Non-collapsing Space-filling Designs for Bounded Non-rectangular Regions

Danel Draguljić and Thomas J. Santner and Angela M. Dean
Department of Statistics
The Ohio State University
Columbus, OH 43210

Abstract

Many researchers use computer simulators as experimental tools when physical experiments are infeasible. When computer codes are computationally intensive, nonparametric predictors can be fitted to training data for detailed exploration of the input-output relationship. The accuracy of such flexible predictors is enhanced by taking training inputs to be "space-filling". If there are inputs that have little or no effect on the response, it is essential that the design be "non-collapsing" in the sense of having space-filling lower dimensional projections. This paper describes an algorithm for constructing space-filling designs for input regions that are bounded convex sets of possibly high dimension. On-line supplementary materials showing the performance of the algorithm accompany the paper.
KEY WORDS: Average reciprocal distance; Column-wise algorithm; Computer experiment; Convex bounded region; High-dimensional input space; Maximin design

1 INTRODUCTION

The purpose of this paper is to present an algorithm, named CoNcaD, (**Co**nstrained **N**on-**c**oll**a**psing **D**esign) for constructing non-collapsing, space-filling designs for bounded convex non-rectangular input regions, of possibly high dimension, which are to be used as the initial set of runs of a "computer experiment." Computer experiments study input/output

relationships using computer code implementations of physics- or biology-based mathematical models. In this paper, such simulators are regarded as deterministic, at least up to numerical noise. Deterministic computer simulators should be contrasted with more traditional "stochastic computer simulations" which are computer code implementations of macro-model descriptions of input/output variable relationships. Multiple runs of stochastic computer simulators are required to understand the *mean* (system) output corresponding to a given set of inputs (see, for example, Nelson (1995)).

In many scientific, engineering, and other applications, non-rectangular input regions occur frequently. Two examples of experiments based on deterministic computational simulators will be used to illustrate the proposed algorithm. Both example have input regions of the form $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ where $\boldsymbol{x}$ is a $p \times 1$ vector of inputs, $\boldsymbol{A}$ is a $r \times p$ matrix, and $\boldsymbol{b}$ is a $r \times 1$ vector.

Example 1.1 Design of a Total Elbow Replacement

Computer simulators are frequently used by biomedical engineers to aid in the engineering design and/or analysis of mechanical performance of prosthetic devices. For example, in his PhD thesis, Hayeck (2009) studied the effects of four variables on the functioning of a total elbow replacement prosthetic device. These variables were the tip displacement (x_1), the rotation of the implant axis about the lateral axis at the tip (x_2), the rotation of the implant axis about the anterior axis at the tip (x_3), and the rotation about the implant axis (x_4). Here x_1 is measured in mm while x_2, x_3, and x_4 are measured in degrees ($^\circ$). The following constraints were imposed on the inputs based on anatomical considerations:

$$\begin{gathered} 0 \leq x_1 \leq 10 \\ -10 \leq 5x_2 + 2x_3 \leq 10 \\ -10 \leq -5x_2 + 2x_3 \leq 10 \\ -15 \leq x_4 \leq 15\,. \end{gathered}$$

For example, these constraints state that the maximum tip displacement is 10 mm and the rotation about the implant axis is $\pm 15^\circ$. The CoNcaD algorithm proposed in Section 3 was used to provide a 10-point space-filling and non-collapsing design for this 4-dimensional

region (Hayeck (2009)). The properties of the design are given in Section 6.

Example 1.2 Design of a Tool Coating

To extend the life of tools that are used in tribological configurations, engineers apply thin, hard coatings to the tool surface. Multilayer systems are one form of protective coating that have been considered for tool surfaces (Subramanian and Strafford (1993)). Our mechanical engineering collaborators at The Ohio State University were interested in studying the performance of coatings applied to the tool surface in pairs under a constant stress: one coating was made of titanium nitride (TiN) and the other was pure titanium (Ti) (Nekkanty; 2009). To prevent destructive damage to n potentially very expensive coatings in a physical experiment, the mechanical engineers implemented a finite element computer model (FEM) of this physical process that calculated a set of stresses related to the failure mode of the coating; the inputs to the FEM were the number of Ti/TiN pairs and the thicknesses of each layer. Let x_i, $i = 1, ..., T$ denote the thickness of the i^{th} layer (in microns, μm) where $T \in \{2, 4, 6, 8\}$ represents the total number of layers. Then the constraint

$$2 \leq \sum_{i=1}^{T} x_i \leq 6 \quad \text{for} \quad T \in \{2, 4, 6, 8\}$$
$$x_i \geq 0.25 \quad \text{for} \quad i = 1, \ldots, T$$

was imposed on the layer system. Some properties of the design produced by the CoNcaD algorithm for this example are given in Section 6.

Frequently, deterministic computer simulators that are used as experimental tools are computationally intensive and may require many hours or even days to solve the underlying mathematical model. In such cases, a rapidly-computable statistical predictor of the computer simulator output, an "emulator" of the code output, is usually developed to understand more fully the input/output process under investigation. Using a limited number of (training) runs from the simulator, regression-based mean models are typically *not* used as predictors; hence D- or other classic optimality criteria cannot be used to select the training data input run sites. Instead, highly flexible nonparametric predictors are ordinarily con-

structed to predict simulator outputs at new input sites (Santner et al. (2003)). Because of the extreme flexibility of non-parametric predictors, their performance depends heavily on the ability of the experimental design to provide output from a broad portion of the input space. This is true, firstly, because process-based predictors are more likely to be accurate at points where the predictor is interpolating the training data. Secondly, the process-based uncertainty quantification of a predicted values at a given input site depends on the "distances" between that input site and the input sites of the training data; space-filling designs can minimize the distance between an arbitrarily selected point in the input space and the nearest point in the training data (see Simpson et al. (2001) or Bursztyn and Steinberg (2006) for additional justifications for choosing space-filling designs).

To achieve space-fillingness, Johnson, Moore and Ylvisaker (1990) advocated maximin and minimax criteria for computer experiment designs with the theoretical justification being that, in an asymptotic sense as the correlations become weak, the maximin designs are D-optimal.

When integration of the output is of primary interest, uniform designs and Hamersley sequence designs are alternative methods for devising space-filling designs. In addition, a p-dimensional uniform design imposes good one-dimensional projections as well as p-dimensional uniformity (Fang et al. (2000)).

None of the above approaches deal with one additional important consideration in determining desirable experimental designs: projections of the inputs onto subspaces of the design should also be space-filling. This "criterion" is motivated by the following two facts. First, computer codes are deterministic and thus it is wasteful to replicate runs at the same input site because the same output is returned. Second, in cases where only a subset of the input variables are "active," input design points which coincide in any subset of coordinates give essentially duplicate runs. We call such designs "collapsing". A number of authors have added the space-fillingness of projections to their list of design considerations. As examples, McKay, Beckman and Conover (1979) introduced Latin hypercube designs for

applications with rectangular input regions that have good one-dimensional projection properties. Morris and Mitchell (1995) advocated selecting from among Latin hypercube designs using a maximin criterion. Recently, van Dam (2008) constructed two-dimensional minimax Latin hypercube designs. Tang (1993) used strength r orthogonal arrays to construct Latin hypercube designs which improve the one-dimensional projection properties of Latin hypercube designs to r-dimensional projections. Audze and Eglais (1977) introduced an intuitive, "energy" based average reciprocal distance (ARD) criterion that can explicitly account for projections onto user-specified r-dimensional subspaces of the input space. Welch (1985) and Liefvendahl and Stocki (2006) implement and provide empirical studies of statistical features of the ARD criterion.

