Chapter 5
Space-Filling Designs for Computer Experiments

5.1 Introduction

This chapter and the next discusses how to select inputs at which to compute the output of a computer experiment to achieve specific goals. The inputs one selects constitute the "experimental design." As in previous chapters, we refer to the inputs as "runs." The region corresponding to the values of the inputs over which we wish to study or model the response is the experimental region. A point in this region corresponds to a specific set of values of the inputs. Thus, an experimental design is a specification of points (runs) in the experimental region at which we wish to compute the response.

This chapter begins by reviewing some of the basic principles of classical experimental design and then present an overview of some of the strategies that have been employed in computer experiments. For details concerning classical design see, for example, the books by Atkinson and Donev (1992), Box and Draper (1987), Dean and Voss (1999), Pukelsheim (1993), Silvey (1980), and Wu and Hamada (2000).

5.1.1 Some Basic Principles of Experimental Design

Suppose that one observes a response and wishes to study how that response varies as one changes a set of inputs. In physical experiments, there are a number of issues that make this problematic. First, the response may be affected by factors other than the inputs we have chosen to study. Unless one can completely control the effects of these additional factors, repeated observations at the same values of the inputs will vary as these additional factors vary. The effects of additional factors can either be unsystematic (random) or systematic. Unsystematic effects are usually referred to as random error, measurement error, or noise. Systematic effects are often referred to as bias. There are strategies for dealing with both noise and bias.

Replication and *blocking* are two techniques used to estimate and control the magnitude of random error. Replication (observing the response multiple times at the same set of inputs) allows one to directly estimate the magnitude and distribution of random error. Also, the sample means of replicated responses have smaller variances than the individual responses. Thus, the relation between these means and the inputs gives a clearer picture of the effects of the inputs because uncertainty from random error is reduced. In general, the more observations one has, the more information one has about the relation between the response and the inputs.

Blocking involves sorting experimental material into, or running the experiment in, relatively homogeneous groups called blocks. The corresponding analysis explores the relation between the response and the inputs within blocks, and then combines the results across blocks. Because of the homogeneity within a block, random error is less within a block than between blocks and the effects of the inputs more easily seen. There is an enormous body of literature on block designs, including both statistical and combinatorial issues. General discussions include John (1980), John (1987), Raghavarao (1971), or Street and Street (1987).

Bias is typically controlled by *randomization* and by exploring how the response changes as the inputs change. Randomization is accomplished by using a well-defined chance mechanism to assign the input values as well as any other factors that may affect the response and that are under the control of the experimenter, such as the order of experimentation, to experimental material. Factors assigned at random to experimental material will not systematically affect the response. By basing inferences on changes in the response as the input changes, bias effects "cancel," at least on average. For example, if a factor has the same effect on every response, subtraction (looking at changes or differences) removes the effect.

Replication, blocking, and randomization are basic principles of experimental design for controlling noise and bias. However, noise and bias are not the only problems that face experimenters. Another problem occurs when one is interested in studying the effects of several inputs simultaneously and the inputs themselves are highly correlated. This sometimes occurs in observational studies. If, for example, the observed values of two inputs are positively correlated so that they increase together simultaneously, then it is difficult to distinguish their effects on the response. Was it the increase in just one or some combination of both that produced the observed change in the response? This problem is sometimes referred to as collinearity. Orthogonal designs are used to overcome this problem. In an orthogonal design, the values of the inputs at which the response is observed are uncorrelated. An orthogonal design allows one to independently assess the effects of the different inputs. There is a large body of literature on finding orthogonal designs, generally in the context of factorial experiments. See, for example, Hedayat et al (1999).

Another problem that can be partly addressed (or at least detected) by careful choice of an experimental design, occurs when the assumptions one makes about the nature of the relation between the response and the inputs (the statistical model) are incorrect. For example, suppose one assumes that the relationship between the response and a single input is essentially linear when, in fact, it is highly nonlinear. Inferences based on the assumption that the relationship is linear will be incorrect.

It is important to be able to detect strong nonlinearities and one will need to observe the response with at least three different values of the input in order to do so. Error that arises because the assumed model is incorrect is sometimes referred to as model bias. Diagnostics, such as scatterplots and quantile plots, are used to detect model bias. The ability to detect model bias is improved by careful choice of an experimental design, for example, by observing the response at a wide variety of values of the inputs. One would like to select designs that will enable one to detect model inadequacies and lead to inferences that are relatively insensitive to model bias. This usually requires specifying both the model one intends to fit to the data as well as the form of an alternative model whose bias one wishes to guard against; thus designs for model bias are selected to protect against certain types of bias. Box and Draper (1987) discuss this issue in more detail.

In addition to general principles, such as replication, blocking, randomization, orthogonality, and the ability to detect model bias, there exist very formal approaches to selecting an experimental design. The underlying principle is to consider the purpose of the experiment and the statistical model for the data and choose the design accordingly. If one can formulate the purpose of our experiment in terms of optimizing a particular quantity, one can then ask what inputs one should observe the response at to optimize this quantity. For example, if one is fitting a straight line to data, one might wish to select our design so as to give the most precise (minimum variance) estimate of the slope. This approach to selection of an experimental design is often referred to as *optimal design*. See Atkinson and Donev (1992), Pukelsheim (1993), or Silvey (1980) for more on the theory of optimal design. In the context of the linear model, popular criteria involve minimizing some function of the covariance matrix of the least squares estimates of the parameters. Some common functions are the determinant of the covariance matrix (the generalized variance), the trace of the covariance matrix (the average variance), and the average of the variance of the predicted response over the experimental region. A design minimizing the first criterion is called *D-optimal*, a design minimizing the second is called *A-optimal*, and a design minimizing the third is called *I-optimal*. In many experiments, especially experiments with multiple objectives, it may not be clear how to formulate the experiment goal in terms of some quantity that can be optimized. Furthermore, even if one can formulate the problem in this way, finding the optimal design may be quite difficult.

In many experiments all the inputs at which one will observe the response are specified in advance. These are sometimes referred to as a single-stage or one-stage experimental designs. However, there are good reasons for running experiments in multiple stages. Box et al (1978) (page 303), advocate the use of sequential or multistage designs.

"In exploring a functional relationship it might appear reasonable at first sight to adopt a comprehensive approach in which the entire range of every factor was investigated. The resulting design might contain all combinations of several levels of all factors. However, when runs can be made in successive groups, this is an inefficient way to organize experimental programs. The situation relates to the paradox that the

best time to design an experiment is after it is finished, the converse of which is that the worst time is at the beginning, when the least is known. If the entire experiment was designed at the outset, the following would have to be assumed known: (1) which variables were the most important, (2) over what ranges the variables should be studied, (3) in what metrics the variables and responses should be considered) e.g., linear, logarithmic, or reciprocal scales), and (4) what multivariable transformations should be made (perhaps the effects of variables x_1 and x_2 would be most simply expressed in terms of their ratio x_1/x_2 and their sum $x_1 + x_2$.

The experimenter is least able to answer such questions at the outset of an investigation but gradually becomes more able to do so as a program evolves.

All the above arguments point to the desirability of a sequence of moderately sized designs and reassessment of the results as each group of experiments becomes available."

5.1.2 Design Strategies for Computer Experiments

Computer experiments, at least as considered here, differ from traditional physical experiments in that repeated observations at the same set of inputs yield (aside from numerical error) identical responses. A single observation at a given set of inputs give perfect information about the response at that set of inputs, so replication is unnecessary. The greatest uncertainty arises in computer experiments because one does not know the exact functional form of the relationship between the inputs and the response, although the response can be computed at any given input. Any functional models that are used to describe the relationship are only approximations. The discrepancy between the actual response produced by the computer code and the response predicted from the fitted model is the error. In the previous subsection such error was referred to as model bias .

Based on these observations, two principles for selecting designs in the types of computer experiments considered are the following.

1. *Designs should not take more than one observation at any set of inputs. (But note that this principle assumes the computer code remains unchanged over time. When a design is run sequentially and the computer code is written and executed by a third party, it may be good policy to duplicate one of the design points in order to verify that the code has not been changed over the course of the experiment.)*
2. *Because one doesn't know the true relation between the response and inputs, designs should allow one to fit a variety of models and should provide information about all portions of the experimental region.*

If a priori one believes that interesting features of the true model are just as likely to be in one part of the experimental region as another, if one's goal is to be able to do prediction over the entire range of the inputs, and if one is running a single-stage experiment it is plausible to use designs that spread the points (inputs, runs) at which

one observes the response evenly throughout the region. There are a number of ways to define what it means to spread points evenly throughout a region and these lead to various types of designs. This chapter discusses a number of these. Among the designs considered are designs based on selecting points in the experimental region by certain sampling methods; designs based on measures of distance between points that allow one to quantify how evenly spread points are; designs based on measures of how close points are to being uniformly distributed throughout a region; and designs that are a hybrid of or variation on these designs. All the designs in this chapter will be referred to as *space-filling* or *exploratory* designs.

The term "space-filling" is used widely in the literature on computer experiments. It seems that in most cases, space-filling is meant in an intuitive sense and as a synonym for "evenly spread." However, it also has a more technical meaning. It can refer to a method for generating designs for any run size n_s such that as n_s increases, the method produces designs that are increasingly dense in the design space (in other words, fill the design space). See Vazquez and Bect (2011) for an analysis of the limiting properties of the prediction variance for such designs. In particular, assuming the GP model is correct, for sufficiently large sample sizes, Vazquez and Bect (2011) show that no design will outperform certain space-filling designs (those with an asymptotic fill distance of $O(n_s^{-\frac{1}{d}})$) in terms of the rate at which the maximum of the mean-squared prediction error decreases as n_s increases. This provides some theoretical justification for using space-filling designs.

When runs of a computer experiment are expensive or time-consuming, and hence observing the response at a "large" number of inputs is not possible, what is a reasonable sample size that will allow one to fit the models described in Chapters 2–4? One rule of thumb suggested by Chapman et al (1994) and Jones et al (1998) is to use a sample size of $10d$ when the input space is of dimension d. However, because the "volume" of the design space increases as a power of d, $10d$ points becomes a very sparse sample as d increases. Obviously 10 points evenly spread over the unit interval are much more densely distributed than 100 points in the ten-dimensional unit cube. So is the $10d$ rule of thumb reasonable? Loeppky et al (2009) carefully investigate this issue and conclude that a sample size of $10d$ is a reasonable rule of thumb for an initial experiment when $d \leq 5$. When the response is sensitive to relatively few of the inputs, the rule is also reasonable for an initial experiment for d up to 20 or even larger. Loeppky et al (2009) also discuss diagnostics one can use to determine whether additional observations are needed (beyond those recommended by the $10d$ rule of thumb) and approximately how many might be needed to improve overall fit. They point out that one should always check the accuracy of the predictor fitted to the data and if it is poor, additional observations (perhaps many) may be needed.

The complexity of the input-output relationship has a direct bearing on the required sample size. Polynomial models provide some insight to the question of sample size. The minimum number of points needed to uniquely determine a response surface of order r in d variables (all monomials of order r or less are included) is

$$\binom{r+d}{r}. \tag{5.1.1}$$

For a second-order response surface ($r = 2$), the $10d$ rule of thumb exceeds equation 5.1.1 up to $d = 16$. For a third-order response surface, the $10d$ rule of thumb exceeds equation 5.1.1 up to $d = 4$. For a fourth-order response surface, the $10d$ rule of thumb is greater than equation 5.1.1 up to $d = 2$. Also, for an input-output relation such as $y = \sin(c\pi x), 0 \leq x \leq 1$ the $10d$ rule won't allow for enough observations in one-dimension to produce an adequate predictor for large c, assuming one has no prior knowledge of the functional form of this relationship.

Are there real examples where one encounters input-output relationships that produce very complicated response surfaces? Chen et al (2011) discuss a computer experiment concerning bistable laser diodes in which the $d = 2$ response surface is quite rough over a portion of the design space and would require substantially more than 20 observations to accurately approximate.

Although not in the context of computer experiments, it is interesting to note that Box et al (1978) (page 304) recommend the following for multi-stage designs: "As a rough general rule, not more than one quarter of the experimental effort (budget) should be invested in a first design."

In practice,the true model that describes the relation between the inputsand the response is unknown. However, if the models to be fit to the data come from a sufficiently broad class, one may be willing to assume some model in this class is (to good approximation) "correct." In this case it is possible to formulate specific criteria for choosing a design and adopt an optimal design approach. Because the models considered in the previous chapters are remarkably flexible, this approach seems reasonable for these models. Thus, Chapter **??** discusses some criterion-based methods for selecting designs.

5.2 Designs Based on Methods for Selecting Random Samples

In the language of Section 1.3 the designs described in this section are used in cases when all inputs $\boldsymbol{x}$ are control variables as well as in cases when they are mixtures of control and environmental variables. However, most of these designs were originally motivated by their usefulness in applications where the inputs were all environmental variables; in this case the inputs are denoted by $\boldsymbol{X}$ to emphasize their random nature. Let $y(\cdot)$ denote the output of the code. When the inputs are environmental variables, the most comprehensive objective would be to find the distribution of the random variable $Y = y(\boldsymbol{X})$ when $\boldsymbol{X}$ has a known distribution. If, as is often the case, this is deemed too difficult, the easier problem of determining some aspect of its distribution such as its mean $E\{Y\} = \mu$ or its variance is considered. Several of the designs introduced in this section, in particular the Latin hypercube design, were developed to solve the problem of estimating μ in such a setting. However, the reader should bear in mind that such designs are useful in more general input settings.

5.2.1 Designs Generated by Elementary Methods for Selecting Samples

Intuitively, one would like designs for computer experiments to be space-filling when prediction accuracy over the entire experimental region is of primary interest. The reason for this is that interpolators are used as predictors (e.g., the BLUP or its Bayesian counterparts such as those that arise as the means of the predictive distributions derived in Section 3.3. Hence, the prediction error at any input site is a function of its location relative to the design points. Indeed, Section 4.2 shows that the prediction error is *zero* at each of the design points. For this reason, designs that are not space-filling, for example, designs that concentrate points on the boundary of the design space, can yield predictors that perform quite poorly in portions of the experimental region that are sparsely observed.

Deterministic strategies for selecting the values of the inputs at which to observe the response are to choose these values so they are spread evenly throughout or fill the experimental region. There are several methods that might be used to accomplish this, depending on what one means by "spreading points evenly" or "filling the experimental region."

A very simple strategy is to select points according to a regular grid pattern superimposed on the experimental region. For example, suppose the experimental region is the unit square $[0,1]^2 = [0,1] \times [0,1]$. If one wishes to observe the response at 25 evenly spaced points, one might consider the grid of points $\{0.1, 0.3, 0.5, 0.7, 0.9\} \times \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

There are several statistical strategies that one might adopt. One possibility is to select a simple random sample of points from the experimental region. In theory, there are infinitely many points between 0 and 1 and this makes selecting a simple random sample problematic. In practice, one only records numbers to a finite number of decimal places and thus, in practice, the number of points between 0 and 1 can be regarded as finite. Therefore, one can assume the experimental region consists of finitely many points and select a simple random sample of these.

Simple random sampling in computer experiments can be quite useful. If the inputs are sampled according to some distribution (for example, a distribution describing how the inputs are distributed in a given application), one can get a sense of how the corresponding outputs are distributed and this can serve as the basis for inferences about the distribution of the output. However, for many purposes, other sampling schemes, such as stratified random sampling, are preferable to simple random sampling. Even if the goal is simply to guarantee that the inputs are evenly distributed over the experimental region, simple random sampling is not completely satisfactory, especially when the sample sizes are relatively small. With small samples in high-dimensional experimental regions, the sample will typically exhibit some clustering and fail to provide points in large portions of the region.

To improve the chances that inputs are spread "evenly" over the experimental region, one might use stratified random sampling. If a design consisting of n_s runs of the simulator is desired, one would divide the experimental region into n_s strata,

spread evenly throughout the experimental region, and randomly select a single point from each. Varying the size and position of the strata, as well as sampling according to different distributions within the strata, allows considerable flexibility in selecting a design. This may be more or less useful, depending on the purpose of the computer experiment. For example, one may wish to explore some portions of the experimental region more thoroughly than others. However, if the goal is simply to select points that are spread evenly throughout the experimental region, spacing the strata evenly and sampling each according to a uniform distribution would seem the most natural choice.

If the output is thought to depend on only a few of the inputs (this is sometimes referred to as factor sparsity), then one might want to be sure that points are evenly spread across the projection of the experimental region onto these factors. A design that spreads points evenly throughout the full experimental region will not necessarily have this property. Alternatively, if one believes the model is well approximated by an additive model (a model that is the sum of terms that are each a function of only one of the inputs), a design that spreads points evenly across the range of each individual input (one-dimensional projection) might be desirable. For n_s runs of the simulator, it can be difficult to guarantee that a design has such projection properties, even with stratified sampling. Latin hypercube sampling, the topic of the next subsection, is a way to generate designs that spread observations evenly over the range of each input separately.

5.2.2 Designs Generated by Latin Hypercube Sampling

Designs generated by Latin hypercube sampling are called Latin hypercube designs (LHD) throughout this book. For simplicity, assume the experimental region is the unit square $[0, 1]^2$. To obtain an LHD consisting of n_s points, divide each axis $[0, 1]$ into the n_s equally spaced intervals $[0, 1/n_s), \ldots, [(n_s - 1)/n_s, 1]$. This partitions the unit square into n_s^2 cells of equal size. Now, fill these cells with the integers 1, 2, $\ldots$, n_s so as to form a Latin square, i.e., by an arrangement in which each integer appears exactly once in each row and in each column of this grid of cells. Select one of the integers at random. In each of the n_s cells containing this integer, select a point at random. The resulting n_s points are an LHD of size n_s (see Figure 5.2 for an example with $n_s = 5$).

The LHD method of choosing the sample ensures that points are spread evenly over the values of each input variable. Of course, such an LH sample could select points that are spread evenly along the diagonal of the square (see Figure 5.3). Although the points in such a sample have projections that are evenly spread out over the values of each input variable separately, we would not regard them as evenly spread out over the entire unit square. Furthermore, recalling the discussion of space-filling in Section 5.2.1, LHDs consisting of n_s points along the diagonal do not become dense in the unit cube as n_s increases.

A general procedure for obtaining an LH sample of size n_s from $\boldsymbol{X} = (X_1, \ldots, X_d)$ when $\boldsymbol{X}$ has independently distributed components will now be described. Stein (1987) discusses the implementation of LH sampling when $\boldsymbol{X}$ has dependent components, but this case is not considered here.

In the independence case the idea is as follows. Suppose that an LH sample of size n_s is to be selected. The domain of each input variable is divided into n_s intervals. The set of all possible Cartesian products of these intervals constitutes a partitioning of the d-dimensional sample space into n_s^d "cells." A set of n_s cells is chosen from the n_s^d population of cells in such a way that the projections of the centers of each of the cells onto each axis yield n_s distinct points on the axis; then a point is chosen at random in each selected cell.

In detail, the LH sample over $\prod_{k=1}^{d}[a_k, b_k]$, where all end-points are finite is constructed as follows. For $k = 1, \ldots, d$, let $F_k(\cdot)$ denote the desired (marginal) distribution of X_k, the k^{th} component of $\boldsymbol{X}$. If X_k has support on $[a_k, b_k]$ then $X_k = \frac{X_k - a_k}{b_k - a_k}$ is used to scale and shift the support to $[0, 1]$; the inverse transform is used to place the support back on the original scale.

Divide the k^{th} axis into n_s parts, each of which has equal probability, $1/n_s$, under $F_k(\cdot)$. The division points for the k^{th} axis are

$$F_k^{-1}(\frac{1}{n_s}), \ldots, F_k^{-1}(\frac{n_s - 1}{n_s}).$$

To choose n_s of the cells so created, let $\boldsymbol{\Pi} = (\Pi_{jk})$ be an $n_s \times d$ matrix having permutations of $\{1, 2, \ldots, n_s\}$ as columns which are randomly selected from the set of all possible permutations. Then the "lower-left hand" coordinates of the j^{th} cell in $\mathbb{R}^d$ are

$$F_k^{-1}(n_s^{-1}(\Pi_{jk} - 1)), \quad k = 1, \ldots, d, \quad j = 1, \ldots, n_s.$$

with the convention $F_k^{-1}(0) = 0$.

