Sequential Designs for Filling Output Spaces

Shangkun Wang, Adam P. Generale, Surya R. Kalidindi & V. Roshan Joseph

Taylor & Francis
Taylor & Francis Group

Check for updates

Sequential Designs for Filling Output Spaces

Shangkun Wang[a], Adam P. Generale[b], Surya R. Kalidindi[b], and V. Roshan Joseph[a]

[a]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA; [b]George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA

ABSTRACT

Space-filling designs are commonly used in computer experiments to fill the space of inputs so that the input–output relationship can be accurately estimated. However, in certain applications such as inverse design or feature-based modeling, the aim is to fill the response or feature space. In this article, we propose a new experimental design framework that aims to sequentially fill the space of the outputs (responses or features). Several examples are given to show the advantages of the proposed method over the traditional input space-filling designs.

1. Introduction

Computer experiments have become an indispensable tool in science and engineering (Santner, Williams, and Notz 2019), such as in rocket engine design (Mak et al. 2018), biomedical engineering (Striegel et al. 2022), and materials design (Iyer et al. 2022). However, with the increasing complexity and resolution of the simulations, the running time of computer experiments is still far from negligible even with the current computational power. Thus, experimental designs that help to gather maximum information with minimum computational budget play a crucial role in computer experiments.

Space-filling designs are widely used as experimental designs for computer experiments (Joseph 2016). Intuitively speaking, a space-filling design tries to place the design points to "fill" the *input* space well in the hope that the estimation and prediction of the statistical model based on the experimental data would be accurate. An attractive feature of space-filling designs is that they are robust to modeling assumptions. Thus, they can be efficiently used for fitting a wide variety of models.

In this article, we are interested in experimental designs that produce response values to fill the output spaces. We refer to them as output space-filling designs (OSFD). Unlike traditional space-filling design (hereafter referred to as input space-filling design (ISFD)), OSFD aims to cover the *output* space well. Generating OSFD may seem like an unusual objective because filling the output space does not guarantee a precise estimation of the input–output relationship. On the other hand, the benefits of filling the input spaces are well known. Johnson, Moore, and Ylvisaker (1990) showed that a maximin distance design in the input space would be asymptotically D-optimal for fitting a Gaussian process model and a minimax design would be asymptotically G-optimal (as the correlations tend to zero). However, as noted by Lu and Anderson-Cook (2021), there are several applications in which filling the output space would be beneficial.

In inverse design problems, the aim is to find input configurations that will achieve a specified set of outputs. Consider, for the example, the design of acoustic metasurfaces to achieve given acoustic properties such as the amplitude and phase of transmitted and reflected waves (Krishna et al. 2022). The aim is to create several acoustic metasurfaces offline using 3D printing, store them, and pick the best one for a given set of acoustic properties. In other words, the aim is to create a "lookup-table" of acoustic metasurface geometries and acoustic properties, where the investigator can quickly identify the geometry based on the set of acoustic properties. This approach will be successful if the acoustic properties in the lookup table is dense, that is, the space of acoustic properties should not have large gaps. Thus, the aim here is to identify the set of geometries that will fill the output space of acoustic properties.

As a second application, consider statistical and machine learning problems involving feature-based modeling. The first step in such problems is to extract "features" from the input space. The modeling is then done between the response and features. In this scenario, the feature-output relationship can be accurately estimated if the points in the feature space is space-filling. Thus, the aim is to identify a set of points in the input space so that the points in the feature space are space-filling. As a real example, consider the crystal structure prediction problem described in Krishna et al. (2023). The input space is the Cartesian coordinates of the atomic configurations of a single crystal structure, and the output is the potential energy computed using Density Functional Theory (DFT). However, since the potential energy is invariant to translational, rotational, and permutational operations of the atoms, the Cartesian coordinate system is not suitable for model building. Therefore, the Cartesian coordinates are converted to a set of features using

AGNI (Adaptive, Generalizable and Neighborhood Informed) fingerprinting (Batra et al. 2019). AGNI fingerprinting is fast compared to DFT computations, but has nonnegligible cost which makes developing space-filling points in the feature space a difficult task.

It is much more challenging to generate space-filling points in the output (response, feature, etc.) space as compared to the input space. ISFD is generated in a known experimental region, usually a hypercube, whereas OSFD aims to fill an unknown region with unknown boundaries. A naive approach of creating a large set of candidate points in the output space to choose a set of space-filling points would not be feasible in either of the two scenarios: (a) high cost of evaluating the input–output function and (b) the region in the input space to cover the output space is small relative to the whole input space. This suggests that a sequential design is the only viable option, as we can learn the "active regions" in the input space gradually, and fill-in the output space with as few function evaluations as possible. Developing such a sequential design is the main aim of this article.

Sequential designs, also known as active learning, is widely used in statistics and machine learning for dealing with expensive black-box functions. Most of these are "model-based" designs and use Gaussian process modeling extensively (Gramacy 2020, chap. 6). However, Gaussian process modeling has a high cost for training, which can be appreciable in several applications. Therefore, we need new sequential design methods that is fast and efficient to fill-in the output spaces.

The article is outlined as follows. Section 2 begins by introducing some notations used in this article and then reviews the traditional ISFDs and related works on OSFDs. Section 3 presents the definition of minimax output space-filling design and proposes efficient algorithms to generate such designs. Section 4 demonstrates the performance of the proposed algorithms using three simulation studies. Section 5 illustrates the application of OSFD on inverse design and feature-based modeling. Section 6 concludes the article with some final remarks.

2. Background

Denote the input space by $\mathcal{X} \subseteq \mathbb{R}^p$, output space by $\mathcal{Y} \subseteq \mathbb{R}^q$, and let the mapping from input space to output space be $f : \mathcal{X} \to \mathcal{Y}$. Typically f is a black-box computer code that is expensive to evaluate. Denote a design of size $n \in \mathbb{N}$ by $\mathcal{D}_n = \{x_i \in \mathcal{X}, i = 1, \dots, n\}$. The corresponding points in the output space is denoted by $\mathcal{M}_n = f(\mathcal{D}_n) = \{y_i : y_i = f(x_i), x_i \in \mathcal{D}_n, i = 1, \dots, n\}$. Our goal is to find a design $\mathcal{D}_n$ such that $\mathcal{M}_n$ is space-filling in $\mathcal{Y}$. Before we formally define what is "space-filling" in $\mathcal{Y}$, we first review the traditional input space-filling design (ISFD).