The vast majority of the literature on the construction of experimental designs for computer experiments, including the papers mentioned above, considers input regions that are hyperrectangles. Some contributions to the constuction of designs for non-rectangular input regions in *physical experiments* are given by Montgomery et al. (2002) and Nguyen and Piepel (2005) who present heuristic algorithms for constructing approximate D– (and other) optimal designs for polynomial models based on a gridding of a non-rectangular input region. In the *computer simulator setting*, Sasena, Papalambros and Goovaerts (2002) developed methodology to deal with global optimization over disconnected design regions. Trosset (1999) described an algorithm to construct approximate maximin designs for constrained regions by employing non-linear programming (NLP) and an L_p criterion. Stinstra et al. (2003) also used an algorithmic approach to construct maximin designs in constrained regions; as with Trosset (1999), their algorithms solve an NLP problem to obtain a constrained maximin design. Chuang and Hung (2010) developed an algorithm that produces a nearly uniform design in any convex experimental region. However, none of the procedures mentioned above guarantee the resulting design to be non-collapsing.

This paper introduces the CoNcaD algorithm for constructing non-collapsing and space-filling designs for a computer experiment whose input region need not be rectangular. Sec-

tion 2 discusses some criteria that can be used to define space-fillingness. Section 3 gives details on the proposed algorithm, CoNcaD, while Section 4 describes some important technicalities that are essential in the design building process. Section 5 explores some additional applications of CoNcaD. Section 6 considers the choice of tuning parameters and discusses the properties of the designs obtained using CoNcaD for the two experiments introduced in this section and the criteria highlighted in Section 2.

2 SPACE-FILLING DESIGN CRITERIA

This section gives a brief review of the space-filling criteria used in this paper. The maximin (Mm) criterion (Johnson et al.; 1990) is an intuitive design construction criterion whose goal is to prevent design points from being too close to each other, thereby ensuring that they are spread across the design space. The Mm criterion accomplishes this task by employing a measure which quantifies the distance between the design points. Let p denote the number of inputs, $\mathcal{X} \subset \mathbb{R}^p$ denote the set of all feasible design points, and let the rows of $n \times p$ matrix $\boldsymbol{X}$ represent n design points selected from $\mathcal{X}$. The set of all possible $n \times p$ design matrices is labeled $\mathcal{D}$. Let $\boldsymbol{x}_i$ represent the i^{th} row of $\boldsymbol{X}$, i.e. the setting of inputs at which the i^{th} computer simulator run will be made.

Define the z^{th} order distance, ρ_z, between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in $\mathcal{X}$ as,

$$\rho_z(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left[\sum_{\ell=1}^{p} |x_{i\ell} - x_{j\ell}|^z\right]^{1/z} \tag{1}$$

where $x_{i\ell}$ is the ℓ^{th} element of $\boldsymbol{x}_i$. A design $\boldsymbol{X}_{Mm} \in \mathcal{D}$ is *maximin* provided it maximizes the minimum interpoint distance (MIPD) between pairs of design points, i.e.,

$$\min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{X}_{Mm}} \rho_z(\boldsymbol{x}_1, \boldsymbol{x}_2) = \max_{\boldsymbol{X} \in \mathcal{D}} \left[\min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{X}} \rho_z(\boldsymbol{x}_1, \boldsymbol{x}_2)\right]. \tag{2}$$

Audze and Eglais (1977) introduced a criterion for space-fillingness in which the average reciprocal distance between points in the design is minimized. We use a modification of this

criterion that adds a requirement that the selected design have good projection properties onto a given set of subspaces of the p-dimensional design space; this same requirement could be added to the minimax or maximin criterion. Suppose the projections of a design $\boldsymbol{X} \in \mathcal{D}$ onto all the subspaces defined by an index set $\boldsymbol{J} \subseteq \{1, 2, \ldots, p\}$ are to be considered. For each $j \in \boldsymbol{J}$ let $\boldsymbol{X}_{kj}$, $1 \leq k \leq \binom{p}{j}$ denote the k^{th} design, ordered in some fashion, among the $\binom{p}{j}$ j-dimensional projections of the points in $\boldsymbol{X}$.

Assuming $\mathcal{X} \subset [0,1]^p$ and given fixed, real numbers $z \geq 1$ and $\mu \geq 1$, a design $\boldsymbol{X}_{ARD} \in \mathcal{D}$ is said to be minimum ARD provided it *minimizes* the average reciprocal distance (ARD) projection function

$$av_{(z,\mu)}(\boldsymbol{X}) = \left(\frac{1}{\binom{n}{2} \sum_{j \in \boldsymbol{J}} \binom{p}{j}} \sum_{j \in \boldsymbol{J}} \sum_{k=1}^{\binom{p}{j}} \sum_{\boldsymbol{x}_h^\star, \boldsymbol{x}_i^\star \in \boldsymbol{X}_{kj}} \left[\frac{j^{1/z}}{\rho_z(\boldsymbol{x}_h^\star, \boldsymbol{x}_i^\star)} \right]^\mu \right)^{1/\mu} \tag{3}$$

among all $\boldsymbol{X} \in \mathcal{D}$. In (3), $\boldsymbol{x}_h^\star$ denotes the projection of $\boldsymbol{x}_h$ onto $\boldsymbol{X}_{kj}$. The numerator, $j^{1/z}$, acts as a scaling factor for points in $[0,1]^j$ in that $j^{1/z}$ is the maximum distance apart of any two points in j-dimensional space. Thus a minimum ARD design is space-filling for projections in all sub-dimensions defined by the set $\boldsymbol{J}$. The ARD, at least for $J = \{p\}$ and Euclidean distance, can be interpreted as the minimum of the potential energy of repulsive forces for a set of points of unit mass assuming that the magnitude of these repulsive forces is inversely proportional to the squared distance between the points.

The CoNcaD algorithm is presented in Section 3 for non-collapsing designs using the MIPD, ARD, or a criterion that accounts for both MIPD and ARD, but it is important to note that these critria can be replaced by any other space-filling criterion of the user's choosing.

Calculating the ARD in (3) for a large design when the number of projections is also large is computationally burdensome. To simplify the computation, we develop a one-point *update* formula for $av_{(z,1)}(\boldsymbol{X})$ when $\mu = 1$ and restrict attention to $\mu = 1$ hereafter. Let $\boldsymbol{X} \in \mathcal{D}$ be an $(n-1) \times p$ non-collapsing design with $n - 1 \geq 2$ rows and $\boldsymbol{w}$ be a $1 \times p$ row vector representing a point in the design space $\mathcal{X}$ such that $\boldsymbol{X}$ augmented by $\boldsymbol{w}$ is also a