For $j = 1, \ldots, n_s$, let X_{jk}, $k = 1, \ldots, d$, denote the k^{th} component of the j^{th} vector, $\boldsymbol{X}_j$. Then define the LH sample to have values

$$X_{jk} = F_k^{-1}(\frac{1}{n_s}(\Pi_{jk} - 1 + U_{jk})),$$

where the $\{U_{jk}\}$ are independent and identically distributed $U[0, 1]$ deviates, for $j = 1, \ldots, n_s$ and $k = 1, \ldots, d$. In sum, the j^{th} row of $\boldsymbol{\Pi}$ identifies the cell that $\boldsymbol{X}_j$ is sampled from, while the corresponding (independently generated) uniform deviates determine the location of $\boldsymbol{X}_j$ within the sampled cell.

Example 5.1. Suppose $\boldsymbol{X} = (X_1, X_2)$ is uniformly distributed over $[0, 1]^2$ so that $F_k^{-1}(w) = w, 0 < w < 1$. To obtain an LH sample of size $n = 3$, compute

$$X_{jk} = F^{-1}\left(\frac{1}{3}\left(\Pi_{jk} - 1 + U_{jk}\right)\right) = \frac{1}{3}\left(\Pi_{jk} - 1 + U_{jk}\right), \quad j = 1, 2, 3;\ k = 1, 2.$$

The actual sample depends on the specific choice of $\boldsymbol{\Pi}$ and the $\{U_{jk}\}$.

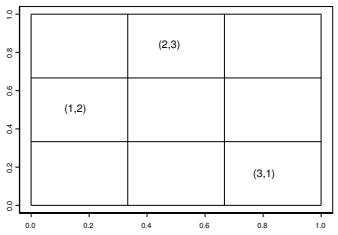


Fig. 5.1 Cells selected by the Latin hypercube sample (1,2), (2,3), and (3,1)

To envision the pattern of the LH sample, divide the unit interval in each dimension into [0,1/3), [1/3,2/3), and [2/3,1], yielding a partition of $[0,1]\times[0,1]$ into nine squares (cells) of equal area. In the LH sample, each of these subintervals will be represented exactly once *in each dimension*. For simplicity of discussion, suppose one labels these subintervals as 1, 2, and 3 in the order given above. One possible LHD would involve points randomly sampled from the (1,1), (2,3), and (3,2) squares and another possible design from the (1,2), (2,3), and (3,1) squares. Figure 5.1 plots the cells selected by the second design. These two selections correspond to the permutations

$$\boldsymbol{\Pi} = \begin{pmatrix} 1 & 1 \\ 2 & 3 \\ 3 & 2 \end{pmatrix} \text{ and } \boldsymbol{\Pi} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}. \tag{5.2.1}$$

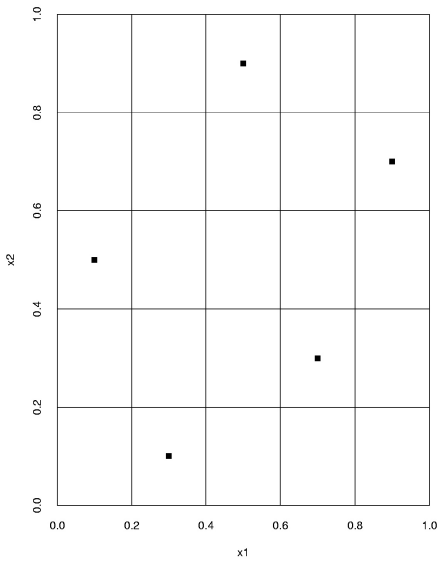
Note that in each dimension, each subinterval appears exactly once. Because each subinterval is of length $1/3$, the addition of $U_{jk}/3$ to the left-hand boundary of the selected subinterval serves merely to pick a specific point in it. ♦

In the computer experiment setting, the input variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ are not regarded as random for purposes of experimental design. As in Example 5.1, suppose that each input variable has been scaled to have domain [0,1]. Denoting the k^{th} component of $\boldsymbol{x}_j$ by x_{jk} for $k = 1, \ldots, d$, suppose one obtains an LHD from a given $\boldsymbol{\Pi}$ as follows:

$$x_{jk} = \frac{\Pi_{jk} - 0.5}{n}, \quad j = 1, \ldots, n_s;\ k = 1, \ldots, d\,.$$

This corresponds to taking $U_{jk} = 0.5$ for each $j = 1, \ldots, n_s$ and $k = 1, \ldots, d$ rather than as a sample from a $U[0,1]$ distribution. The "cells" are now identified with all d-dimensional Cartesian products of the intervals $\{(0, \frac{1}{n_s}], (\frac{1}{n_s}, \frac{2}{n_s}], \ldots, (1 - \frac{1}{n_s}, 1]\}$, and each $\boldsymbol{x}_j$ is sampled from the *center* of the cell indicated by the j^{th} row of $\boldsymbol{\Pi}$. An example of an LHD for $n_s = 5$ and $d = 2$ is given in Figure 5.2 with its associated $\boldsymbol{\Pi}$ matrix.

As mentioned previously, LHDs need not be space-filling over the full experimental region. To illustrate this point visually, consider the LHD for $n_s = 5$ and



$$\boldsymbol{\Pi} = \begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix}$$

Fig. 5.2 A space-filling Latin hypercube design and the corresponding permutation $\boldsymbol{\Pi}$.

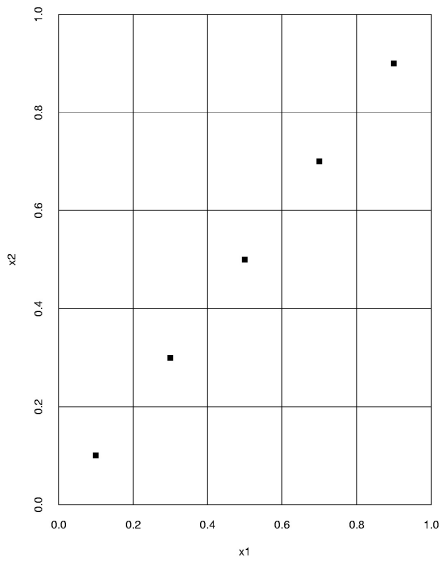
$d = 2$ that is shown in Figure 5.3, which one might *not* view as space-filling. One consequence of computing responses at this set of inputs is that one would expect a predictor fitted using this design to generally perform well only for $x_1 \approx x_2$. For example, consider the deterministic function

$$y(x_1, x_2) = \frac{x_1}{1 + x_2}, \quad \mathcal{X} = [0, 1] \times [0, 1]\,.$$

The MLE-EBLUP (Section **??**) was fitted to the observed responses using the training data for both of the designs shown in Figures 5.2 and 5.3. The predictor was based on the stochastic process

$$Y(x_1, x_2) = \beta_0 + Z(x_1, x_2),$$

where $Z(\cdot)$ is a zero mean Gaussian stochastic process with unknown process variance and product power exponential correlation function (**??**).



$$\boldsymbol{\Pi} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \end{pmatrix}$$

Fig. 5.3 A non space-filling Latin hypercube design

The prediction error $|y(x_1, x_2) - \widehat{Y}(x_1, x_2)|$ was calculated on a grid of 100 equally-spaced (x_1, x_2) points for each design. Figure 5.4 plots a comparison of the prediction errors for the two designs where the symbol "1" ("0") indicates that the prediction error for the design of Figure 5.3 is larger (smaller) than the prediction error for the design of Figure 5.2. The space-filling design of Figure 5.2 clearly yields a better predictor over most of the design space except for the diagonal where the LHD in Figure 5.3 collects most of its data.

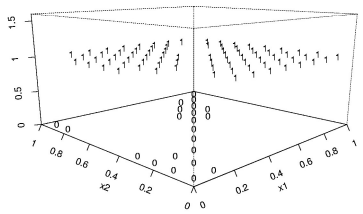


Fig. 5.4 Comparison of two LHDs. The plotting symbol "1" ("0") at location (x_1, x_2) means that the design in Figure 5.2 had lower (higher) mean squared prediction error than the design in Figure 5.3.

It is apparent from this discussion that although all LHDs possess desirable *marginal* properties, only a subset of these designs are truly "space-filling." Section 5.3 will discuss design criteria that have been successfully applied to select space-filling LHDs for use in computer experiments.

LHDs have been used extensively in the computer experiments literature; see, for example, Welch et al (1992) and Bernardo et al (1992). Other examples include Kennedy and O'Hagan (2001), Butler (2001), and Craig et al (2001). Because of their widespread use, it is worth examining in some detail the properties of LHDs in the setting, where all inputs are all environmental variables.

Designs based on LH sampling were introduced by McKay et al (1979) as a competitor to simple random sampling and stratified sampling when estimating the mean, variance, or distribution function of an output random variable. Stein (1987) and Owen (1992b) established additional large sample properties of LH sampling for estimating the mean $E\{Y\}$. Looking carefully at some of the results in these papers. will provide greater insight into the actual properties of LHDs. It will then be worthwhile to reconsider their use in computer experiments.

5.2.3 Properties of Sampling-Based Designs

Suppose that a random vector of inputs $\boldsymbol{X} = (X_1, \ldots, X_d)$ to the computer output $y(\cdot)$ is distributed according to the known joint distribution $F(\cdot)$ over the experimental region $\mathcal{X} \equiv [0,1]^d \subset \mathbb{R}^d$ (possibly after shifting and rescaling). Based on a sample $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_s}$ from the distribution $F(\cdot)$, one is interested in estimating the mean of $g(Y)$, assumed finite, where $Y = y(\boldsymbol{X})$ and $g(\cdot)$ is a known function of the real-valued argument. This mean is

$$\mu = \mathrm{E}\{g(Y)\} = \int_{\mathcal{X}} g(y(\boldsymbol{x}))dF(\boldsymbol{x}).$$

Now consider the properties of the naive moment estimator

$$T = T(y(\boldsymbol{X}_1), \ldots, y(\boldsymbol{X}_{n_s})) = \frac{1}{n_s} \sum_{j=1}^{n_s} g(y(\boldsymbol{X}_j))$$

when $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_s}$ are either a simple random sample, a stratified random sample, or a Latin hypercube sample. To derive the properties of T, assume that the coordinates of $\boldsymbol{X}$ are independent, each with cumulative distribution function $F(\cdot)$. Let

$$\sigma^2 = \text{Var}\{g(Y)\}.$$

For clarity denote the estimator T by T_R when simple random sampling is used, by T_S when stratified sampling is used, and by T_L when LH sampling is used. McKay et al (1979) show the following.

Theorem 5.1. 1. If proportional sampling is used, i.e., if the sample size for stratum i is proportional to the probability under $F(\cdot)$, of a point belonging to stratum i, then $\text{Var}\{T_S\} \leq \text{Var}\{T_R\}$.
2. If $y(x_1, \ldots, x_d)$ is monotonic in each of its arguments, and $g(w)$ is a monotonic function of $w \in \mathbb{R}$, then $\text{Var}\{T_L\} \leq \text{Var}\{T_R\}$.

Section **??** of the Chapter Notes provides a proof of the second part of this theorem.

At this point a few cautions are in order. First, these results show only that for estimating the expected value of $g(Y)$ over the experimental region, designs based on proportional sampling are better than those based on simple random sampling, and, under certain conditions, LHDs are better than those based on simple random sampling. Designs based on LH sampling need not always be better than designs based on simple random sampling nor is it known whether designs based on LH sampling are better than other types of designs, such as stratified sampling. Note, however, that the formulas derived inMcKay et al (1979) do allow one to compare designs based on LH and stratified proportional sampling.

Second, in most computer experiments one does not know the relationship between the output $y(\boldsymbol{x})$ and the component inputs $x_1, \ldots, x_d$. It is unlikely that one would be willing to assume this relationship is monotonic. And if one makes such an assumption, the conditions on $g(\cdot)$ given in the above theorem imply that the extrema of $g(\cdot)$ are on the boundary of the experimental region. If, as is often the case, one is interested in finding the extrema of $g(\cdot)$ and one knows the extrema are on the boundary of the experimental region, one would want to take observations near or on the boundary rather than using an LHD.

Third, the above properties are relevant if one is interested in estimating the expected value of $g(Y)$ over the experimental region. To illustrate, let $I\{E\}$ denote the indicator function as E (1 or 0, as E is true or false) and y_{fixed} be a given point in $\mathbb{R}$. Then setting $g(y) = y$ yields the mean of Y over the experimental region while setting $g(y) = I\{y \leq y_{fixed}\}$ produces the cumulative distribution function of Y at y_{fixed}. However, finding the expected value of $g(Y)$ over the experimental region is not usually the goal in computer experiments. More typically, the goal is to fit a model that approximates $g(\cdot)$ over the experimental region or to determine the points in the experimental region that are extrema of $g(\cdot)$. Thus, although LHDs are quite

popular in computer experiments, the above results do not indicate whether they have good properties in many of the situations where such computer experiments are conducted. Better justification for the use of LHDs comes from the results to be discussed next.

Additional properties of sample means based on Latin hypercube samples have been established by Stein (1987) and Owen (1992b). For simplicity, take $g(y) = y$ *for the remainder of this section* and use $\overline{Y} = \frac{1}{n_s}\sum_{j=1}^{n_s} y(\boldsymbol{X}_j)$ to estimate $\mathrm{E}\{y(\boldsymbol{X})\}$. Let $F_i(\cdot)$ denote the marginal distribution of X_i, the i^{th} coordinate of $\boldsymbol{X}$. As above, assume the coordinates of $\boldsymbol{X}$ are independent so

$$F(\boldsymbol{x}) = \prod_{i=1}^{d} F_i(x_i).$$

For $1 \le j \le d$, let $\boldsymbol{X}_{-j}$ denote $\boldsymbol{X}$ omitting X_j,

$$F_{-j}(\boldsymbol{x}_{-j}) = \prod_{i=1, i\neq j}^{d} F_i(x_i)$$

the distribution function of $\boldsymbol{X}_{-j}$, $\boldsymbol{x}_{-j}$ the corresponding argument extracted from $\boldsymbol{x}$, and $\mathcal{X}_{-j}$ denote the support of $F_{-j}(\cdot)$. Assuming $\int_{\mathcal{X}} y^2(\boldsymbol{x}) dF(\boldsymbol{x}) < \infty$, decompose $y(\boldsymbol{x})$ as follows. Define

$$\mu = \int_{\mathcal{X}} y(\boldsymbol{x})\, dF(\boldsymbol{x}) \quad \text{and} \quad \alpha_j(x_j) = \int_{\mathcal{X}_{-j}} [y(\boldsymbol{x}) - \mu]\; dF_{-j}(\boldsymbol{x}_{-j}).$$

Then μ is the overall mean, the $\{\alpha_j(x_j)\}$ are the "main effect" functions corresponding to the coordinates of $\boldsymbol{x}$, and $r(\boldsymbol{x}) = y(\boldsymbol{x}) - \mu - \sum_{i=1}^{d} \alpha_i(x_i)$ is the residual (from additivity) of $y(\boldsymbol{x})$. These quantities are continuous analogs of an "analysis of variance" decomposition of $y(\boldsymbol{x})$. Further reason for this designation is the fact that

$$\int_0^1 \alpha_j(x_j)\, dF_i(x_j) = 0 \quad \text{and} \quad \int_{\mathcal{X}_{-j}} r(\boldsymbol{x})\, dF_{-j}(\boldsymbol{x}_{-j}) = 0$$

for any x_j and all j (see also **??**)

Stein (1987) shows that for large samples, $\mathrm{Var}\left\{\overline{Y}\right\}$ is smaller under LH sampling than simple random sampling unless all main effect functions are 0. To be precise, Stein (1987) proves the following expansions for the variance of $\overline{Y}$ under the two sampling schemes.

Theorem 5.2. Under Latin hypercube sampling and simple random sampling we have

$$\text{Var}_{LHS}\left\{\overline{Y}\right\} = \frac{1}{n_s}\int_{\mathcal{X}} r^2(\boldsymbol{x})dF(\boldsymbol{x}) + o(n_s^{-1}) \quad \text{and}$$

$$\text{Var}_{SRS}\left\{\overline{Y}\right\} = \frac{1}{n_s}\int_{\mathcal{X}} r^2(\boldsymbol{x})dF(\boldsymbol{x}) + \frac{1}{n_s}\sum_{i=1}^{d}\int_{a_i}^{b_i} \alpha_i^2(x_i)dF_i(x_i) + o(n_s^{-1}),$$

respectively.

The implication of this expansion is that, unless all $\alpha_j(\cdot)$ are identically 0, in the limit, LH sampling has a smaller (order $1/n_s$) variance than simple random sampling.

Further, not only can the variance of Y be estimated but also the normality of $\overline{Y}$ can be established. For simplicity, assume $\mathcal{X} = [0,1]^d$ and that $F(\cdot)$ is uniform. More general cases can often be reduced to this setting by appropriate transformations. Owen (1992b) shows that $\overline{Y}$ computed from inputs based on LH sampling is approximately normally distributed for large samples. This can be used as the basis for statistical inference about μ. Owen (1992b) proves the following.

Theorem 5.3. If $y(\boldsymbol{x})$ is bounded, then under LH sampling, $\sqrt{n}(\overline{Y} - \mu)$ tends in distribution to $N\left(0, \int_{\mathcal{X}} r^2(\boldsymbol{x})d\boldsymbol{x}\right)$ as $n_s \to \infty$.

Owen (1992b) also provides estimators of the asymptotic variance

$$\int_{\mathcal{X}} r^2(\boldsymbol{x})\, d\boldsymbol{x}$$

to facilitate application of these results to computer experiments.

Section 5.7.2 of the Chapter Notes describes the use of LHDs in a generalization of these constant mean results to a regression setting, which has potential for use in computer experiments.

5.3 Latin Hypercube Designs with Additional Properties

Figure 5.3 displays an LHD that would probably not be considered space-filling in $[0,1]^2$ because the points lie along a straight line and are perfectly correlated. By comparison, the LHD in Figure 5.2 appears more space-filling in $[0,1]^2$ and the points appear much less correlated. ls it possible to identify special types of LHDs, or extensions of LHDs, that have desirable properties? For example, are there LHDs that are space filling in $\mathcal{X}$ or that have space-filling projections onto subspaces of dimension greater than 1? As in the previous section, we assume (after possible rescaling) that $\mathcal{X} = [0,1]^d$.

5.3.1 Extensions of Latin Hypercube Designs based on Orthogonal Arrays

One possible extension of LHDs, based on what are known as orthogonal arrays, produces designs with attractive projection properties. An *orthogonal array O on s symbols of strength t* is an $n_s \times d$ ($d \geq t$) matrix whose entries are the s symbols arranged so that in every $n_s \times t$ submatrix of O, all of the s^t possible rows appear the same number, say λ, of times; obviously $n_s = \lambda s^t$. Such an orthogonal array will be denoted by $OA(n_s, d, \lambda, s, t)$. For additional discussion regarding orthogonal arrays see Raghavarao (1971), Hedayat et al (1999), or Wu and Hamada (2009).