2.1. Space-Filling Design

From a geometric point of view, there are two commonly used space-filling design schemes: maximin distance design and minimax distance design (Johnson, Moore, and Ylvisaker 1990). Let d_x be a metric on $\mathbb{R}^p$. Then the maximin distance design maximizes the following criterion:

$$\phi_{\mathrm{Mm}}(\mathcal{D}_n) = \min_{x_i, x_j \in \mathcal{D}_n; i \neq j} d_x(x_i, x_j). \qquad (1)$$

That is, it places design points such that the minimum distance between any two points is as large as possible. Minimax distance design, on the other hand, tries to minimize the maximum distance from all the points $x \in \mathcal{X}$ to their closest neighbor in $\mathcal{D}_n$, which is obtained by minimizing the following criterion:

$$\phi_{\mathrm{mM}}(\mathcal{D}_n) = \max_{x \in \mathcal{X}} \min_{x_i \in \mathcal{D}_n} d_x(x, x_i). \qquad (2)$$

This criterion is also known as the fill distance (Fasshauer 2007), which will be used throughout this article. Maximin and minimax distance designs may not have good projection properties and therefore, they are combined with Latin hypercube designs (LHD) to improve their one-dimensional projections (Morris and Mitchell 1995). We refer the readers to Joseph (2016) for a detailed review of the vast literature on ISFD. These designs allow for a careful exploration of the experimental region by making sure that no part of the input space is left out. This property makes these designs model-free and therefore, they enable the experimenter to fit a wide variety of statistical and machine learning models to the data and make predictions.

2.2. Related Works

The literature on output space-filling design (OSFD) is scarce. Rhee, Zhou, and Qiu (2017) seems to be the first work that discussed about space-filling designs for output spaces. However, their goal is more closely aligned with uniform sampling on a manifold rather than generating an experimental design. To generate n uniform points in the output space, they start with n random samples in the input space and then improve them through weighting and resampling. However, in the context of design of experiments, augmentation of the design points makes more sense than resampling. In this article, we will develop a sequential design strategy that adds points one-at-a-time to the existing set of points, thereby obtaining a space-filling design with minimum number of function evaluations. Nonuniformity of the points in the output space is not a concern at all for us as long as the points can fill-in the output space.

Lu and Anderson-Cook (2021) recently proposed a design strategy that simultaneously achieve space-fillingness in both input and output spaces using Pareto front optimization. They assume that the input–output relationship is known and cheap to evaluate, which is quite different from the problem we tackle in this article. Lu and Anderson-Cook (2021) also proposed a two-stage approach to deal with the unknown input–output relationship by first using an ISFD to estimate the relationship and then using the estimated model to perform the Pareto front optimization. Their second stage design can suffer if the estimated model is wrong from the first stage. In contrast, we develop a fully sequential model-robust design strategy to construct the OSFD.

Our work is motivated by the two applications briefly discussed in Section 1: inverse design of acoustic metasurfaces (Krishna et al. 2022) and crystal structure prediction (Krishna et al. 2023). The authors develop design strategies specific to those two applications. In contrast, the design strategy developed here is more general, efficient, and broadly applicable.
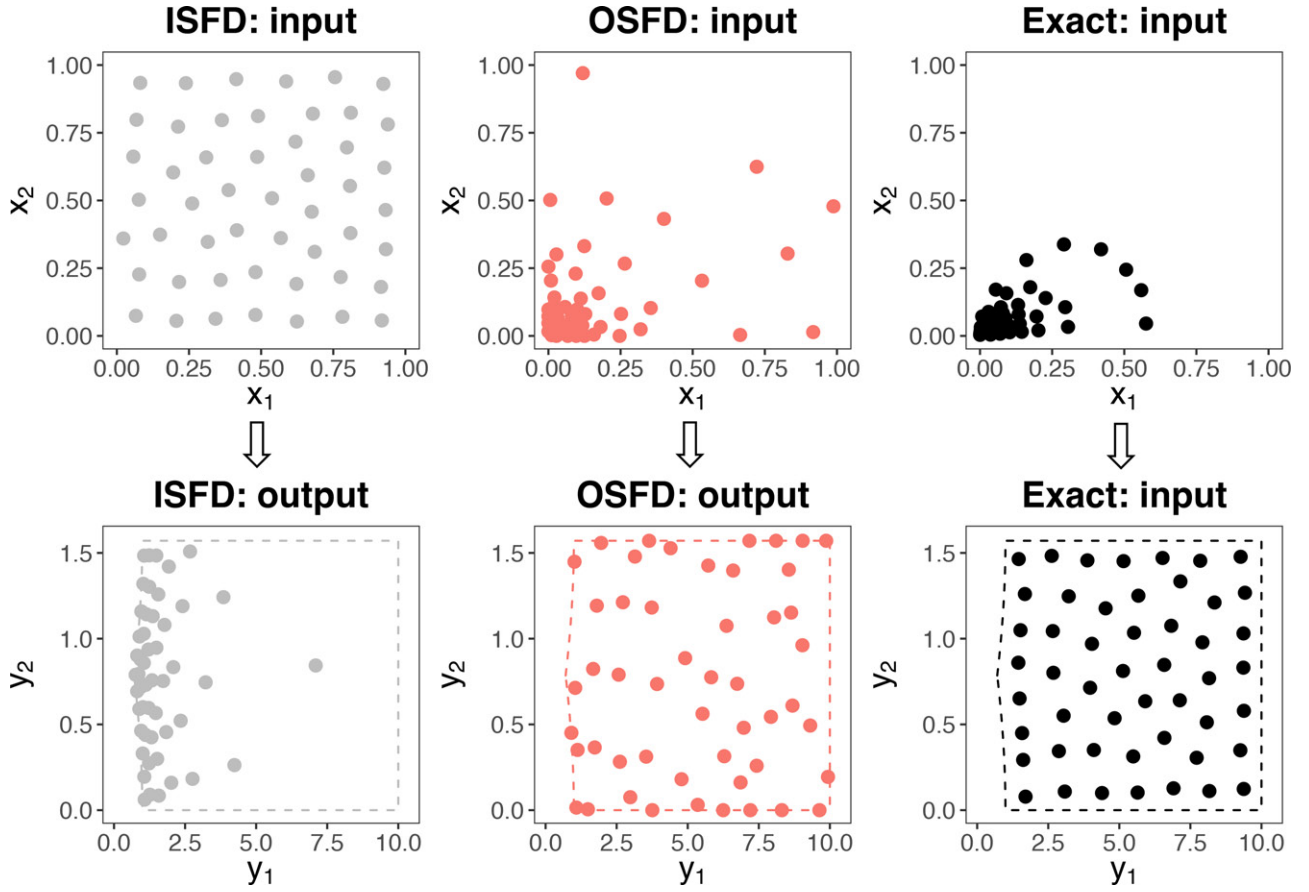
Figure 1. Design points and outputs for the inverse radius function in (4) with $\epsilon = 0.1$. Input space-filling design (left); output space-filling design by OSFD-greedy algorithm (middle); and the exact solution of the minimax OSFD (right). Design size is 50. The output space is enclosed by the dashed line. The initial design for the OSFD-greedy algorithm is a random LHD of size 5.