non-collapsing design. Then

$$
\begin{aligned}
av_{(z,1)}\left(\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{w}\end{bmatrix}\right) &= \frac{1}{\binom{n}{2}\sum_{j\in\boldsymbol{J}}\binom{p}{j}}\sum_{j\in\boldsymbol{J}}\sum_{k=1}^{\binom{p}{j}}\left[\sum_{\boldsymbol{x}_h^\star,\boldsymbol{x}_u^\star\in\boldsymbol{X}_{kj}}\frac{j^{1/z}}{\rho_z(\boldsymbol{x}_h^\star,\boldsymbol{x}_u^\star)}+\sum_{i=1}^{n-1}\frac{j^{1/z}}{\rho_z(\boldsymbol{x}_i^\star,\boldsymbol{w})}\right]\\
&= \frac{\binom{n-1}{2}}{\binom{n}{2}\sum_{j\in\boldsymbol{J}}\binom{p}{j}}\sum_{j\in\boldsymbol{J}}\sum_{k=1}^{\binom{p}{j}}\left[\frac{1}{\binom{n-1}{2}}\sum_{\boldsymbol{x}_h^\star,\boldsymbol{x}_u^\star\in\boldsymbol{X}_{kj}}\frac{j^{1/z}}{\rho_z(\boldsymbol{x}_h^\star,\boldsymbol{x}_u^\star)}\right]\\
&\quad+\frac{1}{\binom{n}{2}\sum_{j\in\boldsymbol{J}}\binom{p}{j}}\sum_{j\in\boldsymbol{J}}\sum_{k=1}^{\binom{p}{j}}\left[\sum_{i=1}^{n-1}\frac{j^{1/z}}{\rho_z(\boldsymbol{x}_i^\star,\boldsymbol{w})}\right] \qquad (4)\\
&= \frac{n-2}{n}av_{(z,1)}(\boldsymbol{X})+\Delta,
\end{aligned}
$$

where Δ is the second term in (4). Thus only Δ need be computed in the search for an additional design point, avoiding the unnecessary computation of $\binom{n-1}{2}$ interpoint reciprocal distances for every $\boldsymbol{X}_{kj}$, $j\in\boldsymbol{J}$, $k=1,\ldots,\binom{p}{j}$. In the algorithm, the updating formula is applied within s-dimensional subspaces of the p-dimensional space. An analogous formula for updating the MIPD in (2) can be developed. For the remainder of the paper, only Euclidean distance between the design points will be considered ($z=2$ in (1)).

3 THE CoNcaD ALGORITHM

Existing results on the construction of designs for rectangular input regions under different space-filling criteria are difficult to extend to non-rectangular input regions because the levels for different inputs cannot be chosen independently. One could proceed to obtain an optimal design by (i) constructing an exhaustive p-dimensional grid of points, (ii) removing those points from the grid which do not satisfy the constraints, and (iii) choosing design points from the remaining sub-grid which satisfy a desired criterion. However, even with the today's technology, this brute force method becomes infeasible even when p is rather small.

The CoNcaD algorithm proposed in this paper constructs a design by adding columns sequentially to the design matrix, thus removing the computational burden of searching within a large p-dimensional grid. CoNcaD chooses the values for the first input variable from a one-dimensional grid of valid points. These selected values represent the first column

of the design matrix $\boldsymbol{X}$. Given selected values for the first input and taking into account the constraints and the optimality criterion, the design values for the second input are chosen from an appropriate one-dimensional grid of points. The algorithm adds further columns to $\boldsymbol{X}$, adapting the grid dynamically to be non-collapsing until values for all p inputs are chosen. The details are given below.

Before the design construction begins, the algorithm scales all inputs so that each input has marginal range [0,1]. For example, if the design region of the form $\boldsymbol{Ax} \leq \boldsymbol{b}$, this scaling is determined from the lower and upper extrema for x_k, $k = 1, \ldots, p$, which are obtained by solving the following pair of linear programming problems:

$$\min x_k \quad \text{subject to} \begin{cases} \boldsymbol{Ax} \leq \boldsymbol{b} \\ \boldsymbol{L} \leq \boldsymbol{x} \leq \boldsymbol{U} \end{cases} \quad \text{and} \quad \max x_k \quad \text{subject to} \begin{cases} \boldsymbol{Ax} \leq \boldsymbol{b} \\ \boldsymbol{L} \leq \boldsymbol{x} \leq \boldsymbol{U} \end{cases} \tag{5}$$

where $\boldsymbol{A}$ is a matrix containing *joint* constraints while $\boldsymbol{L}$ and $\boldsymbol{U}$ are vectors which contain any explicitly stated lower and upper bounds for *individual* x_k. Design regions which are not polytopes will use other formulations to solve the marginal extremal problems, (e.g. Section 5).

In Example 1.1, x_1 and x_4 have individual bounds [0, 10] and [-15, 15], respectively, which determine $\boldsymbol{L}$ and $\boldsymbol{U}$ while the ranges for x_2 and x_3 must be derived from the joint constraints. Solving (5) for the constraints of this example, we obtain $x_2^{\min} = -5$, $x_2^{\max} = 5$ and $x_3^{\min} = -2$, $x_3^{\max} = 2$.

In general, given its extrema, say (x_k^{min}, x_k^{max}), x_k is transformed to [0, 1] by

$$x_k^{scaled} = \frac{x_k - x_k^{min}}{x_k^{max} - x_k^{min}}, \quad k = 1, \ldots, p.$$

Then $x_k = (x_k^{max} - x_k^{min}) x_k^{scaled} + x_k^{min}$ is substituted into $\boldsymbol{Ax} \leq \boldsymbol{b}$, $k = 1, \ldots, p$, to obtain $\boldsymbol{A}^{scaled} \boldsymbol{x}^{scaled} \leq \boldsymbol{b}^{scaled}$. From now on, it will be assumed that the scaling has been performed and, for simplicity, the "scaled" superscript notation will be dropped.

As described above, the algorithm selects design point values for each input sequentially. At the conclusion of each step s with $s < p$, a matrix $\boldsymbol{M}^{(s)}$ having Q ($\geq n$) rows and

s columns is produced as the "current design matrix" for the first s inputs. The rows of $\boldsymbol{M}^{(s)}$ are selected from a matrix $\boldsymbol{C}^{(s)}$ which holds the candidate design points. The $\boldsymbol{M}^{(s)}$ constructed in one step provides the foundation for constructing $\boldsymbol{C}^{(s+1)}$, and hence $\boldsymbol{M}^{(s+1)}$, in the next step. The integer Q is a tuning parameter whose function is to allow greater flexibility to the algorithm in constructing designs by allowing it to carry more than the required number of n points until the final step. The effect of the magnitude of Q on the design characteristics is discussed in Section 6.

One component of the algorithm is the elimination at step s, $s < p$, of those rows $(x_1, x_2, \ldots, x_s)$ from a given candidate set $\boldsymbol{C}^{(s)}$ for which it is *impossible* for these s values to result in a design point $\boldsymbol{x} = (x_1, x_2, \ldots, x_s, \ldots, x_p)$ that satisfies $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$. The matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ contain the simultaneous constraints involving all p inputs. Let $\boldsymbol{C}^{(s)}$ denote and initially proposed set of candidates that might be added to $\boldsymbol{M}^{(s-1)}$ and let $\boldsymbol{C}^{(s)}(i,:)$ denote the ith row of $\boldsymbol{C}^{(s)}$. Row $\boldsymbol{C}^{(s)}(i,:)$ is eliminated $\boldsymbol{C}^{(s)}$ if $\boldsymbol{A}^{(s)}(\boldsymbol{C}^{(s)}(i,:))^\top > \boldsymbol{b}^{(s)}$ where $\boldsymbol{A}^{(s)}$ contains the first s columns of $\boldsymbol{A}$ and a subset of the rows of $\boldsymbol{A}$. The rows $(a_{i,1}, \ldots, a_{i,p})$ selected from $\boldsymbol{A}$ have $a_{i,1}, \ldots, a_{i,s}$ are not all zero, and all the remaining $(p-s)$ entries, $a_{i,s+1}$, $\ldots$, $a_{i,p}$, non-negative. The following example illustrates these ideas.