Example 5.2. The simplest type of orthogonal array is an orthogonal array of strength 1. Any $s \times d$ matrix whose columns are the integers $1, 2, \ldots, s$ in some order is an $OA(s, d, 1, s, 1)$. For example, if $s = 5$ and $d = 2$,

$$\begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \end{pmatrix}$$

are both $OA(5, 2, 1, 5, 1)$. Orthogonal arrays of strength > 1 are more challenging to construct.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{pmatrix}$$

is an $OA(9, 3, 1, 3, 2)$. ♦

For an orthogonal array of strength 1 with $\lambda = 1$, based on the s symbols $1, 2, \ldots, s$, let $O_{j,k}$ be the entry in row j and column k. Let

$$x_{jk} = \frac{O_{jk} - 0.5}{s}, \quad j = 1, \ldots, s;\ k = 1, \ldots, d.$$

Consider the matrix X whose entry in row j and column k is $x_{j,k}$. The s rows of X determine s points in the d-dimensional unit cube $[0,1]^d$. These points form an s-point LHD in $[0,1]^d$ (see the discussion following equation (5.2.1)). Notice that the two $OA(5,2,1,5,1)$ in Example 5.2 produce the LHDs in Figures 5.2 and 5.3. We see that orthogonal arrays of strength 1 and $\lambda = 1$ determine an LHD.

Because of this connection between orthogonal arrays of strength 1 and LHDs, it will not be completely surprising to learn that orthogonal arrays of strength 2 or greater can be used to define extensions of LHDs. In particular, orthogonal arrays of strength t can be used to generate designs with the property that all projections of the points in the design on to any t-dimensions are evenly distributed. Owen (1992a) describes a procedure for generating such designs, namely for generating n_s point space-filling designs in d dimensions from the columns of an $n_s \times d$ orthogonal array. The resulting designs are called *randomized orthogonal arrays*. If one plots the points of a randomized orthogonal array generated from an orthogonal array of strength t, in t or fewer of the coordinates, the result will be a regular grid. For details concerning randomized orthogonal arrays, see Owen (1992a) or Tang (1993). Example 5.3 illustrates the method in 3 dimensions based on an orthogonal array of strength 2. Notice that an LHD is just a randomized orthogonal array based on an orthogonal array of strength 1 and with $\lambda = 1$.

Example 5.3. An example of a randomized orthogonal array is the following. Suppose $d = 3$ and we take $n_s = 9$, $s = 3$, $t = 2$, and $\lambda = 1$. The orthogonal array on three symbols of strength two is the 9×3 matrix that was presented in Example 5.2, namely

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{pmatrix}$$

To construct a randomized orthogonal array, use this 9×3 matrix. Divide the unit cube $[0,1]\times[0,1]\times[0,1]$ into a $3\times3\times3$ grid of 27 cells (cubes). Let (1,1,1) denote the cell (cube) $[0, \frac{1}{3}]\times[0, \frac{1}{3}]\times[0, \frac{1}{3}]$, (1,1,2) denote the cell $[0, \frac{1}{3}]\times[0, \frac{1}{3}]\times[\frac{1}{3}, \frac{2}{3}]$, (1,1,3) denote the cell $[0, \frac{1}{3}] \times [0, \frac{1}{3}] \times [\frac{2}{3}, 1]$, ..., and (3,3,3) the cell $[\frac{2}{3}, 1] \times [\frac{2}{3}, 1] \times [\frac{2}{3}, 1]$. Each row of the above 9×3 matrix corresponds to one of these 27 cells. The points in the centers of the nine cells determined by the rows of the matrix yields a nine point randomized orthogonal array. Projected onto any two-dimensional subspace, the design looks like a regular 3×3 grid. Projected onto any one-dimensional subspace, the design looks like a regular grid at three locations, with three points projecting onto each location (hence the one-dimensional projections are not space-filling in the way in which LHDs are).

Instead of selecting the points in the centers of the nine cells, one could select a point a random from each of these cells. The resulting projections onto two-dimensional subspaces would not be a regular grid, but would be evenly spaced in each of the two-dimensional subspaces. This would also prevent the one-dimensional projections from yielding multiple points at the same locations. ♦

Although randomized orthogonal arrays extend the projection properties of LHDs to more than one dimension, they have the drawback that they only exist for certain values of n_s, namely for $n_s = \lambda s^t$, and only for certain values of d. Also, because $n_s = \lambda s^t$, only for relatively small values of s and t will the designs be practical for use in computer experiments in which individual observations are time-consuming to obtain and hence for which n_s must be small. See Tang (1993) and Tang (1994) for additional information about the use of randomized orthogonal arrays in computer experiments.

Scatterplots of uncorrelated variables produce plots for which points appear more space-filling than plots involving variables that are highly correlated. This suggests that a possible strategy for avoiding LHDs that do not appear to be space-filling in $[0, 1]^d$ is to select those for which the points are uncorrelated. Owen (1992a) and Tang (1993) discuss methods for constructing an LHD with an underlying orthogonal array structure, thus assuring that in some dimension the points in the LHD appear uncorrelated. Tang (1993) uses an orthogonal array to specify a restricted permutation of $\{1, 2, \ldots, n_s\}$ in selecting the matrix $\boldsymbol{\Pi}$ from which the LHD is formed. Let $OA(n_s, d, 1, s, t)$ be an $n_s \times d$ orthogonal array on s symbols of strength t with $\lambda = 1$. For the kth column of $OA(n_s, d, 1, s, t)$ let $r_{k,l+1}$ be the number of rows with entry l. Note that $r_{k,l_1+1} = r_{k,l_2+1} = r$ for all l_1 and l_2 so that $rs = n_s$. We can form a $\boldsymbol{\Pi}$ based on $OA(n_s, d, 1, s, t)$ by selecting each column of $OA(n_s, d, 1, s, t)$ and replacing the r entries of $OA(n_s, d, 1, s, t)$ with level l by a permutation of the integers $rl + 0, rl + 1, \ldots, rl + (r - 1)$ for all $l = 1, 2, \ldots, s$. The LHD formed from this $\boldsymbol{\Pi}$ will have all univariate projections uniformly distributed and all t-variate projections uniformly distributed. Tang (1993) refers to an LHD constructed in this way as an OA-based LHD.

Another way to think about an OA-based LHD is that the structure of the orthogonal array is used to restrict the placement of the points within the unit hypercube (assume for this discussion that we are interested in LHDs on the d-dimensional unit hypercube). In the context of the previous discussion, for the kth column of $OA(n)s, d, 1, s, t)$ we consider the non-overlapping division of $[0, 1]$ into s equal length intervals of the form $[0, \frac{1}{s}) \cup [\frac{1}{s}, \frac{2}{s}) \cup \cdots \cup [\frac{s-1}{s}, 1]$. Because $OA(n_s, d, 1, s, t)$ is an orthogonal array, each of the s symbols appears equally often in each column and we let r denote the number of times each symbol appears in a given column. For a given level $l_j = 0, 1, \ldots, s - 1$ we define the non-overlapping division of the interval $[\frac{l_j}{s}, \frac{l_j+1}{s})$ into r subintervals of the form

$$[\frac{l_j}{s} + \frac{i}{sr}, \frac{l_j}{s} + \frac{i+1}{sr}), i = 0, 1, \ldots, r - 1.$$

For column k let $p_{k_1}, p_{k_2}, \ldots, p_{k_r}$ be a random permutation of the integers $0, 1, \ldots, r-1$. Then the r points corresponding to level l_j are randomly (or systematically) placed one each in the Cartesian product intervals

$$[\frac{l_j}{s} + \frac{p_{k_i}}{sr}, \frac{l_j}{s} + \frac{p_{k_i}+1}{sr}), k = 1, 2, \ldots, d.$$

Notice for each column of $OA(n_s, d, 1, s, t)$, $n_s = rs$ and the Latin hypercube intervals $[\frac{i}{n_s}, \frac{i+1}{n_s})$ are identical to the substratification described so that the resulting array, with placement of points imposed by the strength t orthogonal array is indeed an LHD with t-dimensional projection properties consistent with $OA(n_s, d, 1, s, t)$.

Example 5.4. Suppose we start with the $OA(4, 2, 1, 2, 2)$

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{pmatrix}$$

To obtain $\boldsymbol{\Pi}$, in each column we replace the symbol 1 with a random permutation of the integers 1, 2, and the symbol 2 with a random permutation of the integers 3,4. For example, in column 1 replace the first occurrence of the symbol 1 by 1 and the second occurrence of the symbol 1 by 2. Replace the first first occurrence of the symbol 2 in column 1 by 4 and the second occurrence of the symbol 2 by 3. In column 1 replace the first occurrence of the symbol 1 by 2 and the second occurrence of the symbol 1 by 1. Replace the first first occurrence of the symbol 2 in column 2 by 3 and the second occurrence of the symbol 2 by 4. This gives,

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 4 & 1 \\ 3 & 4 \end{pmatrix}$$

♦

Example 5.5. Suppose we start with the $OA(9, 3, 1, 3, 2$

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{pmatrix}$$

To obtain $\boldsymbol{\Pi}$, in each column we replace the symbol 1 with a random permutation of the integers 1, 2, 3, the symbol 2 with a random permutation of the integers 4, 5, 6, and the symbol 3 by a random permutation of the integers 7, 8, 9. For example, use (1, 3, 2), (3, 2, 1), and (2, 1, 3) for the 1's in columns 1, 2, and 3, respectively. Use (6, 5, 4), (4, 5, 6), and (5, 4, 6) for the 2's in columns 1, 2, and 3, respectively. Finally use (9, 7, 8), (8, 7, 9), and (7, 9, 8) for the 3's in columns 1, 2, and 3, respectively. This gives,

$$\begin{pmatrix} 1 & 3 & 2 \\ 3 & 4 & 5 \\ 2 & 8 & 7 \\ 6 & 2 & 4 \\ 5 & 5 & 9 \\ 4 & 7 & 1 \\ 9 & 1 & 8 \\ 7 & 6 & 3 \\ 8 & 9 & 6 \end{pmatrix}$$

♦

Note that in the step for constructing $\boldsymbol{\Pi}$ from the initial orthogonal array, many choices for the permutations are possible, hence from a given initial orthogonal array, many $\boldsymbol{\Pi}$ can be constructed. One can impose an additional criterion to select one of these $\boldsymbol{\Pi}$, thus insuring that the final LHD has an additional desirable property.

The orthogonal array structure imposed on the design is appealing in that it leads to uniformity in all t-variate projections when the strength of the orthogonal array is t. This helps achieve an additional degree of space-fillingness not easily achieved by a random LHD or one that is numerically optimized according to some criterion.

An important drawback is that OA-based LHDs are limited in the run sizes that are possible. For example, even starting with an orthogonal array on 2 symbols of strength 2 the run size must be a multiple of 4. In addition, the method of construction is not always readily adaptable to algorithmic generation. For these reasons Loeppky et al (2012) introduce a more flexible class of designs, called projection array based designs, that have space-filling properties analogous to OA-based LHDs, but exist for all run sizes.

5.3.2 Cascading, Nested, and Sliced Latin Hypercube Designs

Cascading LHDs are another extension of LHDs. Cascading LHDs are introduced in Handcock (1991) and can be described as follows. Generate an LHD. At each point of this design, consider a small region around the point. In this small region, generate a second LHD. The result is a cluster of small LHDs and is called a *cascading Latin hypercube design*. Such designs allow one to explore both the local (in small subregions) and the global (over the entire experimental region) behavior of the response.

Nested and sliced LHDs are additional extensions of LHDs. Suppose one uses an LHD consisting of n_s points in $\mathbb{R}^d$. After fitting a predictor to the data, suppose one decides the fit is inadequate and m_s additional runs of the computer simulator are necessary. Is it possible to select the m_s runs in such a way that the resulting set of $n_s + m_s$ runs is an LHD? In general, the answer is no. Figure 5.5 displays a 2-point LHD in two dimensions with the two points randomly placed in two of the four cells (outlined by the solid lines). This cannot be extended to a 3-point LHD in two dimensions, because both points are in the same cell when the design space is partitioned into nine cells (outlined by the dashed lines). However, the 2-point LHD could be extended to a 4-point LHD in two dimensions because the two points would now be in two separate cells when the design space is partitioned into 16 cells.

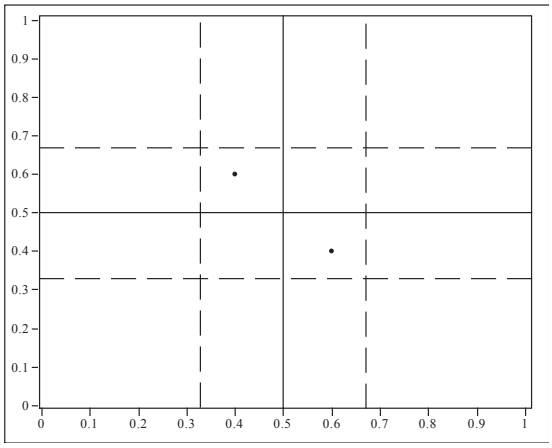


Fig. 5.5 A 2-point LHD that cannot be extended to a 3-point LHD. Points are placed at random in the four cells for a 2-point LHD. The cells are outlined by the solid lines. The dashed lines outline the nine cells for a 3-point LHD. Notice both points are in the same cell.

Notice that if in the original LHD the points were chosen at random in the n_s cells, and if $m_s = an_s$ for some positive integer a, it is possible to add the m_s points in such a way the $m_s + n_s$ points are an LHD. In the initial LHD, the domain of each input variable was subdivided into n_s intervals. Subdivide each of these n_s intervals into $a + 1$ intervals so that now the domain of each input variable is subdivided into $(a+1)n_s$ intervals. The Cartesian product of these intervals constitutes a partitioning of the d-dimensional sample space into $[(a+1)n_s]^d$ cells. Each of the n_s points in the original design are in exactly one of these cells. Choose a subset of $(a+1)n_s$ cells in such a way that they include the n_s cells containing points from the original design and so that the projections of the centers of all $(a+1)n_s$ points onto each component axis yield $(a + 1)n_s$ distinct points on the axis. In terms of the method described before Example 5.1, this will mean that one can select only certain $(a + 1)n_s \times d$ matrices $\boldsymbol{\Pi}$ having permutations of $\{1, 2, \ldots, (a + 1)n_s\}$ as columns. Notice that if

in the original LHD the points were chosen at the center of the n_s cells, it is still possible to add m_s points in such a way that the resulting design is an LHD with points at the center of cells, provided a is even.

Instead of adding points to an existing LHD in such a way that the result is also an LHD, one could consider the "reverse" problem. Do there exist LHDs with the property that they can be divided into several smaller LHDs? One possibility is to generate an an_s-point LHD as described in the previous paragraphs. First generate an n_s-point LHD. Add $(a-1)n_s$ points as described in the previous paragraphs to generate an an_s-point LHD. By construction, the resulting design is both an LHD and contains a subset of n_s points (the starting design) which is also an LHD. More generally, one could start with an n_s-point LHD, extend it to an $a_1 n_s$-point LHD, then extend this $a_1 n_s$-point LHD to an $a_1 a_2 n_s$-point LHD, and continue on to an $a_1 a_2 \cdots a_b n_s$-point LHD. The final design contains subsets of points that are n_s-point, $a_1 n_s$-point, $a_1 a_2 n_s$-point, ..., and $a_1 a_2 \cdots a_{b-1} n_s$-point LHDs.

In the literature, two types of designs, nested LHDs and sliced LHDs, have also been proposed to accomplish this. Qian (2009) introduces nested LHDs. A nested LHD with n_s runs and a layers is an LHD with the property that it can be used to generate a series of $a-1$ successively smaller LHDs. Qian (2009) also investigates the properties of nested LHDs for estimating the means of functions and shows that nested LHDs can outperform i.i.d. sampling under conditions analogous to those in Theorem 5.1.

One application of nested LHDs is to computer experiments involving codes with multiple levels of accuracy, in which experiments with higher levels of accuracy are more expensive (and hence are observed at fewer points) than those of lower accuracy. An n_s-run nested LHD can be used to determine the points at which the lowest accuracy experiment is run. The successively smaller layers of this LHD can be used to determine the points at which the successively higher accuracy experiments are run. Using a nested LHD guarantees that the points at which any particular level-of-accuracy experiment is run are also observed at all lower accuracy experiments. See Kennedy and O'Hagan (2000), Qian et al (2006), and Qian and Wu (2008) for more on such experiments

Qian (2012) constructs LHDs for d inputs with $n_s = am_s$ runs having the property that the LHD can be subdivided into a LHDs for d inputs with m_s runs each. These a divisions of the n_s-run LHD are called slices and the original n_s-run LHD a sliced Latin hypercube design. Qian (2012) also shows that sliced LHDs can outperform both i.i.d. sampling and standard LH sampling in terms of variance reduction in settings where one wishes to estimate a weighted average of expected values of a functions under conditions analogous to those in Theorem 5.1.

One application of sliced LHDs is to computer experiments with both quantitative and qualitative variables. This is discussed in Section 2.4. Each slice provides the design (values of the quantitative variables to be observed) for one set of values of the qualitative variables. In addition, should the qualitative variables have no significant effect on the response, the slices collapse into a larger LHD.

Two recent extensions of LHDs can be found in Ba and Joseph (2011) and Joseph et al (2015). Ba and Joseph (2011) discuss a class of designs, called multi-layer

designs, that have good space-filling properties. These designs are an alternative to LHDs and are developed by splitting two-level factorial designs into multiple layers. Joseph et al (2015) introduce a maximum projection criterion that produces LHDs with good projection properties.

5.3.3 Orthogonal Latin Hypercube Designs

Another attempt to find LHDs that have additional good properties is due to Ye (1998). He discusses a method for constructing LHDs for which all pairs columns are orthogonal to each other, where a pair of columns is orthogonal if their inner product is zero. In general, the columns of an $n_s \times d$ LHD are formed from an $n_s \times d$ matrix $\boldsymbol{\Pi}$ whose columns are permutations of the integers $\{1, 2, \ldots, n_s\}$, as discussed in Subsection 5.2.2. By restricting to only certain permutations of $\{1, 2, \ldots, n_s\}$, Ye (1998) is able to generate LHDs with orthogonal columns, which he calls orthogonal Latin hypercubes (OLHs). The method of construction, in particular the set of permutations needed to generate orthogonal columns, for $n_s = 2^{m-1}$ and$d = 2m - 2$, and where > 1 is an integer, is as follows.

Let $\mathbf{e}$ be the $2^{m-1} \times 1$ column vector with entries $\{1, 2, \ldots, 2^{m-1}\}$. Ye (1998) uses the notation $(r \quad s)$ to represent the permutation of rows of $\mathbf{e}$ (more generally, the permutation of the rows of any column vector) obtained by transposing rows r and s. For example, if $m = 3$, $(1 \quad 3)$ of $\mathbf{e} = (1, 2, 3, 4)^\top$ is $(3, 2, 1, 4)^\top$. Products of such permutations denote the permutation resulting from applying each in turn, i.e. applying their composition. For example, $(2 \quad 4)(1 \quad 3)$ of $\mathbf{e} = (1, 2, 3, 4)^\top$ is $(2 \quad 4)$ of $(3, 2, 1, 4)^\top$, namely $(3, 4, 1, 2)^\top$.

For a given integer $m > 1$, Ye (1998) defines the $m - 1$ permutations

$$A_k = \prod_{j=1}^{2^{m-k-1}} \left\{ \prod_{i=1}^{2^{k-1}} ((j-1)2^k + i \quad j2^k + 1 - i) \right\}, k = 1, \ldots, m-1 \tag{5.3.1}$$

where $\prod$ represents the product (or composition) of permutations.

For example, if $m = 3$,

$$\begin{aligned} A_1 &= \prod_{j=1}^{2} ((j-1)2 + 1 \quad j2) \\ &= (1 \quad 2)(3 \quad 4) \end{aligned}$$

$$\begin{aligned} A_2 &= \prod_{i=1}^{2} (i \quad 4 + 1 - i) \\ &= (1 \quad 4)(2 \quad 3) \end{aligned}$$

Next, let $\mathbf{M}$ be the $2^{m-1} \times (2m - 2)$ matrix with columns

$$\{\mathbf{e}, \mathbf{A}_1\mathbf{e}, \ldots, \mathbf{A}_{m-1}\mathbf{e}, \mathbf{A}_{m-1}\mathbf{A}_1\mathbf{e}, \ldots, \mathbf{A}_{m-1}\mathbf{A}_{m-2}\mathbf{e}\}.$$

For example, if $m = 3$, $\mathbf{M}$ would be the 4×4 matrix

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 2 & 1 \\ 4 & 3 & 1 & 2 \end{pmatrix}.$$

For each integer k between 1 and $m - 1$, let

$$\mathbf{B}_{m-k} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \mathbf{B}_i = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

for all $i \neq m - k$ and define

$$\mathbf{a}_k = \{\bigotimes_{j=1}^{m-1} \mathbf{B}_j\}$$

where $\bigotimes$ is the Kronecker product.