3. Output Space-Filling Design

3.1. Mathematical Formulation

Similar to the traditional minimax distance design, here we quantify the space-fillingness of the design output using the minimax distance. Therefore, our objective is to minimize

$$\phi_{\mathrm{mM}}(\mathcal{M}_n) = \max_{\boldsymbol{y} \in \mathcal{Y}} \inf_{\boldsymbol{y}_i \in f(\mathcal{D}_n)} d_y(\boldsymbol{y}, \boldsymbol{y}_i), \qquad (3)$$

with respect to $\mathcal{D}_n$, where d_y is a metric defined on $\mathbb{R}^q$. We use Euclidean distance as the metric unless otherwise mentioned. We call the minimizer of the foregoing objective function a minimax output space-filling design. The only difference between (2) and (3) is the domain to which we apply the minimax distance criterion. However, due to the unknown mapping f and unknown output space $\mathcal{Y}$ in (3), obtaining a high-quality space-filling design in the output space is almost impossible and the only hope is to develop numerical algorithms that can at least approximate this idealized aim as close as possible.

For illustration, consider a simple "inverse-radius" function f_{ir} that maps $[0, 1]^2$ to a subset in $\mathbb{R}^2$:

$$f_{\mathrm{ir}}(x_1, x_2) = \left(\frac{1}{\sqrt{x_1^2 + x_2^2 + \epsilon^2}}, \arctan \frac{x_2}{x_1} \right). \qquad (4)$$

This function has large gradient near the origin and is flat elsewhere, with ϵ controlling the variation. As shown in the left

panels in Figure 1, if we use the traditional minimax design, the output points will congregate at the left side of the output space and leave most of $\mathcal{Y}$ unexplored. Since f_{ir} is a bijection, to get the theoretical minimax output space-filling design, we can first construct a minimax design in output space, which we denote as $\mathcal{M}_n^* = \arg\min_{\mathcal{M}_n} \phi_{\mathrm{mM}}(\mathcal{M}_n)$, and then map it back to the input space to obtain $\mathcal{D}_n^* = f_{\mathrm{ir}}^{-1}(\mathcal{M}_n^*)$. Because the output space is irregularly shaped, here we use the minimax clustering with particle swarm optimization (mM-PSO) algorithm of Mak and Joseph (2018) to generate the minimax points in the region enclosed by the dashed line on the right panel of Figure 1. In this case, the output space is covered uniformly and the design points in the input space exploits the "interesting" region where the function f_{ir} has large variation. In practice, though, we have no knowledge of the output space and f_{ir} is a black-box function that can only be evaluated in the forward direction. Therefore, this exact optimal design is impossible to attain in practice. Interestingly, our sequential output space-filling design algorithm discussed in the next section can find a compromise between the two aforementioned cases, recovering most part of the output space while exploring the input space well (see the middle panels of Figure 1).

3.2. A Sequential Design Algorithm

Our algorithm consists of two steps: (i) find the largest gap in the output space and (ii) perturb the corresponding input point

to generate a new design point in the input space. These two steps will be continued until the largest gap in the output space is below a specified threshold or when the budget is run out. We will now describe the two steps in detail.

3.2.1. Gap Identification

Suppose we already have m points in the input and output spaces: $(\mathcal{D}_m, \mathcal{M}_m)$. The first step is to identify the largest gap in the output space. For this purpose, we define the local fill distance around each point y_i as

$$h_i = \max_{y \in V_i^{\text{out}}} d_y(y, y_i), \tag{5}$$

where V_i^{out} is the Voronoi region around y_i given by

$$V_i^{\text{out}} = \{y \in \mathcal{Y}, d_y(y, y_i) \le d_y(y, y_j), \ \forall y_j \in \mathcal{M}_m \neq y_i\}, \tag{6}$$

for $i = 1, \ldots, m$. Note that h_i and V_i^{out} depend on the current outputs $\mathcal{M}_m$. For notational compactness, we have dropped their dependence on $\mathcal{M}_m$ as long as it is clear that these quantities will change as more points are added to the design. It is easy to see that the fill distance of $\mathcal{M}_m$ is

$$\phi_{\text{mM}}(\mathcal{M}_m) = \max_{i=1:m} h_i.$$

The index of the point corresponding to the largest gap in the output space is given by

$$i^* = \arg\max_{i=1:m} h_i. \tag{7}$$

With the point of largest local fill distance identified, we can perturb x_{i^*} in the input space. However, the evaluation of local fill distance requires the knowledge of the true $\mathcal{Y}$, which is actually unknown beforehand. This renders a direct segmentation of the output space into Voronoi regions (6) not feasible. Therefore, based on $\mathcal{M}_m$, we first generate a set of points $\mathcal{A}$ to approximate $\mathcal{Y}$ as follows.