Example 3.1 Suppose the constrained region for inputs (x_1, x_2, x_3) is defined by $0 \leq x_i \leq 1$ for $i = 1, 2, 3$, $0.2 \leq x_2 + x_3 \leq 0.6$, $0.2 \leq x_1 + x_2 - x_3 \leq 0.7$, and $\sum_{i=1}^{3} x_i \leq 0.8$, then

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & -1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 0.6 \\ -0.2 \\ 0.7 \\ -0.2 \\ 0.8 \end{bmatrix}, \boldsymbol{L} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{ and } \boldsymbol{U} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{6}$$

When $s = 1$, the constraints for x_1 are determined by $\boldsymbol{A}^{(1)} = [1]$ and $\boldsymbol{b}^{(1)} = [0.8]$. This is because the elements in Rows 1 and 2 of the first column of $\boldsymbol{A}$ are zero and place no constraint on x_1. Rows 3 and 4 of $\boldsymbol{A}$ do not put *definitive* restrictions on x_1 because of the negative coefficients of x_3 and x_2 in these rows, respectively, mean that linear combinations involving (x_1, x_2, x_3) might indeed satisfy $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ even if x_1 is greater than the corresponding element

of $\boldsymbol{b}$. Thus, the only definitive conclusion is that any x_1 listed in $\boldsymbol{C}^{(1)}$ that is greater than 0.8 can be eliminated.

Continuing the example, for $s = 2$, the first and last two rows of $\boldsymbol{A}$ and $\boldsymbol{b}$ form constraints that need to be considered, giving

$$\boldsymbol{A}^{(2)} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix} \text{ and } \boldsymbol{b}^{(2)} = \begin{bmatrix} 0.6 \\ -0.2 \\ 0.8 \end{bmatrix}.$$

Then, given $\boldsymbol{A}$ and the values of x_1, the first constraint implies that x_2 values in the final design have to be less than or equal to 0.6, while the second and third constraints imply that x_2 has to satisfy $(0.2 - x_1) \leq x_2 \leq (0.8 - x_1)$. Any rows of $\boldsymbol{C}^{(2)}$ which do not conform to $\boldsymbol{A}^{(2)}(\boldsymbol{C}^{(2)})^\top \leq \boldsymbol{b}^{(2)}$ are eliminated. At the last step, when $s = 3$, $\boldsymbol{A}^{(3)} = \boldsymbol{A}$ and $\boldsymbol{b}^{(3)} = \boldsymbol{b}$ and the rows of $\boldsymbol{C}^{(3)}$ for which $\boldsymbol{A}^{(3)}(\boldsymbol{C}^{(3)})^\top \leq \boldsymbol{b}^{(3)}$ does not hold are deleted.

The CoNcaD algorithm is flexible in that allows the user to choose among multiple criteria. Specifically, our current implementation allows the choice of either the maximin criterion (2), *or* the minimum ARD criterion which minimizes (3) with user-specified $\boldsymbol{J}$, *or* considers a combination of both criteria, by minimizing the quantity

$$\alpha \text{Rank}_{MIPD} + (1 - \alpha)\text{Rank}_{ARD} \tag{7}$$

for a user-specified α, $0 \leq \alpha \leq 1$. Larger values of α put more weight on the maximin characteristic of the design while the smallest values favor an ARD optimized design. It is important to note, however, that any other criterion could be subsituted for (7).

The criterion (7) is applied as follows. Given the current design matrix $\boldsymbol{M}^{(s)}$ for inputs $1, 2, \ldots, s$ $(\leq p)$ and n_s $(\leq Q)$ points, each row in the candidate set $\boldsymbol{C}^{(s)}$ is added in turn to $\boldsymbol{M}^{(s)}$ and the minimum interpoint distance (MIPD) and Δ shown in (4) are computed. Then all points in $\boldsymbol{C}^{(s)}$ are ranked with that point having the *largest* MIPD receiving rank 1 and that point with the *smallest* Δ value receiving rank 1. The combined ranking for each of the points in $\boldsymbol{C}^{(s)}$ is computed from (7) using the given α. The point with the lowest value of (7) is added to $\boldsymbol{M}^{(s)}$.

The steps of the CoNcaD algorithm are summarized as follows.

Step 1: Fix the desired design size (n, p), input constraints $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{L}, \boldsymbol{U})$, tuning parameters $Q \geq n$ and $u \geq 2$, and the design objective (7), or user-specified replacement. Construct the grid vector $\boldsymbol{m}$ whose i^{th} element is $m_i = i/u$ where $i = 0, 1, \ldots, u$. Set $s = 1$. Go to Step 2.

Step 2: If $s = 1$ then set $\boldsymbol{C}^{(1)} = \boldsymbol{m}$ and proceed to Step 4, otherwise go to Step 3.

Step 3: Given $2 \leq s \leq p$ and $\boldsymbol{M}^{(s-1)}$, construct the $Qu \times s$ candidate set $\boldsymbol{C}^{(s)}$ by

$$\boldsymbol{C}^{(s)} = \begin{bmatrix} \boldsymbol{M}^{(s-1)} & m_1 \mathbf{1}_Q \\ \boldsymbol{M}^{(s-1)} & m_2 \mathbf{1}_Q \\ \vdots & \vdots \\ \boldsymbol{M}^{(s-1)} & m_u \mathbf{1}_Q \end{bmatrix}$$

where $\mathbf{1}_Q$ is a $Q \times 1$ vector of ones; thus each m_i is appended to a copy of each row of $\boldsymbol{M}^{(s-1)}$.

Step 4: Let $\boldsymbol{C}^{(s)}(i,:)$ denote the i^{th} row of $\boldsymbol{C}^{(s)}$. Construct $\boldsymbol{A}^{(s)}$ and $\boldsymbol{b}^{(s)}$ as in the discussion above. Eliminate those rows $\boldsymbol{C}^{(s)}(i,:)$ from $\boldsymbol{C}^{(s)}$ for which $\boldsymbol{A}^{(s)}(\boldsymbol{C}^{(s)}(i,:))^\top > \boldsymbol{b}^{(s)}$ in any component.

Step 5: Choose the first row for $\boldsymbol{M}^{(s)}$ from $\boldsymbol{C}^{(s)}$ randomly.

Step 6: As described in Section 4, adapt $\boldsymbol{C}^{(s)}$ to implement non-collapsingness.