For example, if $m = 3$, with $k = 1$,

$$\mathbf{a}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \bigotimes \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

and with $k = 2$,

$$\mathbf{a}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \bigotimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

Use $\odot$ to denote the element wise product of two vectors, for example,

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \odot \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

Next, denote the $2^{m-1} \times 1$ vector of 1s by $\mathbf{1}$. Ye (1998) defines $\mathbf{S}$ to be the $2^{m-1} \times (2m - 2)$ matrix with columns

$$\{\mathbf{1}, \mathbf{a}_1, \ldots, \mathbf{a}_{m-1}, \mathbf{a}_1 \odot \mathbf{a}_2, \ldots, \mathbf{a}_1 \odot \mathbf{a}_{m-1}\}.$$

For example, when $m = 3$,

$$\mathbf{S} = \{\mathbf{1}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 \odot \mathbf{a}_2\} = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Let $\mathbf{T} = \mathbf{M} \odot \mathbf{S}$. For example, when $m = 3$,

$$\mathbf{T} = \mathbf{M} \odot \mathbf{S} = \begin{pmatrix} 1 & -2 & -4 & 3 \\ 2 & 1 & -3 & -4 \\ 3 & -4 & 2 & -1 \\ 4 & 3 & 1 & 2 \end{pmatrix}.$$

Consider the $(2^m + 1) \times (2m - 2)$ matrix $\mathbf{O}$ whose first 2^{m-1} rows are $\mathbf{T}$, whose next row consists of all 0s, and whose last 2^{m-1} rows are the "mirror image" of $\mathbf{T}$, namely the rows of $-\mathbf{T}$ in reverse order. For example, when $m = 3$,

$$\mathbf{O} = \begin{pmatrix} 1 & -2 & -4 & 3 \\ 2 & 1 & -3 & -4 \\ 3 & -4 & 2 & -1 \\ 4 & 3 & 1 & 2 \\ 0 & 0 & 0 & 0 \\ -4 & -3 & -1 & -2 \\ -3 & 4 & -2 & 1 \\ -2 & -1 & 3 & 4 \\ -1 & 2 & 4 & -3 \end{pmatrix}$$

From $\mathbf{O}$ remove the row consisting of all 0s and rescale levels to be equidistant. Let $\mathbf{O}^*$ denote the resulting $2^m \times (2m - 2)$ matrix. For example, when $m = 3$,

$$\mathbf{O}^* = \begin{pmatrix} 0.5 & -1.5 & -3.5 & 2.5 \\ 1.5 & 0.5 & -2.5 & -3.5 \\ 2.5 & -3.5 & 1.5 & -0.5 \\ 3.5 & 2.5 & 0.5 & 1.5 \\ -3.5 & -2.5 & -0.5 & -1.5 \\ -2.5 & 3.5 & -1.5 & 0.5 \\ -1.5 & -0.5 & 2.5 & 3.5 \\ -0.5 & 1.5 & 3.5 & -2.5 \end{pmatrix}$$

Ye (1998) shows that the columns of $\mathbf{O}$ are orthogonal to each other, the elementwise square of each column of $\mathbf{O}$ is orthogonal to all the columns of $\mathbf{O}$, and that the elementwise product of every two columns of $\mathbf{O}$ is orthogonal to all columns in $\mathbf{O}$. In other words, if $\mathbf{O}$ is used as the design matrix for a second-order response

surface, all estimates of linear, bilinear, and quadratic effects are uncorrelated with the estimates of linear effects. The same holds true for $\mathbf{O}^*$.

Note that the elements of $\mathbf{O}$ and of $\mathbf{O}^*$ are no longer positive integers. However, each column is a permutation of the entries in the first column, hence both can be considered LHDs.

Ye (1998) also shows that the construction described above can be modified to yield additional OLHs. First, one can replace $\mathbf{e}$ by any of its permutations. Second, one can reverse any of the signs of any subset of columns of $\mathbf{O}$ or $\mathbf{O}^*$. The resulting arrays are all OLHs in the sense of having all the properties mentioned prior to Example 5.3.

5.3.4 Symmetric Latin Hypercube Designs

Unfortunately, OLHs exist only for very limited values of n_s, namely $n_s = 2^m$ or $n_s = 2^m + 1, m \geq 2$. Ye et al (2000) introduce a more general class of LHDs, called symmetric LHDs, to overcome this limitation. An LHD is called a symmetric LHD if it has the following property: in an $n_s \times d$ LHD with levels $1, 2, \ldots, n_s$, if $(a_1, a_2, \ldots, a_d)$ is one of the rows, then $(n_s + 1 - a_1, n_s + 1 - a_2, \ldots, n_s + 1 - a_d)$ must be another row. Ye et al (2000) do not discuss the construction of symmetric LHDs, but when n_s is an *even* integer, one obtains a symmetric LHD as follows. The first row can be any $1 \times d$ vector $(a_{11}, a_{12}, \ldots, a_{1d})$ where the $(a_{1j}$ are elements of $\{1, 2, \ldots, n_s\}$. The second row is $(n_s + 1 - a_{11}, n_s + 1 - a_{12}, \ldots, n_s + 1 - a_{1d})$. The third row can be any $1 \times d$ vector $(a_{31}, a_{32}, \ldots, a_{3d})$ where a_{3j} can be any of the integers $1, 2, \ldots, n_s$ that is not equal to either a_{1j} or $n_s + 1 - a_{1j}$. The fourth row is $(n_s + 1 - a_{31}, n_s + 1 - a_{32}, \ldots, n_s + 1 - a_{3d})$. Continue on in this manner, adding the odd rows so that the entries in column j have not year appeared in the previous rows of the column. The even rows have entries $n_s + 1$ minus the entry in the previous row.

When n_s is an *odd* integer, let the first row be $(\frac{n_s+1}{2}, \frac{n_s+1}{2}, \ldots, \frac{n_s+1}{2})$. The second row can be any $1 \times d$ vector $(a_{21}, a_{22}, \ldots, a_{2d})$ where the a_{2j} are elements of $\{1, 2, \ldots, n_s\}$ except $\frac{n_s+1}{2}$. The third row is $(n_s + 1 - a_{21}, n_s + 1 - a_{22}, \ldots, n_s + 1 - a_{2d})$. The fourth row can be any $1 \times d$ vector $(a_{41}, a_{42}, \ldots, a_{4d})$ where a_{4j} can be any of the integers $1, 2, \ldots, n_s$ that is not equal to $\frac{n_s+1}{2}$, a_{2j} or $n_s + 1 - a_{3j}$. Continue on in this manner, adding the even rows so that the entries in column j have not year appeared in the previous rows of the column. The odd rows have entries $n_s + 1$ minus the entry in the previous row.

Note that the non space-filling LHD in Figure 5.3 is a symmetric LHD, so symmetric LHDs need not be "good" LHDs.

Example 5.6. To construct a symmetric LHD with $n_s = 10$ (an even integer) and $d = 3$, suppose we begin with the row $(1, 6, 6)$. Following the algorithm described previously, we might obtain the following symmetric LHD.

$$\begin{pmatrix} 1 & 6 & 6 \\ 10 & 5 & 5 \\ 2 & 2 & 3 \\ 9 & 9 & 8 \\ 3 & 1 & 9 \\ 8 & 10 & 2 \\ 4 & 3 & 4 \\ 7 & 8 & 7 \\ 5 & 7 & 1 \\ 6 & 4 & 10 \end{pmatrix}$$

To construct a symmetric LHD with $n_s = 9$ (an odd integer) and $d = 3$, suppose we begin with rows $(5, 5, 5)$ and $(1, 6, 6)$. Following the algorithm described previously, we might obtain the following symmetric LHD.

$$\begin{pmatrix} 5 & 5 & 5 \\ 1 & 6 & 6 \\ 9 & 4 & 4 \\ 2 & 2 & 3 \\ 8 & 8 & 7 \\ 3 & 1 & 9 \\ 7 & 9 & 1 \\ 4 & 3 & 8 \\ 6 & 7 & 2 \end{pmatrix}$$

♦

Ye et al (2000) point out that symmetric LHDs have certain orthogonality properties. In a polynomial response surface, least squares estimation of the linear effect of each variable is uncorrelated with all quadratic effects and bi-linear interactions (but not necessarily with the linear effects of other variables). This follows from results in Ye (1998) because OLHs have the same symmetry properties as symmetric LHDs but also possess additional orthogonality that guarantees that linear effects are uncorrelated.

These orthogonality properties of OLHs and symmetric LHDs are useful if one plans to fit second order or higher response surface models to the data using standard least squares. However, if one intends to fit a predictor, such as the EBLUP discussed in Chapter 3, in which the generalized least squares estimate of the regression parameters is used, the benefits of orthogonality are less clear.

Symmetric LHDs form a subclass of all LHDs. As we discuss later, one can apply additional criteria to select a particular design from the class of all $n_s \times d$ LHDs, from the class of all $n_s \times d$ OLHs, or from the class of all $n_s \times d$ symmetric LHDs. For the latter, Ye et al (2000) propose a column-wise exchange algorithm that replaces a symmetric LHD with another symmetric LHD, allowing one to search the class of $n_s \times d$ symmetric LHDs for a design that optimizes some additional property of the design.

LHDs that are optimal under some additional criterion are often symmetric LHDs. When searching for an LHD that is optimum under some additional criterion, restricting the search to the smaller class of symmetric LHDs will often yield the global optimum (optimum over the class of all LHDs). This strategy was first proposed in Park (1994).

5.4 Designs Based on Measures of Distance

In this subsection, we consider criteria for selecting a design that are based on a measure or metric that quantifies the spread of a set of points. For all distance-based criteria discussed below, the domain of each input is normalized to the interval [0,1] otherwise inputs with larger ranges can dominate the computation of a maximin design, say. Thus, if the input space in the original problem is

$$\prod_{\ell=1}^{d} [a_\ell, b_\ell]$$

then

$$x_\ell = \frac{x_\ell - a_\ell}{b_\ell - a_\ell}, \quad \ell = 1, \ldots, d$$

is used to scale and shift the input space to $[0, 1]^d$; the inverse transform is used to place the computed design on the scale of the original design problem.

The first way in which points in a design $\mathcal{D}$ might be regarded as spread out over a design space $\mathcal{X}$ is for no point in $\mathcal{X}$ to be "too far" from a point in the design $\mathcal{D}$. To make this precise, again let $\rho_p(\cdot, \cdot)$ be a metric on $\mathcal{X}$. Denote the distance between an arbitrary input site $\boldsymbol{x} \in \mathcal{X}$ and a design $\mathcal{D} \subset \mathcal{X}$ by $\rho_p(\boldsymbol{x}, \mathcal{D})$, where

$$\rho_p(\boldsymbol{x}, \mathcal{D}) = \min_{\boldsymbol{x}_i \in \mathcal{D}} \rho_p(\boldsymbol{x}, \boldsymbol{x}_i)\,.$$

An n_s-point design $\mathcal{D}_{mM}$ is defined to be a *minimax distance design* if the maximum distance between arbitrary points $\boldsymbol{x} \in \mathcal{X}$ and the candidate design $\mathcal{D}_{mM}$ is a minimum over all designs $\mathcal{D}$ whose input vectors $\boldsymbol{x}_\ell \in \mathcal{X}$, $\ell = 1, \ldots, n_s$ namely

$$\max_{x \in \mathcal{X}} \rho(\boldsymbol{x}, \mathcal{D}_{mM}) = \min_{\mathcal{D}} \max_{x \in \mathcal{X}} \rho(\boldsymbol{x}, \mathcal{D}. \tag{5.4.1}$$

If the goal of a computer experiment is good prediction over all of $\mathcal{X}$, and if prediction variance at a point $\boldsymbol{x}_0$ increases as the distance between $\boldsymbol{x}_0$ and $\mathcal{D}$ increases, intuitively a design $\mathcal{D}$ for which no point is far from any $\boldsymbol{x}_0 \in \mathcal{X}$ should perform well. In other words, a minimax design would seem to be a sensible choice if the goal is good prediction (minimizing the maximum prediction variance) over $\mathcal{X}$. The difficulty is that finding minimax designs involves computing the maximum distance between a candidate design $\mathcal{D}$ and all points in $\mathcal{X}$. This is computationally challenging. One might try to find an approximately minimax design by restricting

the computation to a finite grid of points in $\mathcal{X}$. See Tan (2013) for a discussion of this approach.

A second method to measure the spread of n_s points in a design is by the distance of the closest two points in the design. To simultaneously define distances for both rectangular and non-rectangular input regions $\mathcal{X}$ (Section 5.5); let ρ denote an arbitrary metric on $\mathcal{X}$. Let $\mathcal{D}$ be an n_s-point design consisting of *distinct* input sites $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ with $\boldsymbol{x}_\ell \in \mathcal{X}$, $\ell = 1, \ldots, n_s$. For example, one important distance measure is p^{th} order distance between $\boldsymbol{w}, \boldsymbol{x} \in \mathcal{X}$ for $p \geq 1$, which is defined by

$$\rho_p(\boldsymbol{w}, \boldsymbol{x}) = \left[\sum_{j=1}^{d} |w_j - x_j|^p \right]^{1/p}. \tag{5.4.2}$$

Rectangular ("Manhattan") and Euclidean distances are the cases $p = 1$ and $p = 2$, respectively. Then one way to measure of the closeness of the n_s points in $\mathcal{D}$ is the smallest distance between any two points in $\mathcal{D}$, i.e.,

$$\min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{5.4.3}$$

A design that maximizes (5.4.3) is said to be a *maximin distance design* and is denoted by $\mathcal{D}_{Mm}$; thus

$$\min_{x_1, x_2 \in \mathcal{D}_{Mm}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \max_{\mathcal{D} \subset \mathcal{X}} \min_{x_1, x_2 \in \mathcal{D}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{5.4.4}$$

In an intuitive sense, therefore, $\mathcal{D}_{Mm}$ designs guarantee that no two points in the design are too close, and hence the design points are spread over $\mathcal{X}$.

One criticism of the maximin principle is that it judges the goodness of a design by the minimum among all $\binom{n_s}{2}$ input vectors rather than using all possible differences. Figure 5.6 illustrates such a pair of designs both of which have as their three smallest minimum interpoint distances 0.300, 0.361, and 0.412. The only difference in the two designs is that the point $(0.2, 0.2)$ in the left panel design has been moved to $(0.025, 0.025)$ in the right panel, but because of this change, the design in the right panel is, intuitively, more space-filling than the design in the left panel. More careful inspection of these designs shows that the fourth smallest interpoint distance is greater for for right panel design than the left panel design. By using a more comprehensive definition of minimaxity in which the number of pairs of the inputs with smallest, second smallest etc distances are accounted for, Morris and Mitchell (1995) were able to rank cases of equal minimum interpoint distance and eliminate such anomalies. Another criterion that accounts for the distances among all pairs of design vectors is the average of all $\binom{n_s}{2}$ interpoint distances, as will be introduced below.

In sum, despite this initial criticism, Mm designs are often visually attractive and can be justified theoretically under certain circumstances (Johnson et al (1990)).

Example 5.7. Figure 5.7 displays four-point maximin and minimax designs with Euclidean distance ($p = 2$). The differences in the two designs reflect the effect of

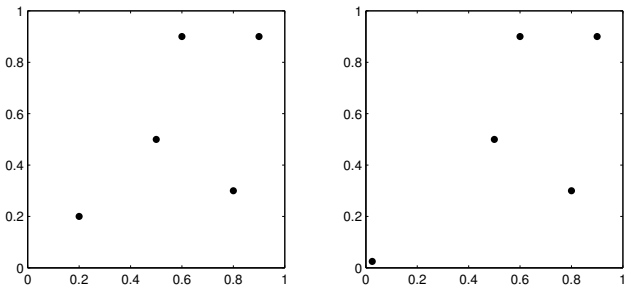


Fig. 5.6 Two designs on $[0, 1]^2$ with the same minimum interpoint distance of 0.30.

the different design criteria. Maximin designs tend to push points out towards the boundary of $\mathcal{X}$. This is not surprising because fthere is more "space" to spread out points at the boundary.

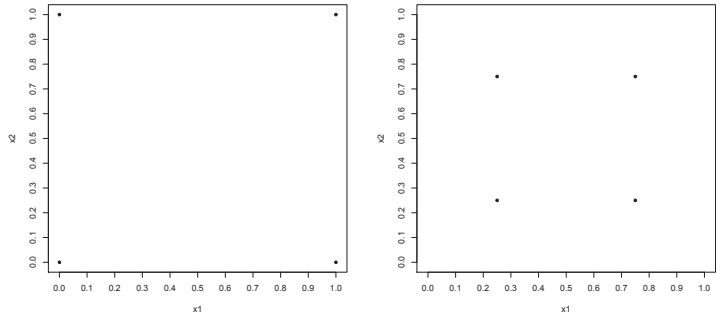


Fig. 5.7 Four-point Maximin design with respect to Euclidean distance ($p = 2$) (left panel) and four-point minimax design (right panel)

Another approach to spreading points in the design space is to consider the distribution of distances between *all pairs of input vectors* and not merely the distance between the closest pair of input vectors. One example of such an approach minimizes the "average" of the reciprocals of the distances between pairs of design points. To describe the details of this proposal, it convenient to again let $\mathcal{D}$ be an arbitrary n-point design consisting of *distinct* input sites $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ from a rectangular or non-rectangular input region $\mathcal{X}$. Define the average reciprocal distance (ARD) among inputs in $\mathcal{D}$ to be

$$m_{(p,\lambda)}(\mathcal{D}) = \left(\frac{1}{\binom{n_s}{2}} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}} \left[\frac{1}{\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right]^{\lambda} \right)^{1/\lambda}, \quad \lambda \geq 1 \,. \tag{5.4.5}$$

The combinatorial coefficient $\binom{n_s}{2}$ is the number of different pairs of points that can be drawn from a total of n_s distinct objects. For example, when $\lambda = 1$, the criterion function $m_{(p,1)}(\mathcal{D})$ is inversely proportional to the harmonic mean of the distances between design points.

For fixed (p, λ), an $n_s \times d$ design $\mathcal{D}_{av}$ is a *minimal ARD* (mARD) design if

$$m_{(p,\lambda)}(\mathcal{D}_{av}) = \min_{\mathcal{D}\subset\mathcal{X}} m_{(p,\lambda)}(\mathcal{D})\,. \tag{5.4.6}$$

The optimality condition (5.4.6) favors designs that possess nonredundancy in the location of input sites; specifically the criterion does not allow design points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that are (simultaneously) the same in *all* coordinates, i.e., with $\boldsymbol{x}_i = \boldsymbol{x}_j$. When $\lambda = 1$, the optimality condition (5.4.6) selects designs which maximize this inverse harmonic mean, of course, preventing any "clumping" of design points. The nonredundancy requirement can be seen even more clearly for large values of λ. Taking $\lambda \to \infty$, the criterion function (5.4.5) limit is

$$m_{(p,\infty)}(\mathcal{D}) = \max_{\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{D}} \frac{1}{\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)}\,. \tag{5.4.7}$$

Minimizing the right hand side of (5.4.7) is equivalent to maximizing (5.4.3). Thus, an n_s-point design$\mathcal{D}_{Mm}$ satisfying condition (5.4.6) for the limiting distances as $\lambda \to \infty$, namely

$$m_{(p,\infty)}(\mathcal{D}_{Mm}) = \min_{\mathcal{D}\subset\mathcal{X}} m_{(p,\infty)}(\mathcal{D})\,,$$

is a maximin distance design as defined previously because this criterion is equivalent to maximizing the minimum distance between all pairs of design points,

$$\max_{\mathcal{D}\subset\mathcal{X}} \min_{\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{D}} \rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j) \propto \frac{1}{m_{(p,\infty)}(\mathcal{D}_{Mm})}\,.$$

Before considering an example, we note several computational strategies that have been used to find optimal space-filling designs. Mm designs can be computed by solving the mathematical programming problem

$$\begin{aligned} &\max z \\ &\text{subject to} \\ &z \le \rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad 1 \le i < j \le n_s \\ &\mathbf{0}_d \le \boldsymbol{x}_\ell \le \mathbf{1}_d, \quad 1 \le \ell \le n_s \end{aligned} \tag{5.4.8}$$

in which an addition decision variable z has been added to the unknown $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_s}$; z is a lower bound for all distances in (5.4.8). While this problem can be solved by standard nonlinear programming algorithms for "small" n_s, the computational difficulty with this approach is the number of constraints on z grows on the order of n_s^2 (see Stinstra et al (2003)).