The approximating point set is comprised of three parts. The first part $\mathcal{A}_1$ is generated by constructing a $(p \wedge q + 1)$-dimensional simplex by connecting each $y_i \in \mathcal{M}_m$ and its $p \wedge q$ nearest neighbors $N^{p \wedge q}(y_i) = \{y_i^{(l)} \in \mathcal{M}_m : y_i^{(l)}$ is the lth nearest neighbor of $y_i, l = 1, \ldots, p \wedge q\}$ and then finding the centroid:

$$c = \frac{1}{p \wedge q + 1} \sum_{l=0}^{p \wedge q} y_i^{(l)}, \tag{8}$$

where $p \wedge q = \min(p, q)$ and $y^{(0)} = y_i$. We also add axial points so that we can go outside of the convex hull of $\mathcal{M}_m$:

$$c_j = \left(\frac{1.5}{p \wedge q} \sum_{\substack{l \neq j; \\ 0 \le l \le (p \wedge q)}} y_i^{(l)} \right) - 0.5 y_i^{(j)}, \tag{9}$$

where $j = 0, 1, \ldots, p \wedge q$. Note that if the output dimension q is larger than the input dimension p, the output space would be a p dimensional manifold in a q dimensional space. Therefore, it is more natural to consider a simplex of the lower dimension. Implicit in this argument is the assumption that the

input variables are all active, otherwise, the manifold dimension can be even lower. The second part $\mathcal{A}_2$ is generated by finding the midpoints between each $y_i \in \mathcal{M}_m$ and its k_1-nearest neighbors $N^{k_1}(y_i)$. These points reside on the (extended) boundaries of the Voronoi cells and have equal distances to the endpoints. We choose $k_1 = 2(p \wedge q)$ by default. The last part $\mathcal{A}_3$ consists of points in $p \wedge q$-dimensional balls around each design output. The rational to use $p \wedge q$-dimensional balls is similar: if $q > p$, we should not generate the approximating points by q-dimensional balls since most of the points would fall outside the manifold. Instead we extract the tangent spaces around each design outputs using the simplexes constructed in the first part and generate uniform points in p-dimensional balls on the tangent spaces. This procedure is presented as Algorithm 1 in the appendix and illustrated by an example in Figure 2. We can see that for each output point y_i, $d_i = \max_{a \in V_i^{\text{out}} \cap \mathcal{A}} d_y(a, y_i)$ is a reasonable approximation of the exact local fill distance $h_i = \max_{a \in V_i^{\text{out}}} d_y(a, y_i)$.

3.2.2. Perturbation

As mentioned earlier, we will perturb the input design point x_{i^*}, corresponding to y_{i^*}, the point in the output space with the largest local fill distance. The only thing we need to decide is how to perturb the x_{i^*}. Ideally, we would like to find a new input point in such a way that the output will minimize the largest gap. However, this cannot be done optimally because $f(\cdot)$ is expensive to evaluate. Therefore, we propose to perturb x_{i^*} to its maximum permissible level, which will also promote the space-fillingness in the input space. Thus, we choose the next point as the furthest point in the Voronoi cell $V_{i^*}^{\text{in}}$ of x_{i^*} as shown in Figure 3, where

$$V_i^{\text{in}} = \{x \in \mathcal{X}, d_x(x, x_i) \le d_x(x, x_j), \ j \neq i\}.$$

The foregoing computation is done as follows. A candidate set $\mathcal{C}$ is first built by uniform samples in a hypercube around x_{i^*}. We then find the k_2-nearest neighbors $N^{k_2}(x_{i^*})$ of x_{i^*}, where we choose $k_2 = 2p$ by default so that there would be two neighbors on each dimension on average if the $\{x_i\}_{i=1}^m$ were uniformly distributed. The set $\mathcal{C}$ is then augmented with uniform points in balls centered at x_{i^*} and its neighbors $N^{k_2}(x_{i^*})$. The next design point x_{m+1} is chosen as

$$x_{m+1} \in \arg\max_{x \in V_{i^*}^{in} \cap \mathcal{C}} d_x(x, x_{i^*}). \tag{10}$$

This step and its parameter specification are detailed in Algorithm 3 in the appendix. The whole output space-filling design algorithm is summarized in Algorithm 5 in the appendix. An illustration of the OSFD algorithm on the inverse radius function is shown in the middle panels of Figure 1. We can see that it does a good job filling the output space.

Now consider the following challenging exponential function $f_{\text{exp}}^{\alpha} : [0, 1]^2 \to \mathbb{R}^3$ taken from Rhee, Zhou, and Qiu (2017):

$$f_{\text{exp}}^{\alpha}(x_1, x_2) = \left(e^{-\alpha x_1} + e^{-\alpha x_2}, e^{-2\alpha x_1} + e^{-2\alpha x_2}, e^{-4\alpha x_1} + e^{-4\alpha x_2} \right). \tag{11}$$

The parameter α controls the gradient of f_{exp}^{α} in the input space $[0, 1]^2$. When α is large, the gradient is large only in a very small area near the origin, making the active region difficult to locate. Consider $\alpha = 10$ and $\alpha = 100$. We can see from
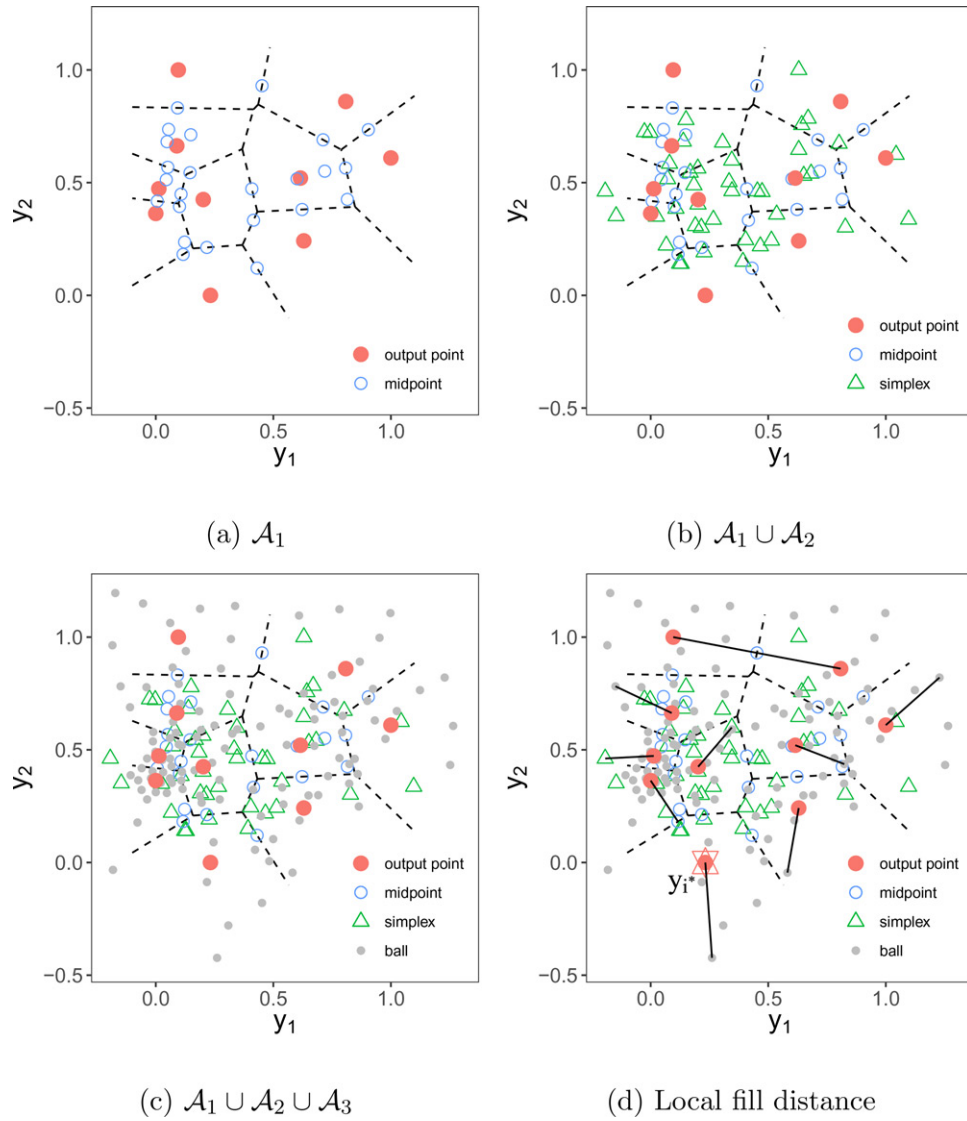
Figure 2. Construction of approximating points $\mathcal{A}$: Red points are the scaled output points $\mathcal{M}_{10}$. Dashed lines divide the output region into 10 Voronoi cells, within which each output point is connected to its furthest approximating point. The star in the last figure is the design output y_{i*} with largest local fill distance.