Step 7: If the number of rows of $\boldsymbol{M}^{(s)} \geq Q$ go to Step 8. Otherwise select that row $\boldsymbol{C}^{(s)}(i,:)$ of $\boldsymbol{C}^{(s)}$ so that the augmented design

$$\boldsymbol{M}^{(s)} = \begin{bmatrix} \boldsymbol{M}^{(s)} \\ \boldsymbol{C}^{(s)}(i,:) \end{bmatrix}$$

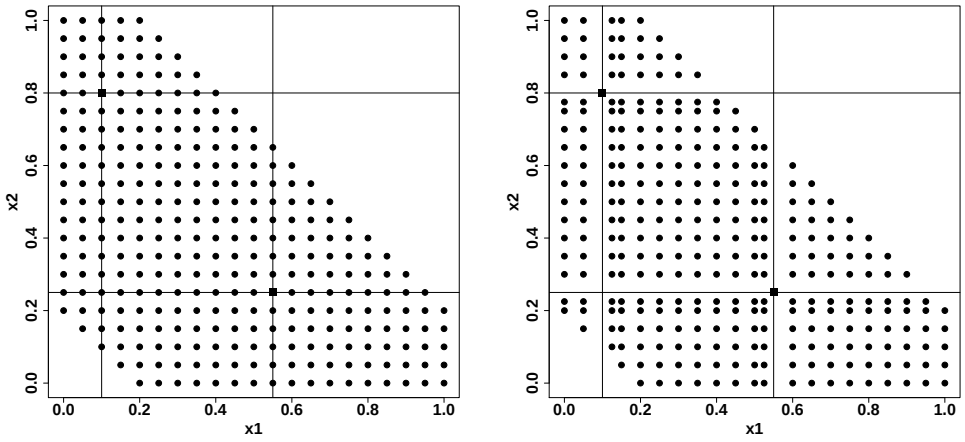
minimizes (7). Add this row to $\boldsymbol{M}^{(s)}$ and eliminate it from $\boldsymbol{C}^{(s)}$. Go to Step 6.

Step 8: Set $s = s + 1$. If $s < p$, then go to Step 3. If $s = p$, then set $Q = n$ and go to Step 3. If $s = p + 1$, then set $\boldsymbol{X} = \boldsymbol{M}^{(p)}$ and finish.

The following section describes, the grid adaptation used in Step 6. This step provides the algorithm with the necessary flexibility to handle high-dimensional design construction.

4 GRID ADAPTATION

Step 6 allows CoNcaD to build designs with unique input values for each one of the p input variables, i.e. non-collapsing designs. Step 6 also provides flexibility to the design building process because, without it, the candidate set $\boldsymbol{C}^{(s)}$ could become empty before the final n-point design has been constructed. To illustrate this last point in a visually simple setting, consider the candidate set $\boldsymbol{C}^{(2)}$ shown in Figure 1(a) that is associated with the region constrained by $0.2 \leq x_1 + x_2 \leq 1.2$. Suppose that the first two points selected are (0.1, 0.8)



(a) Without Adaptation (b) With Adaptation

Figure 1: Candidate Set After Choosing Two Points Randomly

and (0.55, 0.25); Figure 1(b) plots these two points as black squares at the intersections of the solid lines. If there is no adaptation of the grid, then all other points in Figure 1(a) that lie on the vertical or a horizontal lines through these points must be removed from $\boldsymbol{C}^{(2)}$ in order for the remaining selection of (x_1, x_2) points to produce a non-collapsing design. In this case, only 19 unique x_1 coordinates remain and thus designs with $n > 21$ design points are infeasible.

In contrast, Figure 1(b) shows the same starting $\boldsymbol{C}^{(2)}$ with the grid adaptation employed by CoNcaD. Adaptation *replaces* each point in $\boldsymbol{C}^{(2)}$ whose x_1 value is equal to 0.1 or 0.55 or whose x_2 value is equal to 0.8 or 0.25 by alternative nearby values. For example, in Figure 1(b) the lines $x_1 = 0.1$ and $x_1 = 0.55$ have been replaced by the lines $x_1 = 0.125$ and $x_1 = 0.525$, respectively. The value of 0.125 lies half way between the chosen value 0.1 and the next larger x_1 value in $\boldsymbol{C}^{(2)}$ which is 0.15, and, hence, does not coincide with any previously listed x_1 values in $\boldsymbol{C}^{(2)}$. The same idea is applied to the second x_1 value of 0.55 which is replaced by $x_1 = 0.525$, half way between 0.55 and the next lowest value in both $\boldsymbol{C}^{(2)}$ and $\boldsymbol{M}^{(2)}$. Similarly, the horizontal lines $x_2 = 0.8$ and $x_2 = 0.25$ are replaced by the lines $x_2 = 0.775$ and $x_2 = 0.225$, respectively. Apart from any new points falling outside the feasible design region, such an adaptation leaves the number of candidate points remains unchanged.

A similar adaptation is applied to each of the inputs $x_1, x_2, \ldots, x_s$ at step s of the algorithm ($s = 3, \ldots, p$). The new value of x_i is taken to be half way between the chosen value of x_i and either the next larger or the next smaller value in the combined $\boldsymbol{C}^{(s)}$ and $\boldsymbol{M}^{(s)}$, whichever is further away. A random selection is made if the distance above and below the chosen point is the same.

5 OTHER APPLICATIONS OF CoNcaD

The CoNcaD algorithm, described in Section 3, constructs space-filling designs for input regions of the form $\boldsymbol{Ax} \leq \boldsymbol{b}$. This section gives examples of the use of the basic algorithm in two other settings.

5.1 Use in Bounded Non-polygonal Regions

CoNcaD can be adapted for use with non-polygonal bounded regions by replacing $\boldsymbol{Ax} \leq \boldsymbol{b}$ in Step 4 of the algorithm by whichever constraints define the constrained region. All other

steps of the algorithm remain the same. As an example, the results of such a modification are shown below for a quadrant/ball region defined by the constraints $0 \leq x_i \leq 1, i = 1, \ldots, p$, and $\sum_{i=1}^{p} x_i^2 \leq 1$ in $p = 2$ and $p = 10$ dimensions. Table 1 shows the maximum MIPD for designs produced by the modified CoNcaD algorithm using 100 random starts, $\alpha = 1$, and $(Q, u) = (3n, 100)$. For comparison, the results for these problems using SFDP* and SFDP** and reported by Stinstra et al. (2003) (in their Table 2) are also listed. Lastly, Table 1 shows the results for this problem using L_p-optimization as described by Trosset (1999) and again reported by Stinstra et al. (2003). For large $n \times p$, the SFDP* formulation required too many constraints to be implemented using the non-linear programming software and hardware available to Stinstra et al. (2003), but their SFDP** Gauss Seidel type algorithm, was able to obtain solutions for all the quadrant/ball cases considered in Table 1.

The designs produced by the SFDP methods are not guaranteed to be non-collapsing, and, indeed, can be collapsing, as their figures illustrate. Under the maximin criterion, even with the non-collapsing constraint, CoNcaD gives maximum MIPD values that are close to those of SFDP* when the latter is implementable. The designs produced by CoNcaD have maximum MIPD values which are either larger than, or close to, those for designs produced by SFDP**. For $p = 10$, the designs produced by the L_p algorithm are noticably worse than those of any of the other three methods. Finally, we note that, if ARD- or weighted Mm/ARD- optimal designs are desired, then the algorithms of Stinstra et al. (2003) and Trosset (1999) cannot be used but CoNcaD is able to construct non-collapsing designs that have good properties for these criteria.