Example 5.8. Figure 5.8 displays Minimum ARD designs with Euclidean distance ($p = 2$) for $\lambda = 1$ and $\lambda = \infty$ when $n_s = 6$ and $d = 2$; by (5.4.7) the latter design is Mm design for this Euclidean distance case. Both designs concentrate points on or near the boundary of $\mathcal{X}$ so that the projections of the design points onto either axis produces multiple observations in 1-d. If the output depends primarily on one of the inputs, say x_1, this means that such a design will not fully explore x_1 space. We can remedy this feature of the design by restricting the class of available designs to only include, say, LHDs. This provides a computationally-convenient method of generating space-filling designs for computer experiments. Figure 5.2 is an example of a mARD within the class of LHDs for $p = 1$ and $\lambda = 1$. The use of multiple criteria to select designs is discussed further below. ♦

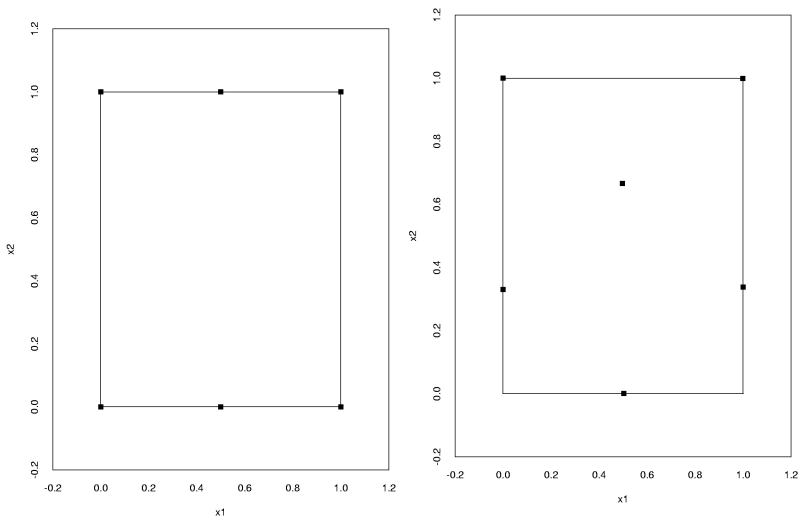


Fig. 5.8 Minimum ARD designs with respect to Euclidean distance ($p = 2$) for $\lambda = 1.0$ (left panel) and for $\lambda = \infty$ (right panel)

As noted above, neither the Mm nor the mARD optimal designs need have projections that are nonredundant. To reiterate, consider a computer experiment involving $d = 5$ input variables, only three of which (say) are active. In this event, a desirable property of an optimal $n_s \times 5$ design is nonredundancy of input sites projected onto the three-dimensional subspace of the active inputs. Such designs can be generated by computing the criterion values (5.4.5) for each relevant projection of the full design $\mathcal{D}$ and averaging these to form a new criterion function which is then minimized by choice of design $\mathcal{D}$. The approach is implemented by the Algorithms for the Construction of Experimental Designs (ACED) software of Welch (1985), among other packages. The Welch (1985) software was used to compute the optimal designs of this section.

Formally, the projection approach sketched in the previous paragraph can be described as follows. Let $J \subseteq \{1, 2, \ldots, d\}$ denote the index set of subspace dimensions in which nonredundancy of input sites is desired. For each $j \in J$, let $\{S_{kj}\}$ denote the k^{th} design in an enumeration of all j-dimensional projections of $\mathcal{D}$ for $k = 1, \ldots, \binom{n_s}{j}$, where $\binom{n_s}{j} = n_s!/(j!(n_s - j)!)$ is the number of subsets of size j that can be drawn from n_s distinct objects. Because the maximum distance apart that points can lie depends on j, it is essential that the distance $\rho_p(\cdot, \cdot)$ of points in a j-dimensional projection be normalized by this maximum distance of $j^{1/p}$ in order for distances to be comparable across different dimensional projections.

For $k = 1, \ldots, \binom{n_s}{2}$ and $j \in J$ define the minimum distance for the projected design $\mathcal{D}_{kj}$ to be

$$\min_{\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star \in \mathcal{D}_{kj}} \frac{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)}{j^{1/p}} \tag{5.4.9}$$

and the average reciprocal distance for $\mathcal{D}_{kj}$ to be the (modified) (5.4.5),

$$m_{J,(p,\lambda)}(\mathcal{D}_{kj}) = \left(\frac{1}{\binom{n_s}{2}} \sum_{\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star \in \mathcal{D}_{kj}} \left[\frac{j^{1/p}}{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)} \right]^\lambda \right)^{1/\lambda} . \tag{5.4.10}$$

Here, $\boldsymbol{x}_i^\star$ denotes the projection of $\boldsymbol{x}_i$ into the appropriate subspace determined by the values of j and k. Define the J-minimum of inputs in the design $\mathcal{D}$ to be

$$\rho_J(x, \mathcal{D}) = \min_{j \in J} \min_{k \in \{1, \ldots, \binom{n_s}{j}\}} \min_{\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star \in \mathcal{D}_{kj}} \frac{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)}{j^{1/p}} \tag{5.4.11}$$

and the J-average reciprocal projection design criterion function to be,

$$\begin{aligned} av_{J,(p,\lambda)}(\mathcal{D}) &= \left(\frac{1}{\binom{n_s}{2} \times \sum_{j \in J} \binom{n_s}{j}} \sum_{j \in J} \sum_{k=1}^{\binom{n_s}{j}} \sum_{\boldsymbol{x}_h^*, \boldsymbol{x}_\ell^* \in \mathcal{D}_{kj}} \left[\frac{j^{1/p}}{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)} \right]^\lambda \right)^{1/\lambda} \\ &= \left(\frac{1}{\sum_{j \in J} \binom{n_s}{j}} \sum_{j \in J} \sum_{k=1}^{\binom{n_s}{j}} [m_{J,(p,\lambda)}(\mathcal{D}_{kj})]^\lambda \right)^{1/\lambda} . \end{aligned} \tag{5.4.12}$$

An n_s-point design $\mathcal{D}_{MmP}$ is maximum with respect the projection criterion (5.4.11) provided

$$\rho_J(x, \mathcal{D}_{MmP}) = \max_{\mathcal{D}} \rho_J(x, \mathcal{D}) \tag{5.4.13}$$

and is $\mathcal{D}_{avp}$ is minimal ARD with respect to the projection criterion (5.4.12) if

$$\text{av}_{J,(p,\lambda)}(\mathcal{D}_{avp}) = \min_{\mathcal{D} \subset \mathcal{X}} \text{av}_{J,(p,\lambda)}(\mathcal{D}) . \tag{5.4.14}$$

Example 5.9. The optimal average projection designs (5.4.14) will also be space-filling if the class of designs searched is restricted to LHDs. As an example, let

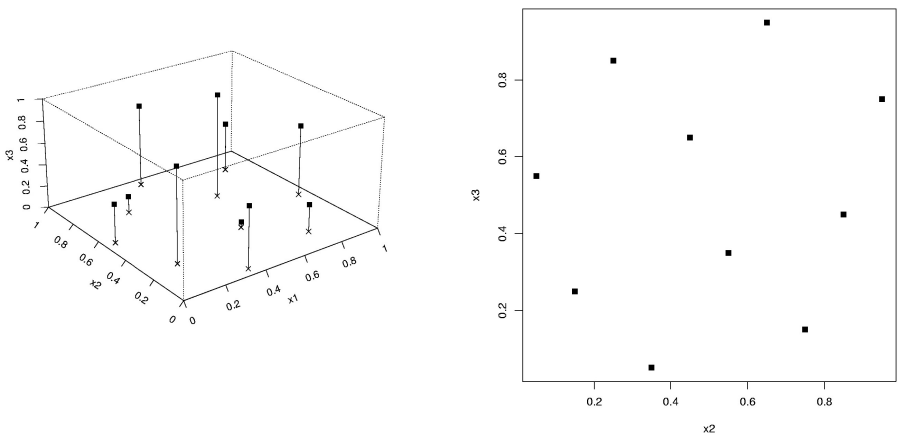


Fig. 5.9 Left panel: a 3-d plot of a $n_s = 10$ point optimal mARD design within the class of LHDs when $p = \lambda = 1$ and $J = \{2, 3\}$. Right panel: projection of left panel design onto x_1-x_2 plane.

$n_s = 10$ and $d = 3$. An optimal mARD design in the class of LHDs was generated with the specifications $p = \lambda = 1$ and $J = \{2, 3\}$. Figure 5.9 presents the design 3-d and the projection of the design onto the (x_2, x_3) subspace. Note that $1 \notin J$, as LHDs are nonredundant in each one-dimensional subspace by definition. ♦

Mm and mARD designs with specified projection dimensions J are alternatives to randomly selected LHDs and randomized orthogonal arrays for producing designs that are space-filling and are (reasonably) uniformly spread out when projected onto given lower dimensional subspaces. Unlike randomized orthogonal arrays that only exist for certain values of n, these designs can be generated for any sample size.

One consideration in using distance-based criteria is the choice of metric. Euclidean distance is a common choice. For the GP model, Euclidean distance is reasonable if the model is isotropic or if there is no prior information about the relative sizes of the correlation parameters. However, if there is prior information about the correlation parameters, Mahalanobis distance or some sort of weighted distance, with weights determined by the correlation parameters, may be more appropriate. See Williams et al (2011) where the use of Mahalanobis distance is considered in the context of a sequential design strategy.

5.5 Distance-based Designs for Non-rectangular Regions

Sections 5.2-5.4 describe several criteria for constructing space-filling designs when the input region is a hyper-rectangular. This section describes how *maximin distance* (Mm) and the *minimum ARD* (mARD) criteria from Section 5.4 have been applied to non-rectangular input regions. Note that the method of Tan (2013), mentioned in Section 5.4, for finding minimax designs over finite design spaces, includes situations in which the finite design space appears non rectangular.

As the following example illustrates, non-rectangular input regions occur naturally in many applications where the range of one or more inputs is related to that of other inputs. Hayeck (2009) studied the effects of four variables, one a biomechanical engineering design input (x_1) and three environmental inputs ($x_2 - x_4$), on the functioning of a total elbow prosthesis. The biomechanical input was the tip displacement (in *mm*), and the environmental inputs were the rotation of the implant axis about the lateral axis at the tip, the rotation of the implant axis about the anterior axis at the tip, and the rotation about the implant axis (all in degrees). The following constraints were imposed on the inputs based on anatomical considerations

$$\begin{array}{rcrcl} 0 & \leq & x_1 & \leq & 10 \\ -10 & \leq & 5x_2 + 2x_3 & \leq & 10 \\ -10 & \leq & -5x_2 + 2x_3 & \leq & 10 \\ -15 & \leq & x_4 & \leq & 15. \end{array} \tag{5.5.1}$$

These constraints state, among other things, that the maximum tip displacement is 10 *mm*; the rotation of the implant axis is 10° about lateral axis at the tip and 4° about anterior axis at the tip; and the rotation about the implant axis is $\pm 15^\circ$. The outputs of the computational simulation where various stresses and strains in the elbow.

Returning to the general case, the bulk of this section restricts attention to input regions that are bounded polytopes, i.e., have the form

$$\{\boldsymbol{x} \in \mathbb{R}^d \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\} \tag{5.5.2}$$

for given $\boldsymbol{A}$ and $\boldsymbol{b}$. The Hayeck (2009) input region (5.5.1) satisfies (5.5.2) for

$$\boldsymbol{A} = \begin{pmatrix} +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & +5 & +2 & 0 \\ 0 & -5 & +2 & 0 \\ 0 & +5 & -2 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

and

$$\boldsymbol{b} = (10, 0, 10, 10, 10, 10, 15, 15)^\top.$$

Recall that a design $\mathcal{D}_{Mm}$ is Mm provided it satisfies (5.4.4) while a design $\mathcal{D}_{av}$ is mARD provided it satisfies (5.4.6). The class of designs used in the maximization in (5.4.6) are those $\mathcal{D}$ having rows $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^\top$, $i = 1, \ldots n_s$, belonging to the desired input region. For example, when the input region is the bounded polytope (5.5.2), then

$$\mathcal{D} = \begin{pmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_{n_s}^\top \end{pmatrix}$$

where $\boldsymbol{A}\boldsymbol{x}_i \le \boldsymbol{b}$, for $i = 1, \ldots, n_s$.

As noted in Section 5.4, there are several practical and philosophical difficulties associated with the computation and use of maximin designs. First, because inputs with different scales can cause the computation of a maximin designed to be dominated by those inputs having larger ranges, the determination of a maximin design for a non-rectangular input region is performed for the problem in which all inputs have been scaled and shifted to the interval [0,1]. For example, for the bounded input region (5.5.2), the maximum of the j-th input x_j of $\boldsymbol{x} = (x_1, \ldots, x_d)^\top$, can be obtained by solving the linear program

$$\max \quad x_j \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \, .$$

Second, maximin designs need not have "space-filling" projections onto subsets of the input variables although using the J maximin criterion and selecting designs from the class of LHDs can eliminate this problem.

Now consider the construction of maximin designs for the case of inputs that satisfy (5.5.2). The mathematical program (5.4.8) for the Mm design can be modified to that of solving

$$\begin{aligned} &\max z \\ &\text{subject to} \\ &z \le \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad 1 \le i < j \le n_s \\ &\boldsymbol{A}\boldsymbol{x}_\ell \le \boldsymbol{b}, \qquad 1 \le \ell \le n_s \end{aligned} \tag{5.5.3}$$

in which the [0,1] bounds for each input are replaced by the bounded polytope constraints. Other constraints on the $\boldsymbol{x}_\ell$ can be handled similarly.

Trosset (1999)) described an approximate solution to the problem of finding a Mm design. He replaced $\rho_p(\cdot, \cdot)$ in (5.4.2) by a decreasing function of $\rho_p(\cdot, \cdot)$, e.g., $\phi(w) = 1/w$ which changes the minimum in (5.4.3) to

$$\max_{i<j} \phi(\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)) \tag{5.5.4}$$

and then replaces the maximization in (5.5.4) with that of minimizing

$$\left\{\sum_{i<j}\phi(\rho_2(\boldsymbol{x}_i,\boldsymbol{x}_j))^{\lambda}\right\}^{1/\lambda}. \tag{5.5.5}$$

For large λ, a design that minimizes (5.5.5) subject to $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$ is an approximate Mm design because (5.5.5) converges to (5.5.4) as $\lambda \to \infty$.

Stinstra et al (2003) introduced an algorithm that allows larger problems of the form (5.5.3) to be solved. Their algorithm solves a set of n_s subproblems to update a current feasible set of points $\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_{n_s}^c\}$ satisfying the constraints in (5.5.3) to an improved solution. The update step finds $\boldsymbol{x}_i^{c+1}$ from $\boldsymbol{x}_i^c$ when components with $\ell < i$ have been updated and those with $\ell > i$ have not been updated by solving

$$\begin{aligned} &\max w \\ &\text{subect to} \\ &w \le \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_\ell^{c+1}), \qquad \ell < i \\ &w \le \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_\ell^{c}), \qquad \ell > i \\ &\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \end{aligned} \tag{5.5.6}$$

for $(w^\star, \boldsymbol{x}_i^\star)$. Set $\boldsymbol{x}_i^{c+1} = \boldsymbol{x}_i^\star$. This cycle of n_s steps is repeated until a given minimum improvement in (5.4.2) occurs or a computational budget is exhausted.

As for rectangular input regions, Draguljić et al (2012) added criteria to that of maximinity which a design is required to satisfy. First, they consider "non-collapsingness" which can be thought of as an attempt to provide a space-filling design in each input and thus is similar in spirit to that of the LHD criterion. Beyond non-collapsingness, their designs can be selected to satisfy either a maximin or a maximum average distance criterion for the design.

5.6 Other Space-filling Designs

5.6.1 Designs Obtained from Quasi-Random Sequences

Quasi-random sequences are intended to produce finite sequences of points that fill the d-dimensional unit hypercube and have the property that a design with sample size n_s is obtained from the design of sample size $n_s - 1$ by adding a point to the design. Although introduced for numerically evaluating multi-dimensional integrals, they also allow one to generate space-filling designs.

Several such sequences have been proposed, including Halton sequences (Halton (1960)), Sobol´ sequences, (Sobol´ (1967) and Sobol´ (1976)), and Niederreiter sequences (Niederreiter (1988)). Section 5.7.4 presents some details for constructing the simplest of these sequences, the Halton sequence, as well as Sobol′ sequences.

Example 5.10. In Section **??**, Sobol´ sequences were used to select the values of the environmental variables and compared to other methods. The left-hand panel

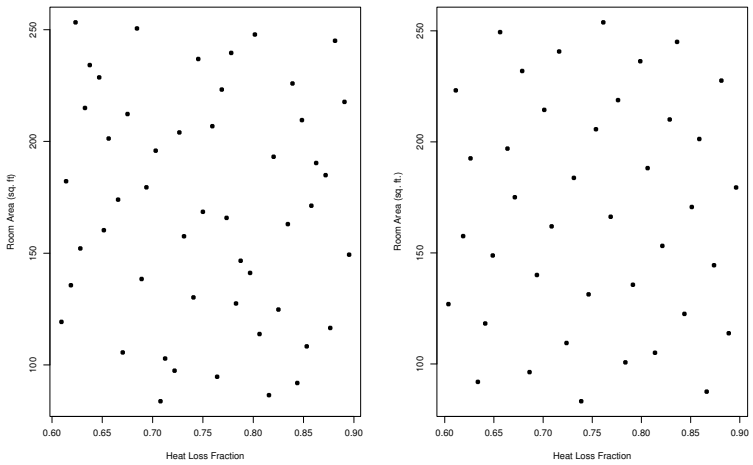


Fig. 5.10 Left Panel—projection of the 40 point, $d = 4$ variable Sobol´ sequence onto the room area × heat loss fraction plane; Right Panel—projection of the 40 point maximin LHD for the same four variables into the room area × heat loss fraction plane.

of Figure 5.10 displays one of the six, two-dimensional projections of the 40 point Sobol´ sequence in $d = 4$ variables that were used as inputs to generate the fire containment data displayed in Figure 1.4. Example **??** also uses this 40 point data set. The corresponding two-dimensional projection for the 40 point maximin LHD is shown in the right-hand panel of the same figure. It is clear from Figure 5.10 that the LHD is more evenly spread out than the design based the Sobol´ sequence. Thus, if it is important that the design be evenly spread out, the LHD appears to be preferable. On the other hand, the design based on the Sobol´ sequence appears to exhibit a greater variety of inter-point distances (distances between pairs of points in the design) than the LHD. If a greater variety of inter-point distances provides more information about the correlation parameters (and hence allows one to better estimate these parameters), then designs based on a Sobol´ sequence (or other types of sequences that have been used in numerical integration) may be preferable to the LHD. ♦

Suppose one uses a space-filling consisting of n_s points in the unit cube. After fitting a predictor to the data generated by the simulator, suppose one decides the fit is inadequate and m_s additional runs of the computer simulator are necessary. Is it possible to select the m_s runs in such a way that the resulting set of $n_s + m_s$ runs is space-filling?