Figure 4 that the proposed algorithm performs well for $\alpha = 10$ but fails to cover the output space for $\alpha = 100$. When $\alpha = 100$, this function has large variation only in the region around $[0, 0.04]^2$ (Figure 4: right panel), which means 99.84% of the input design space will give almost identical responses. It is therefore almost impossible to find such a small area in the initial design. Moreover, at the initial stage, the approximation of the output space is inaccurate because the existing design points are far from the region $[0, 0.04]^2$. Thus, the algorithm can get stuck in a local region. In the next section, we propose an improved algorithm that helps to jump out of the local regions.

3.3. An Improved Algorithm

Bayesian optimization (Garnett 2023) is a popular technique for the global optimization of expensive black-box functions. The key idea in Bayesian optimization is to introduce an acquisition function that includes not only the function value but also its uncertainty estimate. Expected improvement (EI) criterion (Jones, Schonlau, and Welch 1998) is one such acquisition func-

tion. The EI criterion encourages the design points to explore the experimental region while exploiting the function, which aids in jumping out of local regions and enable the design points to move toward the global optimum. The EI algorithm uses Gaussian process (GP) modeling, which automatically gives the uncertainty estimates alongside predictions. However, as mentioned in the introduction, the high training cost of GP models can become a computational bottleneck.

It is well-known that a minimax design is based on a nearest neighbor predictor (Joseph 2006). A nearest neighbor predictor is extremely fast. Its estimation can be done in $O(pm \log m)$ operations and prediction on N points in $O(N \log m)$ operations, which are much smaller than the $O(m^3 p)$ and $O(N^2)$ operations needed for a GP model. Unfortunately, nearest neighbor predictor is not based on a stochastic model and therefore, it does not come with uncertainty estimates as in GP modeling. Thus, the nearest neighbor approach is possible only if we can develop an uncertainty estimate.

Let $h(\boldsymbol{x})$ denote the local fill distance in the output space at an input value $\boldsymbol{x}$. Thus, given the data $\boldsymbol{h} = (h_1, \ldots, h_m)'$, for a
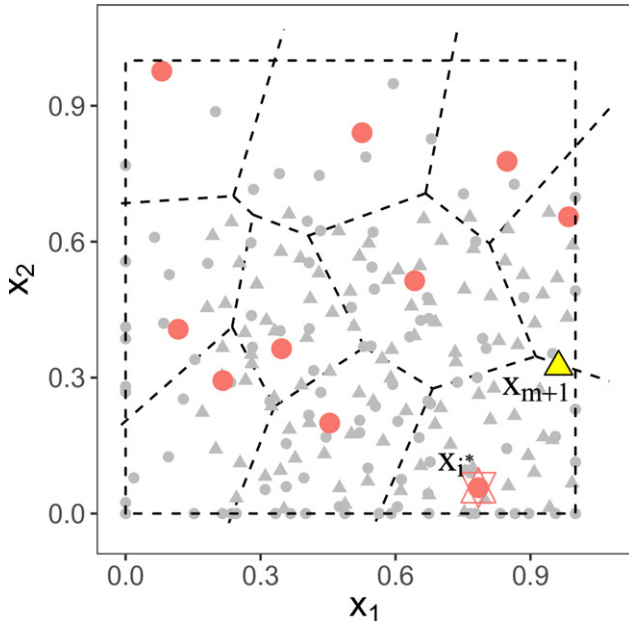
Figure 3. Illustration of the perturbation step. Red points are design points in the input space. The red star is the design points x_{i*} corresponding to the output point y_{i*} with the largest local fill distance. The yellow triangle is the next design point x_{m+1} chosen from the candidate set. Uniform candidates are denoted as the gray points and those candidates generated in the balls around x_{i*} and its neighbor are denoted as gray triangles.

nearest neighbor predictor

$$\mathbb{E}\{h(\boldsymbol{x})|\boldsymbol{h}\} = h_i \text{ for } \boldsymbol{x} \in V_i^{\text{in}}(\boldsymbol{x}).$$

Motivated by the Brownian random fields (Zhang and Apley 2014), we postulate a variance for the nearest neighbor predictor to be

$$\mathbb{V}\{h(\boldsymbol{x})|\boldsymbol{h}\} = \sigma^2 ||\boldsymbol{x} - \boldsymbol{x}_i|| \text{ for } \boldsymbol{x} \in V_i^{\text{in}}(\boldsymbol{x}).$$