5.2 Design Augmentation

Another application of the CoNcaD algorithm is to construct augmented designs as would, for example, be required to construct a sequence of nested designs. Qian (2009) notes that such designs are natural to use in performing multiple computer experiments with different levels of accuracy. Given an *arbitrary* design $\boldsymbol{X}$, where $\boldsymbol{X}$ need not be non-collapsing nor

Table 1: Minimum Interpoint Distances for Two- and Ten-Dimensional Ball Examples (the number of constraints was too large for SFDP* to be solved for NA cased)

p	n	SFDP*	SFDP**	L_p	CoNcaD	Ave. CoNcaD Run Time (sec)
2	10	0.3587	0.3522	0.2554	0.3400	0.29
	20	0.2394	0.1966	0.2212	0.2124	1.31
	50	0.1403	0.1188	0.1336	0.1204	9.00
	100	NA	0.0486	0.0660	0.0789	59.13
	200	NA	0.0259	0.0070	0.0539	302.75
10	10	1.4142	1.2167	0.5017	1.3027	5.62
	20	1.0323	0.9591	0.4138	0.8364	29.24
	50	0.8811	0.7588	0.2952	0.6747	296.89
	100	NA	0.6248	0.2753	0.5838	1915.44
	200	NA	0.5235	0.2559	0.5160	21107.09

satisfy constraints imposed upon the additional design points, Step 5 can be modified to incorporate all rows from $\boldsymbol{X}$ into the new design before any choices are added from the candidate set. In order to allow initial designs $\boldsymbol{X}$ that contain replicates or are collapsing, the calculation of the desired distance measure (MIPD and/or ARD) omits distances between pairs of design points in the initial design $\boldsymbol{X}$.

As an example, Figure 2 shows a nested design constructed using this modified CoNcaD algorithm. The initial design (solid triangles) was a 10×2 maximin Latin hypercube design in $[0,1]^2$ (and had no additional constraints). An additional 10 points (solid circles) were added to optimize the Mm/ARD criterion with $(\alpha, \boldsymbol{J}) = (0.5, \{1,2\})$ and so as to satisfy $0.2 \leq x_1 + x_2 \leq 1$, yielding a final design with 20 points for two inputs. The solid lines represent the boundary of the constrained region for the augmented portion of the design.

The design shown in Figure 2 has MIPD of 0.1562 and ARD of 5.3580. Aside from illustrating the ability of CoCcaD to augment (one or several) designs, this example illustrates the difference between using the Mm and combined Mm/ARD criteria, a point that will be discussed again in Section 7. If a pure Mm criterion had been used to select the 10 additional points, there would be no "hole" in the set of additional design points shown in Figure 2.

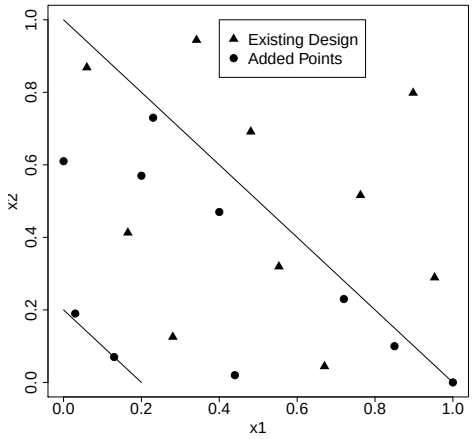


Figure 2: Augmented Design Constructed by CONcaD with $\alpha = 0.5$ and $\boldsymbol{J} = \{1, 2\}$; the first stage design is denoted by solid triangles and the second stage by solid circles

Indeed, if an 11th point were added using the combined Mm/ARD criterion selected for this example, this hole would be filled.

6 TUNING PARAMETERS AND STARTING RUNS

6.1 Choice of Tuning Parameters for Random Start Designs

The CoNcaD algorithm is not deterministic when using random starts to construct $\boldsymbol{M}^{(s)}$ and, whether using random or fixed starts, by its occasional random choice in the grid adaptation process (Section 4). Therefore, the algorithm should be run multiple times and the best design selected. The number of random starts should be selected so that CoNcaD produces stable maximum MPID and minumum ARD values and this is a function of p, u, Q, $\boldsymbol{A}$, and $\boldsymbol{b}$. We recommend $(u, Q) = (100, 3n)$ and 100 random starts. These recommendations are based on the numerical evidence from numerous examples which is illustrated for a typical case in the following paragraph.

Table 2 lists the cumulative maxima of MIPD values produced by CoNcaD after 100, 200, ..., and 500 random starts for Example 1.1 where $(n, p) = (10, 4)$ and the constraints $\boldsymbol{Ax} \leq \boldsymbol{b}$ and $\boldsymbol{L} \leq \boldsymbol{x} \leq \boldsymbol{U}$ are

$$\boldsymbol{A} = \begin{bmatrix} 0 & 5 & 2 & 0 \\ 0 & -5 & 2 & 0 \\ 0 & -5 & -2 & 0 \\ 0 & 5 & -2 & 0 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}, \boldsymbol{L} = \begin{bmatrix} 0 \\ -\infty \\ -\infty \\ -15 \end{bmatrix}, \text{ and } \boldsymbol{U} = \begin{bmatrix} 10 \\ \infty \\ \infty \\ 15 \end{bmatrix}. \tag{8}$$

The maximum MIPD values listed in Table 2 were computed for the scaled designs, i.e. the designs with $[0, 1]$ marginal ranges for all inputs and for $u \in \{20, 100, 200\}$, $Q \in \{n, 3n, 5n\}$, and $\alpha = 1$.

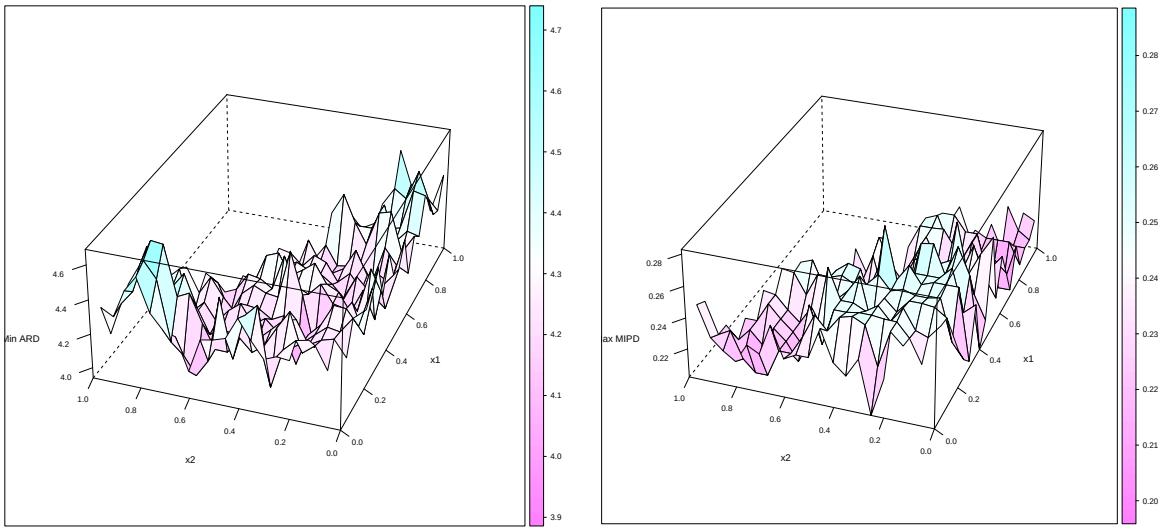
Table 2: Cumulative Maximum, after 100, 200, ..., 500 Random Starts, of the MIPD Values Produced by CoNcaD for Example 1.1 with $n = 10$ when $\alpha = 1.0$ and using various (u, Q)

u	Q	Number of Starts					For Runs 1 to 100	
		100	200	300	400	500	Avg.Time (sec)	SD (sec)
20	10	0.6793	0.6955	0.7068	0.7068	0.7068	0.1485	0.0048
	30	0.7262	0.7471	0.7471	0.7574	0.7574	0.7152	0.0396
	50	0.7644	0.7644	0.7644	0.7644	0.7644	1.8522	0.0896
100	10	0.7164	0.7164	0.7164	0.7164	0.7164	0.3264	0.0147
	30	0.7666	0.7666	0.8011	0.8011	0.8011	2.3137	0.0294
	50	0.7689	0.8044	0.8044	0.8044	0.8044	7.0379	0.3332
200	10	0.6705	0.7126	0.7126	0.7178	0.7178	0.5304	0.0234
	30	0.7626	0.7995	0.7995	0.7995	0.7995	4.3009	0.0853
	50	0.7921	0.7921	0.7929	0.7929	0.7929	12.8428	0.1930

