway, NJ, pp 167–174.

Beers V and Kleijnen JPC (2003). Kriging for interpolation in random simulation. *Journal of the Operational Research Society* **54**(3): 255–262.

Beers V and Kleijnen JPC (2004). Kriging interpolation in simulation: A survey. In *Proceedings of the 2004 Winter Simulation Conference*. Washington, D.C., Vol. 1, pp 113–112.

Box GEP and Behnken DW (1960). Some new three level designs for the study of quantitative variables. *Technometrics* **2**(4): 455–475.

Box GEP and Draper NR (1987). *Empirical Model-Building and Response Surfaces*. Wiley: New York, NY.

Challenor P (2013). Experimental design for the validation of kriging metamodels in computer experiments. *Journal of Simulation* **7**(4): 290–296.

Cioppa TM and Lucas TW (2007). Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics* **49**(1): 45–55.

Fang KT (1980). The uniform design: Application of number-theoretic methods in experimental design. *Acta Mathematicale Applicatae Sinica* **3**: 363–372.

Fang KT, Lin DKJ, Winker P and Zhang Y (2000). Uniform design: Theory and application. *Technometrics* **42**(3): 237–248.

Florian A (1992). An efficient sampling scheme: Updated Latin hypercube sampling. *Probabilistic Engineering Mechanics* **7**(2): 123–130.

Goldberg DE (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1st edn, Addison-Wesley Longman Publishing Co: Boston, MA.

Goldfarb HB, Borror CM, Montgomery DC and Anderson-Cook C (2005). Using genetic algorithms to generate mixture-process experimental designs involving control and noise variables. *Journal of Quality Technology* **37**(1): 60–74.

Heredia-Langner A, Carlyle WM, Montgomery DC and Borror CM (2004). Model-robust optimal designs: A genetic algorithm approach. *Journal of Quality Technology* **35**(1): 263–279.

Heredia-Langner A, Carlyle WM, Montgomery DC, Borror CM and Runger GC (2003). Genetic algorithms for the construction of D-optimal designs. *Journal of Quality Technology* **35**(1): 28–46.

Hernandez AS, Lucas TW and Carlyle WM (2012). Constructing nearly orthogonal Latin hypercubes for any nonsaturated run-variable combination. *ACM Transactions on Modeling and Computer Simulation* **22**(20): 4.

Hickernell FJ (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation* **67**(221): 299–322.

Holland JH (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press: Ann Arbor.

Jin R, Chen W and Sudjianto A (2005). An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference* **134**(1): 268–287.

Johnson ME, Moore LM and Ylvisaker D (1990). Minimax and maxmin distance design. *Journal of Statistical Planning and Inference* **26**(2): 131–148.

Jones B and Nachtsheim CJ (2011). A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology* **43**(1): 1–15.

Joseph VR and Hung Y (2008). Orthogonal-maximin Latin hypercube designs. *Statistica Sinica* **18**(1): 171–186.

Kim L and Loh H (2003). Classification trees and bivariate linear discriminant node models. *Journal of Graphical and Statistics* **12**(3): 512–530.

Kleijnen JPC (2005). An overview of the design and analysis of simulation experiments for sensitivity analysis. *European Journal of Operations Research* **164**(2): 287–300.

Kleijnen JPC (2008a). Simulation experiments in practice: Statistical design and regression analysis. *Journal of Simulation* **2**(1): 19–27.

Kleijnen JPC (2008b). Design of experiments: Overview. In *Proceedings of the 2008 Winter Simulation Conference*. Miami, FL.

Kleijnen JPC, Sanchez SM, Lucas TW and Cioppa TM (2005). A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing* **17**(3): 263–289.

Koehler JR and Owen AB (1996). Computer experiments. *Handbook of Statistics* **13**: 261–308.

Law A (2007). *Simulation Modeling and Analysis*. 4th edn, McGraw-Hill: New York.

Liefvendahl M and Stocki R (2006). A study on algorithms for optimization of Latin hypercubes. *Journal of Statistical Planning and Inference* **136**(9): 3231–3247.

Lucas TW, Kelton WD, Sanchez SM and Sanchez PJ (2015). Changing the paradigm: Simulation, often the method of first resort. *Naval Research Logistics* **62**: 293–303.