This has the desirable property that $\mathbb{V}\{h(\boldsymbol{x}_i)\} = 0$ for $i = 1, \ldots, m$ and that the variance increases as the prediction point moves away from the design points, just like the posterior variance in a Gaussian process model. Assuming normality, we have

$$h(\boldsymbol{x})|\boldsymbol{h} \sim \mathcal{N}\left(h_i, \sigma^2||\boldsymbol{x} - \boldsymbol{x}_i||\right) \text{ for } \boldsymbol{x} \in V_i^{\text{in}}(\boldsymbol{x}). \quad (12)$$

Thus, the expected improvement acquisition function can be obtained as

$$\begin{aligned} \text{EI}(\boldsymbol{x}; \mathcal{D}_m) &= \mathbb{E}\left[\max\left(0, h(\boldsymbol{x}) - h_{\max}\right)|\boldsymbol{h}\right] \\ &= s(\boldsymbol{x})\{u(\boldsymbol{x})\Phi((u(\boldsymbol{x})) + \phi(u(\boldsymbol{x}))\} \text{ for } \boldsymbol{x} \in V_i^{in}(\boldsymbol{x}), \end{aligned} \quad (13)$$

where $h_{\max} = \max_i h_i$, $s(\boldsymbol{x}) = \sqrt{\sigma^2||\boldsymbol{x} - \boldsymbol{x}_i||}$, $u(\boldsymbol{x}) = (h(\boldsymbol{x}) - h_{\max})/s(\boldsymbol{x})$, $\phi(\cdot)$ is the density function of the standard normal random variable and $\Phi(\cdot)$ is its cumulative distribution function. An estimate of σ^2 can be obtained by maximizing the leave-one-out cross-validation likelihood as (Geisser and Eddy 1979):

$$\widehat{\sigma}^2 = \frac{1}{m} \sum_{i=1}^{m} \frac{\left(h_i - h(\boldsymbol{x}_i^{(1)})\right)^2}{||\boldsymbol{x}_i - \boldsymbol{x}_i^{(1)}||}, \quad (14)$$

where $\boldsymbol{x}_i^{(1)} = \arg\min_{\boldsymbol{x} \in \mathcal{D}_m \setminus \{\boldsymbol{x}_i\}} ||\boldsymbol{x}_i - \boldsymbol{x}||$.

As before, we generate a candidate set of points $\mathcal{C}$ and obtain the next design point as

$$\boldsymbol{x}_{m+1} = \arg\max_{\boldsymbol{x} \in \mathcal{C}} \text{EI}(\boldsymbol{x}; \mathcal{D}_m). \quad (15)$$

To promote exploring across the whole input space, the candidate set $\mathcal{C}$ includes uniform random points in the hypercube $[0, 1]^p$ instead of a local region around $\boldsymbol{x}_{i*}$. The new algorithm is shown in Algorithm 4 in the appendix. To distinguish this design from the previous greedy strategy, we will refer to the new design as OSFD-EI and the previous design as OSFD-greedy.

The 300 points generated using the OSFD-EI algorithm for the exponential function with $\alpha = 100$ is shown in Figure 5. The initial design is random LHD of size 30. We can see that the new design tends to explore the input space like an ISFD and is able to jump out of the local regions and fill the output space reasonably well. On the other hand, the approach in Rhee, Zhou, and Qiu (2017) required thousands of function evaluations to get a similar result.

We also tried a GP model instead of the nearest neighbor (NN) predictor on the exponential function with $\alpha = 100$. The GP model was fitted using the R package `DiceKriging` (Roustant, Ginsbourger, and Deville 2012) at the default settings. The fill distance (left) and computational time (right) averaged over 10 replications are shown in Figure 6 for various values of n. As expected, the NN predictor makes the algorithm run much faster than with GP. The fill distance is comparable to that of the GP model.

4. Simulations

In this section, we investigate the performance of the two proposed OSFD algorithms. We consider three test functions: the inverse-radius function $\boldsymbol{f}_{\text{ir}}$ (4), the exponential function f_{exp}^{α} (11) with different α, and a modified Easom function (Solteiro Pires et al. 2010):

$$f_{\text{esm}}^{p}(\boldsymbol{x}) = \prod_{i=1}^{p} \cos(2\pi x_i) \exp\left(-\frac{\pi^2(2x_i - 1)^2}{p}\right) \quad (16)$$

with different input dimension p. We will use the fill distance in the output space to quantify the performance. A two-phase algorithm adapted from Lu and Anderson-Cook (2021) is also included for comparison. In the first phase of this algorithm, we use a random LHD of size $n/4$ (rounded to the nearest integer) to build a multivariate GP model. In the second phase, we predict the responses of $100n$ uniform random inputs using this GP model and choose $3n/4$ predicted responses based on maximin criterion. The corresponding input points are chosen as the remaining $3n/4$ points of the n-point design. Clearly, the performance of this algorithm depends largely on how well the GP model fits the underlying mapping $\boldsymbol{f}$ in the first phase. We used `DiceKriging` (Roustant, Ginsbourger, and Deville 2012) for fitting the GP model and the R package `maximin` (Sun and Gramacy 2021) for finding the maximin points.