In Section 5.2 we saw that for LHDs this is but only possible in special cases. However, because of the method of construction, this is possible for designs generated by Halton, Sobol´, and Niederreiter sequences. Thus, if the initial n_s design consists of the first n_s points in one of these sequences, simply add the next m_s points

in the sequence to generate the larger design. To the extent that Halton, Sobol´, and Niederreiter sequences are space-filling, both the initial and final designs will also be space-filling (although the degree to which the designs look space filling will depend on the particular values of n_s and m_s).

This ability to add points so that both the initial and final design are reasonably space-filling, makes quasi-random sequences such as Halton, Sobol´, and Niederreiter sequences appear attractive in the context of sequential experimentation. However, quasi-random sequences are usually space-filling only in the sense described in Section 5.1, namely as the number of points in the sequence increases, the sequence becomes increasingly dense in the design space (here assumed to be the d-dimensional unit cube). As Figure 5.10 suggests, quasi-random sequences need not look particularly space-filling for small to moderate sample sizes. Furthermore, if the number of runs is not a power of 2 (assuming the common case of base 2 for the construction of the sequence described in the Chapter Notes), then subsequent points do not necessarily fill in the most empty part of input space. Finally, such sequences can have bad projection properties. Liefvendahl and Stocki (2006) show in some analytic test problems that the statistical accuracy of predictors based on designs generated by the minimum ARD criterion is superior to that based on designs produced by a Sobol´ sequence. For these reasons, designs based on quasi-random sequences are much less popular in practice than other space-filling designs.

5.6.2 Uniform Designs

In Section 5.2 we considered criteria for selecting a space-filling design based on sampling methods and, in Sections 5.4 and 5.5, criteria based on distances between points. In this section, we consider a third intuitive design principle based on comparing the distribution of the points in a design to the uniform distribution.

As in Subsection 5.2.2, suppose that the vector of inputs is d-dimensional and denoted by $\boldsymbol{x} = (x_1, \ldots, x_d)$. Also again assume that $\boldsymbol{x}$ must fall in the d-dimensional hyper-cube $\mathcal{X} = [0, 1]^d$, possibly after recentering and rescaling of the inputs. Let $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ denote the set of n_s points at which we will observe the response $y(\boldsymbol{x})$. If we wish to emphasize that $\boldsymbol{x}$ is a random variable, we will use the notation $\boldsymbol{X}$. This would be the case, for example, if we are interested in $\mathrm{E}\{y(\boldsymbol{X})\}$. Below we take $\boldsymbol{X} \sim F(\cdot)$ where

$$F(\boldsymbol{x}) = \prod_{i=1}^{d} x_i \tag{5.6.1}$$

is the uniform distribution on $[0, 1]^d$ (other choices of distribution function are possible).

Fang et al (2000) and Fang et al (2005) discuss the notion of the *discrepancy* of a design $\mathcal{D}$, which measures the extent to which $\mathcal{D}$ differs from a completely uniform distribution of points. To be specific, let F_{n_s} be the empirical distribution function

of the points in $\mathcal{D}$, namely

$$F_{n_s}(\boldsymbol{x}) = \frac{1}{n_s} \sum_{i=1}^{n_s} I\{\boldsymbol{X}_i \leq \boldsymbol{x}\}, \tag{5.6.2}$$

where $I\{E\}$ is the indicator function of the event E and the inequality is with respect to the componentwise ordering of vectors in $\mathbb{R}^d$. The L_∞ discrepancy, sometimes called *star discrepancy* or simply discrepancy, is denoted $D_\infty(\mathcal{D})$ and is defined as

$$D_\infty(\mathcal{D}) = \sup_{\boldsymbol{x} \in \mathcal{X}} | F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x}) | . \tag{5.6.3}$$

This is perhaps the most popular measure of discrepancy and is the Kolmogorov-Smirnov statistic for testing fit to the uniform distribution.

Example 5.11. Suppose $d = 1$ and $\mathcal{X} = [0, 1]$ is the unit interval. It is not too difficult to show that the n_s point set

$$\mathcal{D} = \left\{ \frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s} \right\}$$

has discrepancy $D_\infty(\mathcal{D}) = 1/2n_s$ because $F(x) = x$ in this case. ♦

Another important measure of discrepancy is the L_p discrepancy of $\mathcal{D}$ which is denoted by $D_p(\mathcal{D})$ and defined by

$$D_p(\mathcal{D}) = \left[\int_{\mathcal{X}} \left| F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x}) \right|^p d\boldsymbol{x} \right]^{1/p}. \tag{5.6.4}$$

The L_∞ discrepancy of $\mathcal{D}$ is a limiting case of L_p discrepancy obtained by letting p go to infinity.

Niederreiter (1992) discusses the use of discrepancy for generating uniformly distributed sequences of points by quasi-Monte Carlo methods. Designs taking observations at sets of points with small discrepancies would be considered more uniform or more spread out than designs corresponding to sets with larger discrepancies. *Uniform designs* take observations at a set of points that minimizes D_p.

Other than the fact that it seems intuitively reasonable to use designs that are spread uniformly over $\mathcal{X} = [0, 1]^d$, why might one consider using a uniform design? One reason that has been proposed is the following. Suppose we are interested in estimating the mean of $g(y(\boldsymbol{X}))$,

$$\mu = \mathrm{E}\{g(y(\boldsymbol{X}))\} = \int_{\mathcal{X}} g(y(\boldsymbol{x}))d\boldsymbol{x},$$

where $g(\cdot)$ is some known function. We consider the properties of the naíve moment estimator

$$T = T(y(\boldsymbol{X}_1), \ldots, y(\boldsymbol{X}_{n_s})) = \frac{1}{n_s} \sum_{j=1}^{n_s} g(y(\boldsymbol{X}_j)).$$

The Koksma-Hlawka inequality (Niederreiter (1992)) gives an upper bound on the absolute error of this estimator, namely

$$| T(y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{n_s})) - \mu | \leq D_\infty(\mathcal{D})V(g),$$

where $V(g)$ is a measure of the variation of g that does not depend on $\mathcal{D}$ (see page 19 of Niederreiter (1992) for the definition of $V(g)$). For fixed $g(\cdot)$, this bound is a minimum when $\mathcal{D}$ has minimum discrepancy. This suggests that a uniform design may control the maximum absolute error of T as an estimator of μ. Also, because this holds for any $g(\cdot)$, it suggests that uniform designs may be robust to the choice of $g(\cdot)$ because they have this property regardless of the value of $g(\cdot)$.

However, just because an upper bound on the absolute error is minimized, it does not necessarily follow that a uniform design minimizes the maximum absolute error over $\mathcal{X}$ or has other desirable properties. Furthermore, in the context of computer experiments, we are usually not interested in estimating μ. Thus, the above is not a completely compelling reason to use a uniform design in computer experiments as discussed here.

Wiens (1991) provides another reason for considering uniform designs. Suppose one believes the response $y(\boldsymbol{x})$ follows the regression model

$$y(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i f_i(\boldsymbol{x}) + \varphi(\boldsymbol{x}) + \epsilon,$$

where the $\{f_i\}$ are known functions, the β_i unknown regression parameters, φ is an unknown function representing model bias, and ϵ normal random error. Wiens (1991) shows that under certain conditions on φ, the uniform design is best in the sense of maximizing the power of the overall F test of the regression.

Fang et al (2000) provide yet another reason why one may wish to use uniform designs. They note that in orthogonal designs, the points are typically uniformly spread out over the design space. Thus, there is the possibility that uniform designs may often be orthogonal. To explore this further, they use computer algorithms to find designs that minimize a variety of measures of discrepancy and in doing so generate a number of orthogonal designs. Efficient algorithms for generating designs that minimize certain measures of discrepancy, therefore, may be useful in searching for orthogonal designs.

Fang et al (2000) discuss a method for constructing (nearly) uniform designs. In general, finding a uniform design is not easy. One way to simplify the problem is to reduce the domain of $\mathcal{X}$, perhaps to a finite set of candidate points. Obviously, a uniform design over this reduced domain may not be uniform over $\mathcal{X}$, but suitable selection of a reduced domain may yield designs which are nearly uniform.

A related way to simplify the problem is to reduce the set of candidate designs to some large, finite set. For example, one could restrict attention to only LHDs and then select the one with the minimum discrepancy from the uniform distribution.

As previously, for purposes of what follows, assume $\mathcal{X} = [0, 1]^d$. Based on the uniform design for $d = 1$ given in Example 5.11, one might proceed as follows. Let

$\boldsymbol{\Pi} = (\Pi_{ij})$ be an $n_s \times d$ matrix such that each column of $\boldsymbol{\Pi}$ is a permutation of the integers $\{1, 2, \ldots, n_s\}$. Let $\boldsymbol{X}(\boldsymbol{\Pi}) = (x_{ij})$ be the $n_s \times d$ matrix defined by

$$x_{ij} = (\Pi_{ij} - 0.5)/n_s$$

for all i, j. The n_s rows of $\boldsymbol{X}$ define n_s points in $\mathcal{X} = [0,1]^d$. Hence, each matrix $\boldsymbol{\Pi}$ determines an n_s-point design. For example, when $d = 1$, if $\boldsymbol{\Pi} = (1, 2, \ldots, n_s)^\top$, then

$$\boldsymbol{X}(\boldsymbol{\Pi}) = \left(\frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s}\right)^\top,$$

which is the uniform design in $d = 1$ dimension. Note that the n_s rows of $\boldsymbol{X}(\boldsymbol{\Pi})$ correspond to the sample points of an LHD with points at the centers of each sampled cell. One might search over the set $\mathcal{P}$ of all possible permutations $\boldsymbol{\Pi}$, selecting the $\boldsymbol{\Pi}$ that produces the n_s-point design with minimum discrepancy. One would hope that this choice of design is nearly uniform over $\mathcal{X}$. Fang et al (2000) describe two algorithms for conducting such a search. Bratley et al (1994) is an additional source for an algorithm that can be used to generate low-discrepancy sequences of points and hence (near) uniform designs.

The discrepancies D_∞ for two designs that appear to be equally uniform may not be the same. The following example illustrates such a case.

Example 5.12. Suppose $d = 2$, $\mathcal{X} = [0,1]^2$, and consider the class of all designs generated by the set of permutations $\mathcal{P}$ introduced in the previous paragraph. One member of this class of designs is

$$\mathcal{D}_{diag} = \left\{\left(\frac{1}{2n_s}, \frac{1}{2n_s}\right), \left(\frac{3}{2n_s}, \frac{3}{2n_s}\right), \ldots, \left(\frac{2n_s - 1}{2n_s}, \frac{2n_s - 1}{2n_s}\right)\right\}.$$

This n_s-point design takes observations along the diagonal extending from the origin to the point $(1, 1)$. Intuitively, we would expect $\mathcal{D}_{diag}$ to be a poor design, because it takes observations only along the diagonal and does not spread observations over $[0,1]^2$. To compute the discrepancy of $\mathcal{D}_{diag}$, we first compute the empirical distribution function F_{n_s} for $\mathcal{D}_{diag}$ at an arbitrary point $\boldsymbol{x} = (x_1, x_2)$ in $[0,1]^2$. Notice that points in $\mathcal{D}_{diag}$ have both coordinates equal and it is not too hard to show from Equation (5.6.2) that

$$F_{n_s}(x_1, x_2) = \frac{\text{number of pts. in } \mathcal{D}_{diag} \text{ with first coordinate} \le \min\{x_1, x_2\}}{n_s}.$$

Notice that $F_{n_s}(\cdot, \cdot)$ is constant almost everywhere except for jumps of size $1/n_s$ at points for which one of the coordinates takes one of the values $\frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s-1}{2n_s}$. In particular, $F_{n_s}(x_1, x_2)$ has value $\frac{m}{n_s}$ $(1 \le m \le n_s)$ on the set

$$\mathcal{X}_m = \left\{(x_1, x_2) \in [0,1]^2 : \frac{2m-1}{2n_s} \le \min\{x_1, x_2\} < \frac{2m+1}{2n_s}\right\}.$$

Recall from (5.6.1) that $F(\cdot)$ is the uniform distribution,

$$F(\boldsymbol{x}) = x_1 x_2$$

on $\mathcal{X} = [0,1]^2$. On $\mathcal{X}_m$, the minimum value of $F(\boldsymbol{x})$ is $\left(\frac{2m-1}{2n_s}\right)^2$ and the supremum of $F(\boldsymbol{x})$ is $\frac{2m+1}{2n_s}$. This supremum is obtained in the limit as $\epsilon \to 0$ along the sequence of points $\left(\frac{2m+1}{2n_s} - \epsilon, 1\right)$. Thus, over $\mathcal{X}_m$, the supremum of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ is either $\left|\frac{m}{n_s} - \left(\frac{2m-1}{2n_s}\right)^2\right|$ or $\left|\frac{m}{n_s} - \frac{2m+1}{2n_s}\right| = \frac{1}{2n_s}$. For $1 \le m \le n_s$, it is not difficult to show that

$$\left|\frac{m}{n_s} - \left(\frac{2m-1}{2n_s}\right)^2\right| > \frac{1}{2n_s}.$$

Hence, over the set of all points $\boldsymbol{x}$ for which $F_{n_s}(\boldsymbol{x})$ has value $\frac{m}{n_s}$, the supremum of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ is

$$\frac{m}{n_s} - \left(\frac{2m-1}{2n_s}\right)^2 = \frac{n_s m - m^2 + m}{n_s^2} - \frac{1}{4n_s^2},$$

and this occurs at the point $(\frac{2m-1}{2n_s}, \frac{2m-1}{2n_s}) \in \mathcal{D}_{diag}$. Using calculus, one can show that the value of m that maximizes $\frac{n_s m - m^2 + m}{n_s^2} - \frac{1}{4n_s^2}$ is $\frac{n_s+1}{2}$ if n_s is odd, and $\frac{n_s}{2}$ if n_s is even. If n_s is odd, one gets

$$D_\infty(\mathcal{D}_{diag}) = \sup_{\{\boldsymbol{x} \in \mathcal{X}\}} \left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| = \frac{1}{4} + \frac{1}{2n_s}$$

and if n_s is even,

$$D_\infty(\mathcal{D}_{diag}) = \frac{1}{4} + \frac{1}{2n_s} - \frac{1}{4n_s^2}.$$

However, notice that when n_s is odd, *any* design corresponding to a permutation in $\mathcal{P}$ and taking $\frac{n_s+1}{2}$ of its observations at points which are less than or equal to $(1/2, 1/2)$ (under componentwise ordering of vectors) will have support on a set with a discrepancy that is greater than or equal to that of $\mathcal{D}_{diag}$. To see this, simply notice this discrepancy must be at least equal to the value of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ at $\boldsymbol{x} = (1/2, 1/2)$, which is equal to $D_\infty(\mathcal{D}_{diag})$. Likewise, if n_s is even, *any* design taking half of its observations at points less than or equal to $\left(\frac{n_s-1}{2n_s}, \frac{n_s-1}{2n_s}\right)$ will have support on a set with a discrepancy that is greater than or equal to that of $\mathcal{D}_{diag}$. Thus, $\mathcal{D}_{diag}$ is more uniform than any such design, even if such a design spreads points more evenly over $[0,1]^2$ than simply placing them along the diagonal.

Now consider the n_s-point design,

$$\mathcal{D}_{antidiag} = \left\{\left(\frac{1}{2n_s}, \frac{2n_s-1}{2n_s}\right), \left(\frac{3}{2n_s}, \frac{2n_s-3}{2n_s}\right), \ldots, \left(\frac{2n_s-1}{2n_s}, \frac{1}{2n_s}\right)\right\}.$$

This design takes observations along the antidiagonal that runs from the point $(0,1)$ to the point $(1,0)$. For this design, we notice that when n_s is odd, $F_{n_s}(\boldsymbol{x}) = 0$ at

$\boldsymbol{x} = \left(\frac{1}{2} - \epsilon, \frac{n_s+2}{2n_s} - \epsilon\right)$ and so, at this $\boldsymbol{x}$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| = \left(\frac{1}{2} - \epsilon\right)\left(\frac{n_s + 2}{2n_s} - \epsilon\right).$$

In the limit as $\epsilon \to 0$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| \to \frac{1}{4} + \frac{1}{2n_s}.$$

One can show that this is, in fact, the supremum value of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ for $\mathcal{D}_{antidiag}$, hence its discrepancy is $D_\infty(\mathcal{D}_{antidiag}) = \frac{1}{4} + \frac{1}{2n_s}$. Notice that $\frac{1}{4} + \frac{1}{2n_s}$ is also the value of $D_\infty(\mathcal{D}_{diag})$, so D_∞ considers $\mathcal{D}_{diag}$ and $\mathcal{D}_{antidiag}$ equally uniform when n_s is odd.

When n_s is even, by considering the point $\boldsymbol{x} = \left(\frac{n_s+1}{2n_s} - \epsilon, \frac{n_s+1}{2n_s} - \epsilon\right)$, one can show that in the limit as $\epsilon \to 0$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| \to \frac{1}{4} + \frac{1}{2n_s} + \frac{1}{4n_s^2}.$$

In this case, $D_\infty(\mathcal{D}_{antidiag})$ is at least as large as $\frac{1}{4} + \frac{1}{2n_s} + \frac{1}{4n_s^2}$. Notice that this quantity is larger than the discrepancy of $\mathcal{D}_{diag}$ when n_s is even, so in this case $\mathcal{D}_{diag}$ is a more uniform design than $\mathcal{D}_{antidiag}$. Most readers would consider both designs to be equally uniform. ♦

This example shows that discrepancy, at least as measured by D_∞, may not adequately reflect our intuitive notion of what it means for points to be evenly spread over $\mathcal{X}$. Other measures of discrepancy may perform better. In view of Wiens (1991), uniform designs may be promising, but additional study of their properties in the context of computer experiments is needed. In addition, it is not clear to what extent our intuitive notions of points being evenly spread over $\mathcal{X}$ correspond to objective measures of the performance of a design.

It should be noted that in Fang et al (2000), the design $\mathcal{D}_{diag}$ is eliminated from consideration because only matrices $\boldsymbol{\Pi}$ of rank d are considered, and the matrix $\boldsymbol{\Pi}$ corresponding to $\mathcal{D}_{diag}$ is of rank 1.

Fang et al (2005) includes an extensive discussion of uniform designs. Also, see the Chapter Notes list software for constructing uniform designs.

5.7 Chapter Notes

5.7.1 Proof That T_L is Unbiased and of the second part of Theorem 5.1

We use the same notation as in Section 5.2.3. To compute $\mathrm{E}\{T_L\}$, we need to describe how the LH sample is constructed. For each i, divide the range $[0, 1]$ of the i^{th} coor-

dinate of $\boldsymbol{X}$ into n_s intervals of equal marginal probability $\frac{1}{n_s}$ under F. Sample once from each of these intervals and let these sample values be denoted $X_{i1}, X_{i2}, \ldots, X_{in_s}$. Form the $d \times n_s$ array

$$\begin{pmatrix} X_{11} & X_{12} & \ldots & X_{1n_s} \\ X_{21} & X_{22} & \ldots & X_{2n_s} \\ & & \vdots & \\ X_{d1} & X_{d2} & \ldots & X_{dn_s} \end{pmatrix}$$

and then randomly permute the elements in each row using independent permutations. The n_s columns of the resulting array are the LH sample. This is essentially the procedure for selecting an LH sample that was discussed in Section 5.2.2. Another way to select an LH sample is as follows. The Cartesian product of the d subintervals $[0, 1]$ partitions $\mathcal{X}$ into n_s^d cells, each of probability $1/n_s^d$. Each of these n_s^d cells can be labeled by a set of d coordinates

$$\boldsymbol{m}_i = (m_{i1}, m_{i2}, \ldots, m_{id}),$$

where $1 \le i \le n_s^d$ and m_{ij} is a number between 1 and n corresponding to which of the n_s intervals of $[0, 1]$ is represented in cell i. For example, suppose $n_s = 3$, $d = 2$, and $F(\cdot)$ is uniform. We divide $[0, 1]$ into the three intervals $[0, \frac{1}{3})$, $[\frac{1}{3}, \frac{2}{3})$, and $[\frac{2}{3}, 1]$. Similarly for $[a_2, b_2]$. In this case the cell $[\frac{1}{3}, \frac{2}{3}) \times [\frac{1}{3}, \frac{2}{3})$ would have cell coordinates $(2, 2)$.