MacCalman AD (2013). Flexible space-filling designs for complex system simulations. Doctoral dissertation. Naval Postgraduate School: Monterey, CA.

McKay MD, Beckman RJ and Conover WJ (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**(2): 239–245.

Michalewicz Z and Fogel DB (2010). *How to Solve it: Modern Heuristics*. Springer-Verlag Berlin: Heidelberg, NY.

Moon H, Santner TJ and Dean A (2011). Algorithms for generating maximin Latin hypercube and orthogonal designs. *Journal of Statistical Theory and Practice* **5**(1): 81–98.

Morris MD and Mitchell TJ (1995). Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference* **43**(3): 381–402.

Myers RH, Montgomery DC and Anderson-Cook CM (2009). *Response Surface Methodology: Process and Product Optimization using Designed Experiments*. 3rd edn, Wiley: Hoboken, NJ.

National Research Council (2006). *Defense Modeling, Simulation, and Analysis: Meeting the Challenge Committee on Modeling and Simulation for Defense Transformation*. The National Academies Press: Washington, DC.

Owen AB (1994). Controlling correlations. In latin hypercube samples. *Journal of the American Statistical Association: Theory and Methods* **89**(428): 1517–1522.

Pang FM, Liu Q and Lin DKJ (2009). A construction method for orthogonal Latin hypercube designs with prime power levels. *Statist. Sinica* **19**(4): 1721–1728.

Ryan TP (2007). *Modern Experimental Design*. 1st edn, Wiley-Interscience: Hoboken, NJ.

Sacks JW, Welch J, Mitchell TJ and Wynn HP (1989). Design and analysis of computer experiments (includes comments and rejoinder). *Statistical Science* **4**(4): 409–435.

Sall J (2007). *JMP Start Statistics: A Guide to Statistics and Data Analysis Using JMP, Fourth Edition*. 4th edn, SAS Publishing: Cary, NC.

Sanchez SM (2008). Better than a petaflop: The power of efficient experimental design. In *Proceedings of the 40th Conference on Winter Simulation*, pp 73–84.

Sanchez SM, Lucas TW, Sanchez PJ, Nannini CJ and Wan H (2012). Designs for large-scale simulation experiments, with application to defense and homeland security. In: Hinkelmann K (ed). *The Design and Analysis of Computer Experiments. Volume 3: Special Designs and Applications*. Wiley: Hoboken, NJ, pp 413–441.

Santner TJ, Williams BJ and Notz WI (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag: New York, NY.

Shewry MC and Wynn HP (1987). Maximum entropy sampling. *Journal of Applied Statistics* **14**(7): 165–170.

Siggelkow N (2002). Misperceiving interactions among complements and substitutes: Organizational consequences. *Management Science* **48**(7): 900–916.

Steinberg DM and Lin DKJ (2006). A construction method for orthogonal Latin hypercube designs. *Biometrika* **93**(2): 279–288.

Sun FS, Liu MQ and Lin DKJ (2009). Construction of orthogonal Latin hypercube designs. *Biometrika* **96**(4): 971–974.

Vieira Jr H (2008). Optimizing stochastic functions using a genetic algorithm: An aeronautic military application. In: Siarry P and Michalewicz Z (eds). *Advances in Metaheuristics for Hard Optimization*. Springer-Verlag: Berlin, pp 353–363.

Vieira Jr H, Sanchez SM, Kienitz KH and Belderrain MCN (2011). Generating and improving orthogonal designs by using mixed integer programming. *European Journal of Operational Research* **215**(3): 629–638.

Vieira Jr H, Sanchez SM, Kienitz KH and Belderrain MCN (2013). Efficient, nearly orthogonal-and-balanced, mixed designs: An effective way to conduct trade-off analysis via simulation. *Journal of Simulation* **7**(4): 264–275.

Yang JF, Lin CD, Qian PZG and Lin DKJ (2013). Construction of sliced orthogonal Latin hypercube Designs. *Statitsica Sinica* **23**(3): 1117–1130.

Ye KQ (1998). Orthogonal column Latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association: Theory and Methods* **93**(444): 1430–1439.

Received 22 April 2014;
accepted 31 March 2016 after two revisions