Figure 7 shows the simulation results for the inverse-radius function. We initialize the OSFD by a random LHD with sample size $n_0 = 10$ and replicate the simulation 20 times. The solid line indicates the mean value of the fill distance and the shaded
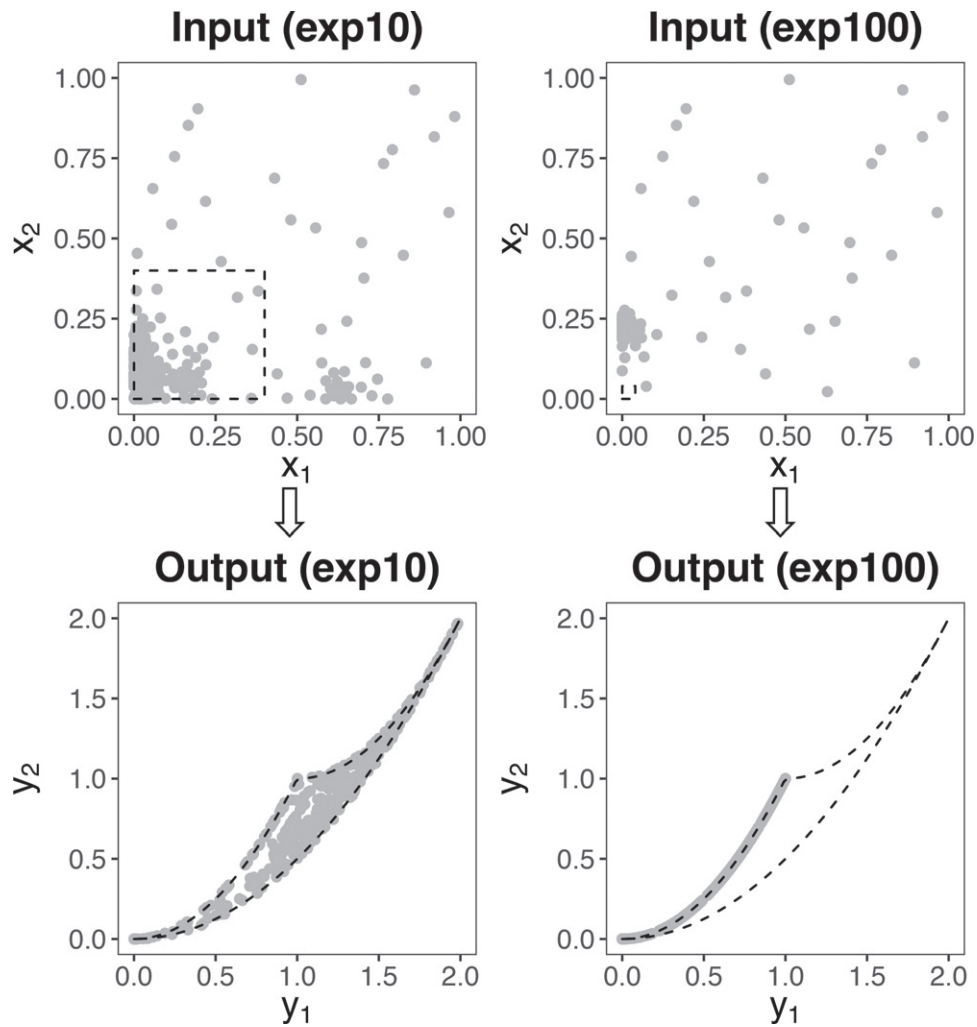
Figure 4. Top row: 300 design points generated by the OSFD-greedy algorithm for the exponential function in (11) with $\alpha = 10$ (left) and $\alpha = 100$ (right). Bottom row: the corresponding output points projected to the first two coordinates. The intial design is generated by random LHD of size 30. The active region in the input space is shown as a box using dashed lines. The true output space is also shown using dashed lines.
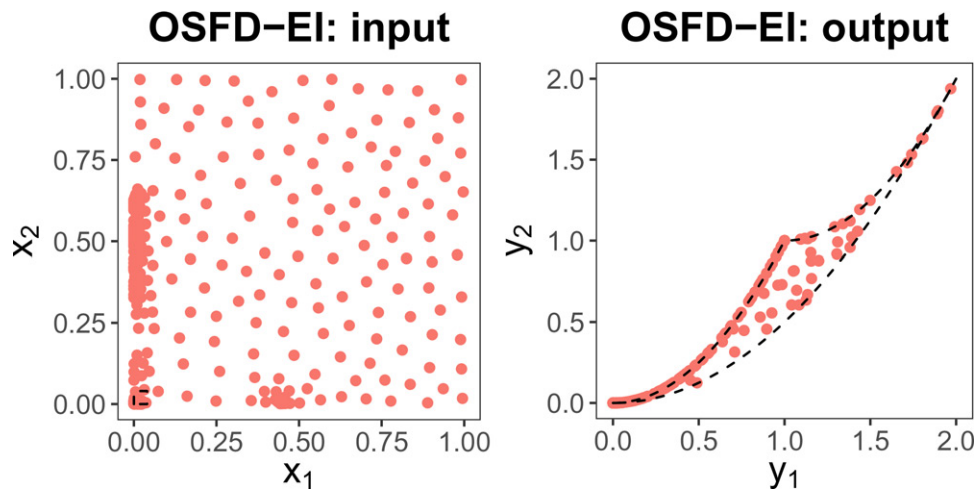

**Figure 5.** Design points and outputs for f_{exp}^{100} generated by OSFD-EI.

band represent the 90% confidence intervals. For each n, we also generate an ISFD using random LHD and compute its fill distance in the output space. We can see that both OSFD-greedy and OSFD-EI algorithms outperform the ISFD and the two-

phase algorithm. In fact, both the proposed algorithms quickly attain the optimal fill distance after a few steps. The optimal value is obtained by directly running the mMc-PSO algorithm using the R package `minimaxdesign` (Mak 2016) on the true
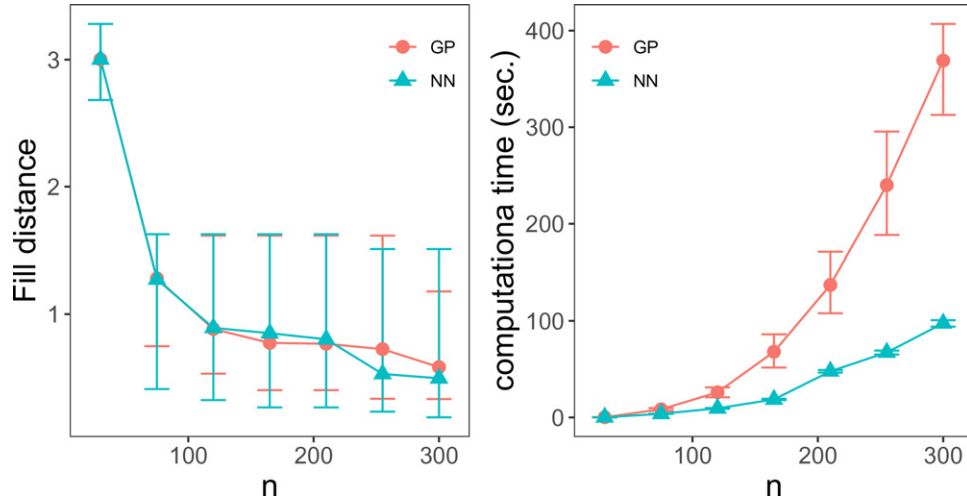
Figure 6. The fill distance and computational time for an OSFD of size *n* generated by nearest neighbor-based OSFD-EI and GP-based OSFD-EI. The underlying function is f_{exp}^{100}. The lines denote the average over 10 replications and the error bars represent the 5th and 95th quantiles.
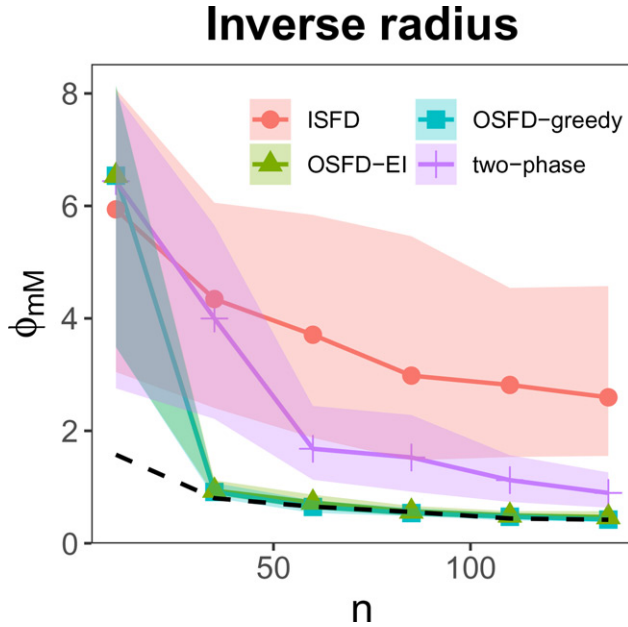
Figure 7. Fill distance for OSFD-EI, OSFD-greedy, ISFD, two-phase, and optimal minimax OSFD (black dashed line) against run size for the inverse radius function in (4). Lines denote the average values and the shaded bands mark the 5th and 95th quantiles.