To obtain an LH sample, select a random sample of n_s of the n_s^d cells, say $\boldsymbol{m}_{i_1}, \boldsymbol{m}_{i_2}, \ldots, \boldsymbol{m}_{i_{n_s}}$, subject to the condition that for each j, the set $\{m_{i_\ell j}\}_{\ell=1}^{n_s}$ is a permutation of the integers $1, 2, \ldots, n_s$. We then randomly select a single point from each of these n_s cells. For an LH sample obtained in this manner, the density of $\boldsymbol{X}$, given $\boldsymbol{X} \in$ cell i, is

$$f(\boldsymbol{x} \mid \boldsymbol{X} \in \text{cell } i) = \begin{cases} \frac{1}{n_s^d} f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \text{cell } i \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the distribution of the output $y(\boldsymbol{X})$ under LH sampling is

$$\begin{aligned} P(y(\boldsymbol{X}) \le y) &= \sum_{i=1}^{n_s^d} P(y(\boldsymbol{X}) \le y \mid \boldsymbol{X} \in \text{cell } i) P(\boldsymbol{X} \in \text{cell } i) \\ &= \sum_{i=1}^{n_s^d} \int_{\text{cell } i \text{ and } y(\boldsymbol{x}) \le y} n_s^d f(\boldsymbol{x}) \left(\frac{1}{n_s^d}\right) d\boldsymbol{x} \\ &= \int_{y(\boldsymbol{x}) \le y} f(\boldsymbol{x}) d\boldsymbol{x}, \end{aligned}$$

which is the same as for random sampling. Hence we have $\mathrm{E}\{T_L\} = \mu$.

To compute $\mathrm{Var}\{T_L\}$, we view our sampling as follows. First we select the $\boldsymbol{X}_i$ independently and randomly according to the distribution of F from each of the n_s^d cells. We next independently select our sample of n_s cells as described above, letting

$$W_i = \begin{cases} 1 \text{ if cell } i \text{ is in our sample} \\ 0 \text{ otherwise} \end{cases}$$

and

$$G_i = g(y(\boldsymbol{X}_i)).$$

Then

$$\begin{aligned} \text{Var}\{T_L\} &= \text{Var}\left\{\frac{1}{n_s}\sum_{j=1}^{n_s} G_j\right\} \\ &= \frac{1}{n_s^2}\left[\sum_{i=1}^{n_s^d} \text{Var}\{W_i G_i\}\right. \\ &\quad \left. + \sum_{i=1}^{n_s^d}\sum_{j=1, j\neq i}^{n_s^d} \text{Cov}\left((W_i G_i), (W_j G_j)\right)\right]. \end{aligned}$$

To compute the variances and covariance on the right-hand side of this expression, we need to know some additional properties of the W_i. Using the fundamental rule that the probability of an event is the proportion of samples in which the event occurs, we find the following. First, $P(W_i = 1) = n_s/n_s^d = 1/n_s^{d-1}$ so W_i is Bernoulli with probability of success $1/n_s^{d-1}$. Second, if W_i and W_j correspond to cells having at least one common cell coordinate, then these two cells cannot both be selected, hence $\text{E}\{(W_i W_j)\} = 0$. Third, if W_i and W_j correspond to cells having no cell coordinates in common, then

$$\text{E}\{W_i\, W_j\} = P\{W_i = 1, W_j = 1\} = \frac{1}{n_s^{d-1}(n_s - 1)^{d-1}}.$$

This follows from the fact that, taking order into account, there are $n_s^d(n_s - 1)^d$ pairs of cells with no coordinates in common and in our sample of size n_s, there are $n_s(n_s - 1)$ such pairs.

Using the fact that for two random variables Z and V,
$\text{Var}\{Z\} = \text{E}\{\text{Var}\{Z \mid V\}\} + \text{Var}\{\text{E}\{Z \mid V\}\}$, we have

$$\begin{aligned} \text{Var}\{W_i G_i\} &= \text{E}\{\text{Var}\{W_i G_i \mid W_i\}\} + \text{Var}\{\text{E}\{W_i G_i \mid W_i\}\} \\ &= \text{E}\{W_i^2 \text{Var}\{G_i \mid W_i\}\} + \text{Var}\{W_i \text{E}\{G_i \mid W_i\}\} \\ &= \text{E}\{W_i^2 \text{Var}\{G_i\} + \text{Var}\{W_i \text{E}\{G_i\}\} \qquad (5.7.1) \\ &= \text{E}\{W_i^2\}\text{Var}\{G_i\} + \text{E}^2\{G_i\}\text{Var}\{W_i\}, \end{aligned}$$

where in (5.7.1) above we use the fact that $\boldsymbol{X}_i$ (and hence G_i) and W_i are independent. Letting

$$\mu_i = \text{E}\{g(y(\boldsymbol{X}_i))\} = \text{E}\{g(y(\boldsymbol{X})) \mid \boldsymbol{X} \in \text{cell } i\}$$

and recalling that W_i is Bernoulli, we have

$$\begin{aligned}
\sum_{i=1}^{n_s^d} \mathrm{Var}\{W_i G_i\} &= \sum_{i=1}^{n_s^d} \left[\mathrm{E}\{W_i^2\}\mathrm{Var}\{G_i\} + \mathrm{E}^2\{G_i\}\mathrm{Var}\{W_i\}\right] \\
&= \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \Big[E\{G_i - \mu_i\}^2 \\
&\quad + \frac{1}{n_s^{d-1}}(1 - \frac{1}{n_s^{d-1}})\mu_i^2\Big] \\
&= \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \Big[\int_{\text{cell } i} (g(y(\boldsymbol{x}))) - \mu + \mu - \mu_i)^2 \; n_s^d \, f(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad + \frac{1}{n_s^{d-1}}(1 - \frac{1}{n_s^{d-1}})\mu_i^2\Big] \\
&= n_s \, \mathrm{Var}\{y(\boldsymbol{X})\} - \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \left[(\mu - \mu_i)^2 + \frac{1}{n_s^{d-1}}(1 - \frac{1}{n_s^{d-1}})\mu_i^2\right].
\end{aligned}$$

Because W_ℓ and $G_\ell = g(y((\boldsymbol{X}_\ell)))$ are independent, then for $i \neq j$,

$$\begin{aligned}
\mathrm{Cov}\left((W_i G_i), (W_j G_j)\right) &= \mathrm{E}\{W_i G_i W_j G_j\} \\
&\quad - \mathrm{E}\{W_i G_i\} \mathrm{E}\left\{W_j G_j\right\} \\
&= \mathrm{E}\left\{W_i W_j\right\} \mathrm{E}\left\{G_i G_j\right\} \\
&\quad - \mathrm{E}\{W_i\} \, \mathrm{E}\{G_i\} \, \mathrm{E}\left\{W_j\right\} \, \mathrm{E}\left\{G_j\right\} \\
&= \mathrm{E}\left\{W_i W_j\right\} \, \mathrm{E}\{G_i\} \, \mathrm{E}\left\{G_j\right\} \\
&\quad - \frac{1}{n_s^{d-1}} \mathrm{E}\{G_i\} \; \frac{1}{n_s^{d-1}} \, \mathrm{E}\left\{G_j\right\} \\
&= \mathrm{E}\left\{W_i W_j\right\} \mu_i \mu_j - \frac{1}{n_s^{2d-2}} \, \mu_i \mu_j.
\end{aligned}$$

Hence

$$\sum_{i=1}^{n_s^d} \sum_{j=1, j\neq i}^{n_s^d} \mathrm{Cov}\left(W_i G_i, W_j G_j\right) = \sum_{i=1}^{n_s^d} \sum_{j=1, j\neq i}^{n_s^d} \left[\mathrm{E}\left\{W_i W_j\right\} \mu_i \mu_j - \frac{1}{n_s^{2d-2}} \mu_i \mu_j\right].$$

Recall that $\mathrm{E}\{W_i W_j\} = 0$ if cells i and j have at least one common cell coordinate. Let R denote the $n_s^d(n_s - 1)^d$ pairs of cells (with regards to order) having no cell coordinates in common. On this set we saw that

$$\mathrm{E}\left\{W_i W_j\right\} = \frac{1}{n_s^{d-1}(n_s - 1)^{d-1}}$$

so we have

$$\operatorname{Var}\left\{\frac{1}{n_s}\sum_{j=1}^{n_s} G_j\right\} = \frac{1}{n_s^2}\left[n_s \operatorname{Var}\left\{g(y(\boldsymbol{X})\right\} - \frac{1}{n_s^{d-1}}\sum_{i=1}^{n_s^d}(\mu-\mu_i)^2 \right.$$
$$+\frac{1}{n_s^{d-1}}\left(1-\frac{1}{n_s^{d-1}}\right)\sum_{i=1}^{n_s^d}\mu_i^2$$
$$\left.+\frac{1}{n_s^{d-1}(n_s-1)^{d-1}}\sum_{R}\mu_i\mu_j - \frac{1}{n_s^{2d-2}}\sum_{i=1}^{n_s^d}\sum_{j=1,j\neq i}^{n_s^d}\mu_i\mu_j\right].$$

Notice that

$$\begin{aligned}\sum_{i=1}^{n_s^d}\mu_i &= \sum_{i=1}^{n_s^d}\mathrm{E}\left\{g(y(\boldsymbol{X}))\mid \boldsymbol{X}\in \text{cell } i\right\}\\ &= \sum_{i=1}^{n_s^d}\int_{\text{cell } i} g(y(\boldsymbol{x}))n_s^d f(\boldsymbol{x})d\boldsymbol{x}\\ &= n_s^d\int_{\mathcal{X}} g(y(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x} = n_s^d\mu.\end{aligned}$$

So

$$\operatorname{Var}\left\{\frac{1}{n_s}\sum_{j=1}^{n_s} G_j\right\} = \frac{1}{n_s}\operatorname{Var}\left\{g(y(\boldsymbol{X}))\right\} - \frac{1}{n_s^{d+1}}\sum_{i=1}^{n_s^d}\left(\mu^2-2\mu_i\mu+\mu_i^2\right)$$
$$+\left(\frac{1}{n_s^{d+1}}-\frac{1}{n_s^{2d}}\right)\sum_{i=1}^{n_s^d}\mu_i^2$$
$$+\frac{1}{n_s^{d+1}(n_s-1)^{d-1}}\sum_{R}\mu_i\mu_j$$
$$-\frac{1}{n_s^{2d}}\sum_{i=1}^{n_s^d}\sum_{j=1,j\neq i}^{n_s^d}\mu_i\mu_j$$
$$= \operatorname{Var}\left\{T_R\right\} + \frac{1}{n_s}\mu^2 - \frac{1}{n_s^{2d}}\left(\sum_{i=1}^{n_s^d}\mu_i\right)^2$$
$$+\frac{1}{n_s^{d+1}(n_s-1)^{d-1}}\sum_{R}\mu_i\mu_j$$

$$
\begin{aligned}
&= \mathrm{Var}\{T_R\} - \frac{n_s - 1}{n_s}\mu^2 \\
&\quad + \left(\frac{n_s - 1}{n_s}\right)\left(\frac{1}{n_s^d(n_s - 1)^d}\right)\left(\sum_R \mu_i\mu_j\right) \\
&= \mathrm{Var}\{T_R\} \\
&\quad - \left(\frac{n_s - 1}{n_s}\right)\left(\frac{1}{n_s^d(n_s - 1)^d}\right)\left(\sum_R \mu^2\right) \\
&\quad + \left(\frac{n_s - 1}{n_s}\right)\left(\frac{1}{n_s^d(n_s - 1)^d}\right)\left(\sum_R \mu_i\mu_j\right) \\
&= \mathrm{Var}\{T_R\} \\
&\quad + \left(\frac{n_s - 1}{n_s}\right)\left(\frac{1}{n_s^d(n_s - 1)^d}\right) \\
&\quad \times \sum_R (\mu_i - \mu)(\mu_j - \mu) \qquad (5.7.2) \\
&\leq \mathrm{Var}\{T_R\},
\end{aligned}
$$

provided the last term in (5.7.2) is less than or equal to 0. Thus, whether LH sampling is superior to simple random sampling depends on the sign of this term, which in turn depends on the nature of g and f. Note also that LH sampling is superior to stratified random sampling with proportional sampling if

$$
\left(\frac{n_s - 1}{n_s}\right)\left(\frac{1}{n_s^d(n_s - 1)^d}\right)\sum_R (\mu_i - \mu)(\mu_j - \mu) \; < -\frac{1}{n_s}\sum_{i=1}^{I} p_i(\mu - \mu_i)^2
$$

McKay et al (1979) prove that under the assumptions of Theorem 5.1, if $(y(x_1, \ldots, x_d)$ is monotonic in each of its arguments and $g(w)$ is a monotonic function of w), then $\sum_R (\mu_i - \mu)(\mu_j - \mu) \leq 0$. This completes the proof of Theorem 5.1. ♦

5.7.2 The Use of LHDs in a Regression Setting

Owen (1992b) presents a multivariate extension of Theorem 5.3 and its application to computer experiments when fitting a regression to output data (rather the constant mean described in Section 5.2.3). The basis for the application is the following multivariate version of Theorem 5.3. The setting is as follows. Suppose that $\boldsymbol{X}$ has independent components with distribution function $F(\cdot)$, $\boldsymbol{y}(\boldsymbol{X}) = (y_1(\boldsymbol{X}), \ldots, y_k(\boldsymbol{X}))^\top$, $\overline{\boldsymbol{Y}} = \frac{1}{n_s}\sum_{i=1}^{n_s} \boldsymbol{y}(\boldsymbol{X}_i)$ and $\boldsymbol{\mu} = \int_{\mathcal{X}} \boldsymbol{y}(\boldsymbol{x})\, d\boldsymbol{x}$.

Corollary 5.1. Let $r_\ell(\boldsymbol{x})$ be the residual from additivity for $y_\ell(\boldsymbol{x})$ (see the discussion preceding Theorem 5.2 for the definition of the residual from additivity) and define

$$\sigma_{ij} = \int_{\mathcal{X}} r_i(\boldsymbol{x})\, r_j(\boldsymbol{x})\, dF(\boldsymbol{x}).$$

Let $\boldsymbol{\Sigma}$ be the $d \times d$ matrix whose (i, j) entry is σ_{ij}. Then $\sqrt{n_s}(\overline{\boldsymbol{Y}} - \boldsymbol{\mu})$ tends in distribution to $N_k(\boldsymbol{0}, \boldsymbol{\Sigma})$ as $n_s \to \infty$.

Let $\boldsymbol{Z}(\boldsymbol{x})$ be a vector valued function for which a linear model $\boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{\beta}$ is an appropriate approximation to $\boldsymbol{Y}(\boldsymbol{x})$. The "population" least squares value of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}_{\text{POP}} \equiv \left[\int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Z}^T(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Y}(\boldsymbol{x})\, dF(\boldsymbol{x}).$$

Assuming $\int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Z}^\top(\boldsymbol{x})\, dF(\boldsymbol{x})$ is known or easily computable (this would be the case for polynomial regression, for example), we can estimate $\boldsymbol{\beta}_{\text{POP}}$ by

$$\widehat{\boldsymbol{\beta}}_{\text{POP}} = \left[\int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Z}^\top(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \frac{1}{n_s} \sum_{i=1}^{n_s} \boldsymbol{Z}(\boldsymbol{X}_i)Y(\boldsymbol{X}_i).$$

The variance of $\widehat{\boldsymbol{\beta}}_{\text{POP}}$ is of the "sandwich" form

$$\left[\int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Z}^\top(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \boldsymbol{\Sigma} \left[\int_{\mathcal{X}} \boldsymbol{Z}(\boldsymbol{x})\boldsymbol{Z}^\top(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1},$$

where $\boldsymbol{\Sigma}$ is defined in Corollary 5.1 above using the j^{th} component of $\boldsymbol{Z}(\boldsymbol{x})$ times $\boldsymbol{Y}(\boldsymbol{x})$ in place of $Y_j(\boldsymbol{x})$ in the definition of $r_j(\boldsymbol{x})$. Appealing to Theorem 5.2, one might argue that to the extent that $\boldsymbol{Z}(\boldsymbol{x})\,\boldsymbol{Y}(\boldsymbol{x})$ is additive, the regression may be more accurately estimated from a LHD than from a design based on a simple random sample.

Owen (1992b) discusses some other estimators of $\boldsymbol{\beta}_{\text{POP}}$. The point is that when a linear model is likely to provide a good approximation to $y(\boldsymbol{x})$, using a LHD followed by regression modeling is not an unreasonable way to conduct computer experiments.

5.7.3 Other Space-Filling Designs

The methods discussed in this chapter are not the only ones that generate space-filling designs. The literature on numerical integration contains numerous suggestions for constructing evenly-spaced designs. Niederreiter (1992) contains a wealth of information about such designs, including their mathematical properties.

As mentioned in Section 5.2.1, one possibility is to choose points on a regularly spaced grid superimposed on the experimental region. For example, if the experimental region is $\mathcal{X} = [0, 1]^d$, the d-fold Cartesian product of the n_s point set

$$S = \left\{ \frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s} \right\}$$

would be a grid consisting of n_s^d points. Grid designs consist of an array of evenly spaced points, but projections onto subspaces have many replicated points.

An improvement over grids is obtained by the method of good lattice points. Such designs are appealing in that they appear evenly spaced and in some cases have attractive properties in numerical integration. Niederreiter (1992) discusses these designs in more detail. Bates et al (1996) consider lattice designs in the context of computer experiments.

Nets form another class of designs that appear space-filling and which are popular in numerical integration. See (Niederreiter (1992)) and (Owen (1995)) for more details.

Because these designs are intended for use in numerical integration, they are generally used in situations where a large sample size is employed. Their properties tend to be for large numbers of observation and their small-sample behavior is not clear (and thus their usefulness in computer experiments in which the total number of observations is constrained to be small).

5.7.4 A Primer on Constructing Quasi-Monte Sequences

To construct a Halton sequence $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ of n_s points on the d-dimensional unit hypercube, begin by choosing d prime numbers, or bases, $b_1, b_2, \ldots, b_d$. These could be, for example, the first d prime numbers. The base b_j will be used to construct the $j - th$ coordinates of the $\{\boldsymbol{x}_i\}$.

Next, select an integer m. Next, for suitably large t_{mj} (the highest power of b_j used in the representation of m in base b_j), represent the integer m in base b_j as

$$m = \sum_{k=0}^{t_{mj}} a_{jk}(m) b_j^k, \; j = 1, \ldots, d. \tag{5.7.3}$$

Next, form

$$x_{1j} = \sum_{k=0}^{t_{mj}} a_{j,k-t_{mj}-1}(m) b_j^{k-t_{mj}-1}, \; j = 1, \ldots, d. \tag{5.7.4}$$

Note that in forming x_{1j} we have simply reversed the digits in the representation of m in base b_j and placed these reversed digits after a decimal point.

Set $m = m + i - 1$ and repeat the above to form the x_{ij}.