As Q and u increase, the set of potential designs becomes denser but their evalution requires additional computing time. Table 2 shows that, as Q increases, the gain in maximum MIPD is not linear but that the increase in maximum MIPD is much greater as Q increases from 10 to 30 than from 30 to 50. Since the execution time for a $Q = 30$ design is less than half of the time for a $Q = 50$ design, the small gain in maximium MIPD for the $Q = 50$ design appears to be offset by the increase in execution time. Therefore, from this and other examples we have studied, we recommend $Q = 3n$.

Fixing $Q = 3n$, Table 2 shows clear increases in maximum MIPD values when u increases from 20 to 100, while there is no significant increase in maximum MIPD values when u increases further to 200. Thus we recommend using $u = 100$, although the average execution

time roughly triples as u increases from 20 to 100. (It further doubles as u increases from 100 to 200). Finally we note that, as can be seen in Table 2, the increase in the maximum MIPD can improve on the order of about 3-5% as the number of random starts increases from 100 to 500 while the increase in the computational time increases by roughly 400 times the cost of a single run, in this example by 400×2.3137 sec ≈ 925 sec for $(u, Q) = (100, 30)$. Again, this is a user decision depending on the application, but for most applications using 100 starts will produce reasonable results.



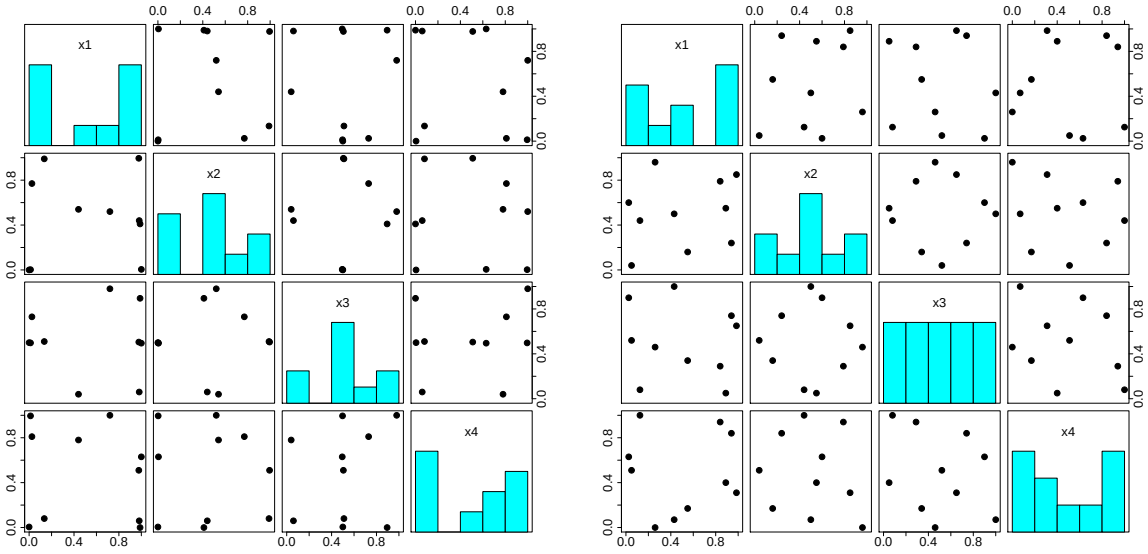
(a) ARD with $\alpha = 0.5$, $\boldsymbol{J} = \{1, 2\}$ (b) maximum MIPD with $\alpha = 1$

Figure 3: ARD and MIPD Values for the Designs with Fixed First Design Point

6.2 Deterministic Starting Runs

We next illustrate how one can produce more stable final designs by replacing random starts with deterministic ones. This is accomplished by fixing a starting point, i.e., choosing a starting design $\boldsymbol{X}$ with a *single* row and letting the algorithm construct the remaining rows. The only randomness in the execution of the algorithm results from the gridding process. This stability is illustrated for a variant of Example 1.2 when $T = 2$ where a grid with mesh size 0.05 was constructed in the $2-d$ polygonal input region $0.2 \leq x_1 + x_2 \leq 1.0$ with $0.0 \leq x_1, x_2 \leq 1.0$ that is shown in Figure 2. Each of the grid points was used, in turn,

as a starting point to construct ten Mm/ARD ($\alpha = 0.5$) and ten Mm ($\alpha = 1.0$) designs where $(u, Q) = (100, 3n)$. For *every* fixed grid start point, all ten Mm/ARD designs had identical ARD values; these values are plotted in Figure 3(a). Most, but not all, of the fixed start points resulted in Mm designs that had identical MIPD values. Figure 3(b) shows the maximum MIPD values for each set of ten Mm designs. Because smaller values are desired for ARD and larger values are preferable for MIPD, Figure 3 suggests that it is more reasonable to take a grid of starting design points centered at the *centroid of the constrained region* and use these as fixed starts instead of using random starts. The same conclusions hold when the constrained region is defined as in Example 1.1.



(a) $\alpha = 1$ (b) $\alpha = 0.5$, $\boldsymbol{J} = \{1, 2\}$

Figure 4: Two-dimensional Projections of the Best Mm and ARD Designs for Example 1.1

7 SUMMARY and DISCUSSION

This paper describes an algorithm, CoNcaD, for constructing space-filling, non-collapsing designs for computer experiments where the input space can be a bounded non-rectangular region and that allows the user to choose among multiple design criteria. The use of this algorithm is illustrated using two (hyper-)polygonal regions arising from the engineering

applications, Examples 1.1 and 1.2. Examples for 2- and 10-dimensional balls are given in Section 5.1. It is possible that points produced by this algorithm may lie on a set of parallel planes and hence be collapsing with respect to an affine transformation of the input space; in our experience this phenomenon has only been seen to occur in problems when the maximin criterion puts points on a bounding facet of the input space. In Example 1.1, the constraints on x_1, x_4 form a rectangular region, but those on x_2, x_3 for a diamond-shaped region. Although the x_2, x_3 space could be rotated, a non-collapsing design in the rotated space does not guarantee a non-collapsing design in the original space.

The algorithm can also be used to augment a given design as illustrated in Section 5.2. Comparisons of the maximum MIPDs for CoNcaD designs with some maximin (hyper-rectangular) LHDs centered on the subcubes defining the LHD are given in Supplementary Material I and further exploration of the abilities of the algorithm to produce good designs for thin regions is given in Supplementary Material II.

The code that implements CoNcaD for bounded polygons $\boldsymbol{Ax} \leq \boldsymbol{b}$ is available at `http://www.stat.osu.edu/~comp_exp/CoNcaD`; it is written in the open-source software R.

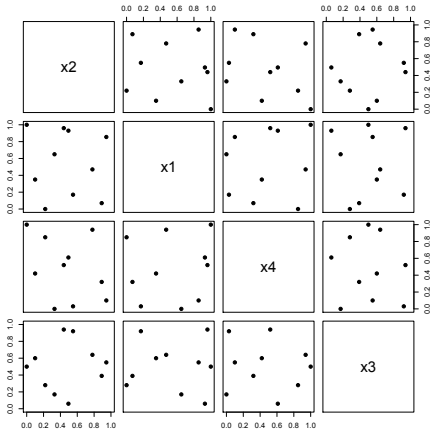


Figure 5: Two-dimensional Projections of the Best $\alpha = .05, \boldsymbol{J} = \{1, 2\}$ Design for Example 1.1 with 500 Runs and Column Order x_2, x_1, x_4, x_3

Our empirical work shows that the distance criteria and projection properties of the final non-collapsing designs appear to be reasonably invariant to the sequential nature of

the algorithm. To illustrate, Figure 5 displays two-dimensional projections for a design constructed from the constraints of Example 1.1, where the variables were added in the order x_2, x_1, x_4, x_3 instead of the order x_1, x_2, x_3, x_4 shown in Figure 4(b). The MIPD and ARD values for the best design found from 500 random starts of the algorithm for the order x_2, x_1, x_4, x_3 are 0.5555 and 3.2208, respectively as compared with the values 0.6348 and 3.1670 for the original input ordering.

A question raised by a referee of this paper which deserves further research is that of perturbing or "jittering" the design points in a maximin (or other type of space-filling) collapsing design to make it non-collapsing; in particular, under the maximin criterion, "how uniform the projections can be made with this jittering operation without substantially changing the criterion value". Certainly, the size of the jitter would need to be less than .25 times the grid spacing, otherwise different pairs of points could collapse after jittering. But not all jitters of this magnitude would result in larger distances than the cirrent minimum and not all jitters would be within the boundaries of the non-rectangular design space. For other criteria, such as ARD, which involve all the design points in calculating the criterion value, the question becomes more complicated.

If the experimenter's main goal is to obtain a design whose points are as far apart as possible in p-dimensional space, then the usual maximin computed in in p-dimensional space is clearly the best choice of design criterion. However, such designs completely ignore the projection properties of the design, which is undesirable if there are inputs which do not affect the response. In such a case, Mm/ARD designs, with say $\alpha = 0.5$ provide an improvement over maximin designs. An investigation of the trade-off between these two criteria is given in the example below.

Example 1.1 (Continued)

We illustrate the trade-off between the best Mm and the best compromise Mm/ARD designs for the $(n, p) = (10, 4)$ experiment of Example 1.1. The best $\alpha = 1$ design produced by CoNcaD in Table 2 when $(u, Q) = (100, 3n)$ with 500 random starts has MIPD value

0.8011 and ARD of 11.6503; this design required an average of 2.3137 seconds to compute each component design over the 500 starts. In contrast, the best $(\alpha, \boldsymbol{J}) = (0.5, \{1, 2\})$ Mm/ARD design for this same selection of CoNcaD tuning parameters, using 500 random starts in order to be comparable, shows roughly a 370% improvement in the ARD which decreases to 3.1481, with 28% degradation in MIPD value to 0.5768; however, an average of 241.0498 seconds were required to compute each component design over the 500 starts. In addition, the best compromise Mm/ARD design which is shown in Figure 4(b) has visually more attractive one- and two-dimensional projections than the Mm design (Figure 4(a)). This is because the Mm designs tend to explore the corners of the space first, resulting in poor exploration of the center of the design region. To contrast the use of CoNcaD with recommended number of 100 random starts for this problem results in a design with MIPD of 0.5959, ARD of 3.1434, and required a total of 278 minutes to construct.

Acknowledgments

The authors wish to thank the referees and an associate editor for suggestions which led to the improvement of this paper. This research was sponsored, in part, by the National Science Foundation under Agreements DMS-0806134 (The Ohio State University). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The paper was revised while the last two authors were visiting the Isaac Newton Institute, Cambridge, UK.

References

Audze, P. and Eglais, V. (1977). New approach for Planning out of Experiments. *Problems of Dynamics and Strengths* **35**, 104–107.

Bursztyn, D. and Steinberg, D. M. (2006). Comparison of designs for computer experiments. *Journal of Statistical Planning and Inference* **136**, 1103–1119.

Chuang, S. C. and Hung, Y. C. (2010). Uniform designs over general input domains with applications to target region estimation in computer experiments. *Computational Statistics and Data Analysis* **54**(1), 219–232.

Fang, K.-T., Lin, D. K. J., Winker, P. and Zhang, Y. (2000). Uniform Design: Theory and Application. *Technometrics* **42**(3), 237–248.

Hayeck, G. T. (2009). *The kinematics of the upper extremity and subsequent effects on joint loading and surgical treatment*. PhD thesis. Cornell University.

Johnson, M. E., Moore, L. M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26**, 131–148.

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

McKay, M. D., Beckman, R. J. and Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **21**, 239–245.

Montgomery, D. C., Loredo, E. N., Jearkpaporn, D. and Caner, M. (2002). Experimental Designs for Constrained Regions. *Quality Engineering* **14**(4), 587–601.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference* **43**, 381–402.

Nekkanty, S. (2009). *Characterization of damage and optimization of thin film coatings on ductile substrates*. PhD thesis. The Ohio State University.

Nelson, B. L. (1995). *Stochastic Modeling: Analysis And Simulation*. Dover Publications, Inc., New York.

Nguyen, N.-K. and Piepel, G. F. (2005). Computer-Generated Experimental Designs for Irregular-Shaped Regions. *Quality Technology & Quantitative Management* **2**(2), 147–160.

Qian, P. Z. G. (2009). Nested Latin hypercube designs. *Biometrika* **96**, 957–970.

Santner, T. J., Williams, B. J. and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. Springer Verlag, New York.

Sasena, M., Papalambros, P. and Goovaerts, P. (2002). Global Optimization of Problems with Disconnected Feasible Regions via Surrogate Modeling. *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. AIAA-2002-557.

Simpson, T. W., Lin, D. K. J. and Chen, W. (2001). Sampling strategies for computer experiments: Design and Analysis. *International Journal of Reliability and Applications* **2**(3), 209–240.

Stinstra, E., den Hertog, D., Stehouwer, P. and Vestjens, A. (2003). Constrained Maximin Designs for Computer Experiments. *Technometrics* **45**(4), 340–346.

Subramanian, C. and Strafford, K. N. (1993). Review of multicomponent and multilayer coatings for tribological applications. *Wear* **165**(1), 85–95.

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

Trosset, M. W. (1999). Approximate Maximin Distance Designs. *ASA Proceedings of the Section on Physical and Engineering Sciences* (), pp. 223–227, American Statistical Association (Alexandria, VA).

van Dam, E. (2008). Two-dimensional minimax Latin hypercube designs. *Discrete Applied Mathematics* **156**(18), 3483–3493.

Welch, W. J. (1985). ACED: Algorithms for the Construction of Experimental Designs. *The American Statistician* **39**(2), 146.