Example 5.13. We compute the first five points in a 2-dimensional Halton sequence. We use bases $b_1 = 2$ and $b_2 = 3$. We begin with $m = 4$. In base 2, 4 is 100 and in base 3 11. Reversing the digits and adding a decimal point, $\boldsymbol{x}_1 = (.001_2, .11_3)$, where the subscript indicates the base. Converting to base 10, $.001_2 = 0\times2^{-1}+0\times2^{-2}+1\times2^{-3} =$

$1/8 = 0.125$ and $.11_3 = 1 \times 3^{-1} + 1 \times 3^{-2} = 1/3 + 1/9 = 0.444$. Thus, the first point in our Halton sequence is $\boldsymbol{x}_1 = (.125, .444)$.

Next, we increase m by 1 to 5. In base 2, 5 is 101 and in base 3 12. Reversing the digits and adding a decimal point, $\boldsymbol{x}_2 = (.101_2, .21_3)$. Converting to base 10, $.101_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 1/2 + 1/8 = 0.625$ and $.21_3 = 2 \times 3^{-1} + 1 \times 3^{-2} = 2/3 + 1/9 = 0.7784$. Thus, the second point in our Halton sequence is $\boldsymbol{x}_2 = (.625, .778)$.

The next 3 points correspond to $m = 6, 7, 8$. In base 2, these are 110, 111, and 1000. In base 3, these are 20, 21, and 22. Reversing digits and adding a decimal point, $\boldsymbol{x}_3 = (.011_2, .02_3)$, $\boldsymbol{x}_4 = (.111_2, .12_3)$, and $\boldsymbol{x}_5 = (.0001_2, .22_3)$. Converting to base 10, one finds $\boldsymbol{x}_3 = (.375, .222)$, $\boldsymbol{x}_4 = (.875, .556)$, and $\boldsymbol{x}_5 = (.0625, .8889)$. Figure 5.11 shows the resulting 5-point design.

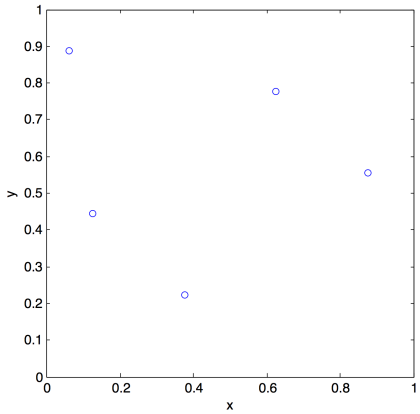


Fig. 5.11 5 point, $d = 2$ variable Halton sequence.

Halton sequences are relatively easy to calculate and have been found to be acceptably uniform for lower dimensions (d up to about 10). For higher dimensions the quality degrades rapidly because two-dimensional planes occur in cycles with decreasing periods.

Methods for creating sequences that behave better (appear uniform even in higher dimensions) have been developed by Sobol´ and Niederreiter.

To introduce the construction of the Sobol′ sequence consider working in one-dimension. To generate a sequence of values $x_1, x_2 ...$ with $0 < x_i < 1$, first we need to construct a set of *direction numbers* $v_1, v_2, ...$ Each v_i is a binary fraction that can be written $v_i = \frac{m_i}{2^i}$, where m_i is an odd integer such that $0 < m_i < 2^i$.

To obtain m_i the construction starts by choosing a primitive polynomial in the field $\mathbb{Z}_2$, i.e. one may choose $P = x^u + a_1 x^{u-1} + ... + a_{u-1} x + 1$ where each a_i is 0 or 1 and P is an arbitrary chosen primitive polynomial of degree u in $\mathbb{Z}_2$. Then, the m_i's can be calculated recurrently as

$$m_i = 2a_1 m_{i-1} \oplus 2^2 a_2 m_{i-2} \oplus ... \oplus 2^{u-1} a_{u-1} m_{i-u+1} \oplus 2^u m_{i-u} \oplus m_{i-u}$$

where each term is expressed in base 2 and $\oplus$ denotes a bit-by-bit exclusive-or operation, i.e

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

When using a primitive polynomial of degree d, the initial values $m_1, ..., m_u$ can be arbitrarily chosen provided that each m_i is odd and $m_i < 2^i$, $i = 1, ..., u$.

Example 5.14. If we choose the primitive polynomial $x^3 + x + 1$, so that $a_1 = 0, a_2 = 1$, and $a_3 = 1$, and choose the initial values $m_1 = 1$, $m_2 = 3$, $m_3 = 7$, the m_i's are calculated as follows:

$$m_i = 4m_{i-2} \oplus 8m_{i-3} \oplus m_{i-3}.$$

Then
$m_4 = 12 \oplus 8 \oplus 1 = 1100 \oplus 1000 \oplus 0001 = 0101 = 0 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 5$
$m_5 = 28 \oplus 24 \oplus 3 = 11100 \oplus 11000 \oplus 00011 = 00111 = 7$
$m_6 = 20 \oplus 56 \oplus 7 = 010100 \oplus 111000 \oplus 000111 = 43$
and
$v_1 = \frac{m_1}{2^1} = \frac{1}{2^1} = 0.1$ in binary
$v_2 = \frac{m_2}{2^2} = \frac{3}{2^2} = 0.11$ in binary
$v_3 = \frac{m_3}{2^3} = \frac{7}{2^3} = 0.111$ in binary
$v_4 = \frac{m_4}{2^4} = \frac{5}{2^4} = 0.0101$ in binary, and so on.

In order to generate the sequence $x_1, x_2, ...$, Sobol′ proposed using

$$x_i = b_1 v_1 \oplus b_2 v_2 \oplus \cdots$$

and

$$x_{i+1} = x_i \oplus v_c$$

where $\cdots b_3 b_2 b_1$ is the binary representation of i and b_c is the rightmost zero-bit in the binary representation of i.

The first few values of x are thus generated as follows. To start the recurrence, take $x_0 = 0$.

$$
\begin{aligned}
Initialization: \quad x_0 &= 0 \\
i &= 0 \text{ in binary so} \\
c &= 1 \\
Step\ 1: \quad x_1 &= x_0 \oplus v_1 \\
&= 0.0 \oplus 0.1 \text{ in binary} \\
&= 0.1 \text{ in binary} \\
&= \frac{1}{2} \\
i &= 01 \text{ in binary so} \\
c &= 2 \\
Step\ 2: \quad x_2 &= x_1 \oplus v_2 \\
&= 0.10 \oplus 0.11 \text{ in binary} \\
&= 0.01 \text{ in binary} \\
&= \frac{1}{4} \\
i &= 10 \text{ in binary so} \\
c &= 1
\end{aligned}
$$

$$
\begin{aligned}
Step\ 3: \quad x_3 &= x_2 \oplus v_1 \\
&= 0.01 \oplus 0.10 \text{ in binary} \\
&= 0.11 \text{ in binary} \\
&= \frac{3}{4} \\
i &= 011 \text{ in binary so} \\
c &= 3
\end{aligned}
$$

and so on.

To generalize this procedure to s dimensions, Sobol´ (1976) shows that in order to obtain $O(\log^s n)$ discrepancy, where n represents the number of points, it suffices to choose s distinct primitive polynomials, calculate s sets of *direction numbers* and then generate each component x_{ij} of the quasi-random vector separately.

The uniformity of a Sobol´ sequence can be very sensitive to the starting values, especially in higher dimensions. Various criteria exist for starting values, $m_1, m_2, \ldots$ to improve uniformity.

Niederreiter (1988) proposed a new method of generating quasi-Monte Carlo sequences intended to improve on Sobol´ sequences. Let $\triangle(N)$ denote $n \times D^*_{n_s}$, where $D^*_{n_s}$ is the star discrepancy. It is believed that the best possible bound for the discrepancy of the first n_s terms of a sequence of points in $[0, 1)^d$ is of the form

$$\triangle(n_s) \leq C_d(\log n_s)^d + O((\log n_s)^{d-1})$$

for all $n_s \geq 2$. The methods proposed by Niederreiter (1988) yield sequences with the lowest C_d currently known. We will not discuss the construction of Niederreiter sequences, but details can be found in, for example, Lemieux (2009).

5.7.5 Software for Constructing Space-filling Designs

The notes below are a partial list of software that can construct space-filling designs for computer experiments rather than a definitive list of possibilities.

5.7.5.1 LHD and Distance-based Designs

Both R and MATLAB will generate randomly selected Latin Hypercube Designs. The DiceDesign package in R will generate random LHD designs, (nearly) maximin LHD designs, and LHD designs with low discrepancy. It will also generate maximin designs. DiceDesign was created by Jessica Franco, Delphine Dupuy, Olivier Roustant, Guillaume Damblin and Bertrand Iooss.

Given (n_s, d), the program ACED can generate maximin designs within the class of LHD designs.

JMP will generate LHD and maximin designs for rectangular, user-specified design regions and for any (n_s, d). For LHD designs, JMP chooses points so that the design is (nearly) maximin subject to a constraint that maintains even spacing between factor levels. JMP refers to maximin designs as sphere-packing designs.

JMP will generate two types of designs with good distance-based properties, and for non rectangular regions. One type is called "fast flexible filling designs", and can be generated for design regions determined by user-specified linear constraints on the inputs, and for any (n_s, d). The algorithm produces designs with good properties under the minimax distance criterion. The other type is called "minimum potential designs" and can be generated for spherical regions for any (n_s, d).

To generate these designs, one must run the Space Filling Design command under the DOE menu. See the Help menu in JMP for details.

DAKOTA is a software package developed at Sandia National Laboratories for the analysis of data from predictive simulations. This package will generate several types of space-filling designs including orthogonal array designs, LHDs, and orthogonal array-based LHDs. DAKOTA can be downloaded from `https://dakota.sandia.gov`

The C code of Williams will generate optimal designs for the following scenarios

1. Given the set of dimensions $\mathcal{J} \subset \{1, \ldots, d\}$ over which projections are of interest, the C code OALHD will construct a design that maximize the minimum (normalized) interpoint distance (5.4.11) in the class of OA-based LHDs based on a given starting OA. It can also construct a design that maximizes the average reciprocal distance (5.4.12) in for same class of OA-based LHDs. A comprehensive library of OAs is available at the website `http://neilsloane.com/oadir/` maintained by Neal Sloane.

2. Given the set of dimensions $\mathcal{J} \subset \{1, \ldots, d\}$ over which projections are of interest, the C code SLHD will construct a design that maximize the minimum (normalized) interpoint distance (5.4.11) in the class of symmetric LHDs (see Section 5.3.3) of a given number of runs. It can also construct a design that maximizes the average reciprocal distance (5.4.12) for the same class of symmetric LHDs

The R package SLHD by Ba, Brenneman, and Myers will generate optimal sliced LHDs. See **?** for a discussion of optimal sliced LHDs and the construction of such designs.

The R package MaxPro by Ba and Joseph generates maximum projection designs. See Joseph et al (2015) for a discussion of maximum projection designs and the construction of such designs.

5.7.5.2 concad Software

The R package concad is an that can be obtained from D. Draguljic (ddraguljic@gmail.com) that computes optimum distance designs of a given number of runs for bounded polygonal input regions, i.e., regions of points $\boldsymbol{x}$ that satisfy $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ for given $\boldsymbol{A}$ and $\boldsymbol{b}$. It can also be used to augment a given design using the maximin or minimum ARD criteria.

5.7.5.3 Quasi-Monte Carlo Sequences

Code exists for generating many different Quasi-Monte Carlo sequences. A web search will identify several options. Specifically, the R function runif.sobol will generate Sobol´ sequences and the function runif.halton will generate Halton sequences. (See the R documentation for the details of the bases used in the construction and the required R packages).

In MATLAB, Halton sequences in d dimensions can be generated using the haltonset function while Sobol´ sequences can be generated using the sobolset function. Both functions require that the Statistics toolbox is available. One can also find online MATLAB code for generating Niederreiter sequences. For example, see http://people.sc.fsu.edu/~burkardt/m_src/niederreiter2/niederreiter2.html.

The open-source GSL library contains code for generating Sobol´ and Niederreiter sequences which can be accessed using a calling routine in C, say.

5.7.5.4 Uniform Designs

Constructing uniform designs is nontrivial. The Mathematics Department of Hong Kong Baptist University maintains a web site with information about uniform designs, including lists of publications about uniform designs and tables of uniform designs. The web site is located at the URL www.math.hkbu.edu.hk/UniformDesign/.

JMP will generate uniform designs. JMP uses the centered L_2 discrepancy measure of Hickernell (1998). To generate uniform designs, one must run the Space Filling Design command under the DOE menu. See the Help menu in JMP for details.

The DiceDesign package in R will generate uniform designs under a variety of discrepancies including the L_2 discrepancy, the centered L_2 discrepancy, and the star discrepancy.

5.7.5.5 Online Catalogs of Designs

In addition there are websites that provide catalogs of designs tat are optimal under distance or other criteria. Of these, the site `www.spacefillingdesigns.nl` is particularly rich.

References

Atkinson AC, Donev AN (1992) Optimum experimental designs. Oxford University Press

Ba S, Joseph VR (2011) Multi-layer designs for computer experiments. JASA 106:1139–1149

Bates RA, Buck RJ, Riccomagno E, Wynn HP (1996) Experimental design and observation for large systems. Journal of the Royal Statistical Society B 58:77–94

Bernardo MC, Buck RJ, Liu L, Nazaret WA, Sacks J, Welch WJ (1992) Integrated circuit design optimization using a sequential strategy. IEEE Transactions on Computer-Aided Design 11:361–372

Box G, Hunter W, Hunter J (1978) Statistics for Experimenters. J. Wiley, New York

Box GE, Draper NR (1987) Empirical model-building and response surfaces. John Wiley & Sons, New York

Bratley P, Fox BL, Niederreiter H (1994) Algorithm 738: Programs to generate niederreiter's low-discrepancy sequences. ACM Transactions on Mathematical Software 20:494–495

Butler NA (2001) Optimal and orthogonal latin hypercube designs for computer experiments. Biometrika 88:847–857

Chapman WL, Welch WJ, Bowman KP, Sacks J, Walsh JE (1994) Arctic sea ice variability: Model sensitivities and a multidecadal simulation. Journal of Geophysical Research C 99(1):919–936

Chen RB, Wang W, Wu CFJ (2011) Building surrogates with overcomplete bases in computer experiments with applications to bistable laser diodes. IEE Transactions 182:978–988

Craig PC, Goldstein M, Rougier JC, Seheult AH (2001) Bayesian forecasting for complex systems using computer simulators. Journal of the American Statistical Association 96:717–729

Dean AM, Voss D (1999) Design and Analysis of Experiments. Spring-Verlag, New York, New York

Draguljić D, Santner TJ, Dean AM (2012) Non-collapsing spacing-filling designs for bounded polygonal regions. Technometrics 54:169–178

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

Fang KT, Li R, Sudjianto A (2005) Design and Modeling for Computer Experiments. Chapman and Hall

Halton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer Math 2:84–90

Handcock MS (1991) On cascading latin hypercube designs and additive models for experiments. Communications Statistics—Theory Methods 20:417–439

Hayeck GT (2009) The kinematics of the upper extremity and subsequent effects on joint loading and surgical treatment. PhD thesis, Cornell University, Ithaca, NY USA

Hedayat A, Sloane N, Stufken J (1999) Orthogonal Arrays. Springer-Verlag, New York, New York

Hickernell FJ (1998) A generalized discrepancy and quadrature error bound. Math Comp 67:299–322

John JA (1987) Cyclic Designs. Chapman & Hall Ltd, New York

John PWM (1980) Incomplete Block Designs. M. Dekker, Inc., New York

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

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black–box functions. Journal of Global Optimization 13:455–492

Joseph VR, Gul E, Ba S (2015) Maximum projection designs for computer experiments. Biometrika 102:1–1

Kennedy MC, O'Hagan A (2000) Predicting the output from a complex computer code when fast approximations are available. Biometrika 87:1–13

Kennedy MC, O'Hagan A (2001) Bayesian calibration of computer models (with discussion). Journal of the Royal Statistical Society B 63:425–464

Lemieux C (2009) Monte Carlo and Quasi-Monte Carlo sampling,. Springer, New York, NY, USA

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

Loeppky JL, Sacks J, Welch WJ (2009) Choosing the sample size of a computer experiment: A practical guide. Technometrics 51(4):366–376

Loeppky JL, Moore LM, Williams BJ (2012) Projection array based designs for computer experiments. Journal of Statistical Planning and Inference 142:1493–1505

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

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

Niederreiter H (1988) Low-discrepancy and low-dispersion sequences. Journal of Number Theory 30:51–70

Niederreiter H (1992) Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia

Owen AB (1992a) A central limit theorem for Latin hypercube sampling. Journal of the Royal Statistical Society, Series B: Methodologic al 54:541–551

Owen AB (1992b) Orthogonal arrays for computer experiments, integration and visualization (Corr: 93V3 p261). Statistica Sinica 2:439–452

Owen AB (1995) Randomly permuted (t, m, s)-nets and (t, s) sequences. In: Niederreiter H, Shiue PJS (eds) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer-Verlag, New York, pp 299–317

Park JS (1994) Optimal latin-hypercube designs for computer experiments. Journal of Statistical Planning and Inference 39:95–111

Pukelsheim F (1993) Optimal Design of Experiments. J. Wiley, New York

Qian PZ, Seepersad CC, Joseph VR, Allen JK, Wu CFJ (2006) Building surrogate models with details and approximate simulations. ASME Journal of Mechanical Design 128:668–677

Qian PZG (2009) Nested Latin hypercube designs. Biometrika 96:957–970

Qian PZG (2012) Sliced latin hypercube designs. Journal of the American Statistical Association 107:393–399

Qian PZG, Wu CFJ (2008) Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. Technometrics 50(2):192–204

Raghavarao D (1971) Constructions and Combinatorial Problems in Design of Experiments. J. Wiley, New York

Silvey SD (1980) Optimal design: An introduction to the theory for parameter. Chapman & Hall Ltd, New York

Sobol´ IM (1967) Distribution of points in a cube and approximate evaluation of integrals. USSR Comput Maths Math Phys 7:86–112

Sobol´ IM (1976) Uniformly distributed sequences with an additional uniform property. USSR Comput Maths Math Phys 16:236–242

Stein ML (1987) Large sample properties of simulations using latin hypercube sampling. Technometrics 29:143–151

Stinstra E, den Hertog D, Stehouwer P, Vestjens A (2003) Constrained maximin designs for computer experiments. Technometrics 45(4):340–346

Street AP, Street DJ (1987) Combinatorics of experimental design. Oxford University Press, Oxford

Tan MHY (2013) Minimax designs for finite design regions. Technometrics 55:346–358

Tang B (1993) Orthogonal array-based latin hypercubes. Journal of the American Statistical Association 88:1392–1397

Tang B (1994) A theorem for selecting OA-based latin hypercubes using a distance criterion. Communications in Statistics - Theory and Methods 23:2047–2058

Trosset MW (1999) Approximate maximin distance designs. In: ASA Proceedings of the Section on Physical and Engineering Sciences, American Statistical Association (Alexandria, VA), pp 223–227

Vazquez E, Bect E (2011) Sequential search based on kriging: Convergence analysis of some algorithms. Proceedings 58th ISI World Statistics Congress

Welch WJ (1985) Aced: Algorithms for the construction of experimental designs. The American Statistician 39:146

Welch WJ, Buck RJ, Sacks J, Wynn HP, Mitchell TJ, Morris MD (1992) Screening, predicting, and computer experiments. Technometrics 34:15–25

Wiens DP (1991) Designs for approximately linear regression: Two optimality properties of uniform designs. Statistics and Probability Letters 12:217–221

Williams BJ, Loeppky JL, Moore LM, Macklem MS (2011) Batch sequential design to achieve predictive maturity with calibrated computer models. Reliability Engineering and System Safety 96(9):1208 – 1219

Wu CFJ, Hamada M (2000) Experiments: Planning, Analysis, and Parameter Design Optimization. J. Wiley, New York

Wu CFJ, Hamada M (2009) Experiments: Planning, Analysis, and Parameter Design Optimization, Second Edition. J. Wiley, New York

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

Ye KQ, Li W, Sudjianto A (2000) Algorithmic construction of optimal symmetric Latin hypercube designs. Journal of Statistical Planning and Inference 90(1):145–159