output space. Another advantage of our algorithms compared to the two-phase algorithm is its speed. To generate a design of size 150, the computational time for OSFD-EI and OSFD-greedy are 1.1 sec and 0.7 sec, respectively, while it takes 6.8 sec for the two-phase algorithm in a 2.6 GHz 6-Core Intel Core i7 processor. This computational saving becomes even more substantial as the design size increases.

The foregoing simulation is repeated on the exponential function in (11) with $\alpha = 10, 40, 100$. We initialize the OSFD algorithms using $n_0 = 50$ random LHD points in the input space. We can see from Figure 8 that both the OSFD algorithms are superior to ISFD (using random LHD) when $\alpha = 10$. As the α increases, the problem becomes more challenging. We can see that the OSFD-EI performs much better than the

OSFD-greedy and the two-phase algorithm for large α. We have also tested another version of the OSFD algorithm that uses the Upper Confidence Bound (UCB) criterion (Garnett 2023). The results are shown in the supplementary materials. We can see that OSFD-UCB performs almost as good as the OSFD-EI. This shows that the normality assumption used in (12) for deriving the EI criterion is not critical and that one can use any other criterion that encourages exploration to improve upon the OSFD-greedy algorithm.

Finally, we investigate the impact of input dimension on the performance by considering the modified Easom function in (16). Figure 9 shows that OSFD consistently outperforms ISFD (using random LHD) and the two-phase algorithm, however, the advantage of OSFD diminishes as the dimension increases. This is not unexpected as high dimension makes any feasible set of design points sparse in the input space. Improving the performance of OSFD for high dimensional problems could be an important topic for future research.

5. Applications

In this section, we will present two applications of output space-filling design.

5.1. Inverse Design

In the inverse design problem, the goal is to provide a suitable input that can produce a desired output within a reasonable degree of accuracy. This output can be crystal properties (Ren et al. 2022), modulating properties of optical devices (Molesky et al. 2018), or the acoustic properties of material structures (Krishna et al. 2022), etc. If a large number of targets are of interest, OSFD design can quickly give the set of inputs that can approximately achieve the targets.

Here we consider a simple example in which we hope to control a robot arm in a two-dimensional plane (An and Owen 2001). The robot arm has four extendable segments of lengths L_1, L_2, L_3, L_4 and are at angle $\theta_1, \theta_2, \theta_3, \theta_4$ to the horizontal
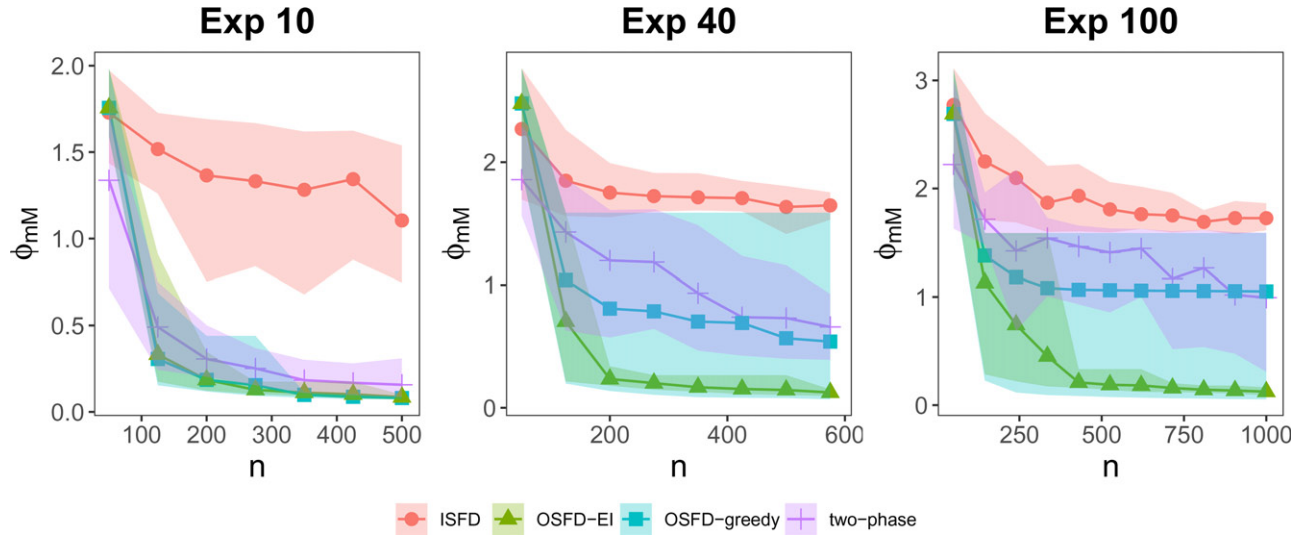
Figure 8. Fill distance for OSFD-EI, OSFD-greedy, two-phase, and random LHD against run size for the exponential function in (11) with $\alpha = 10, 40, 100$. Lines denote the average values and the shaded bands mark the 5th and 95th quantiles.
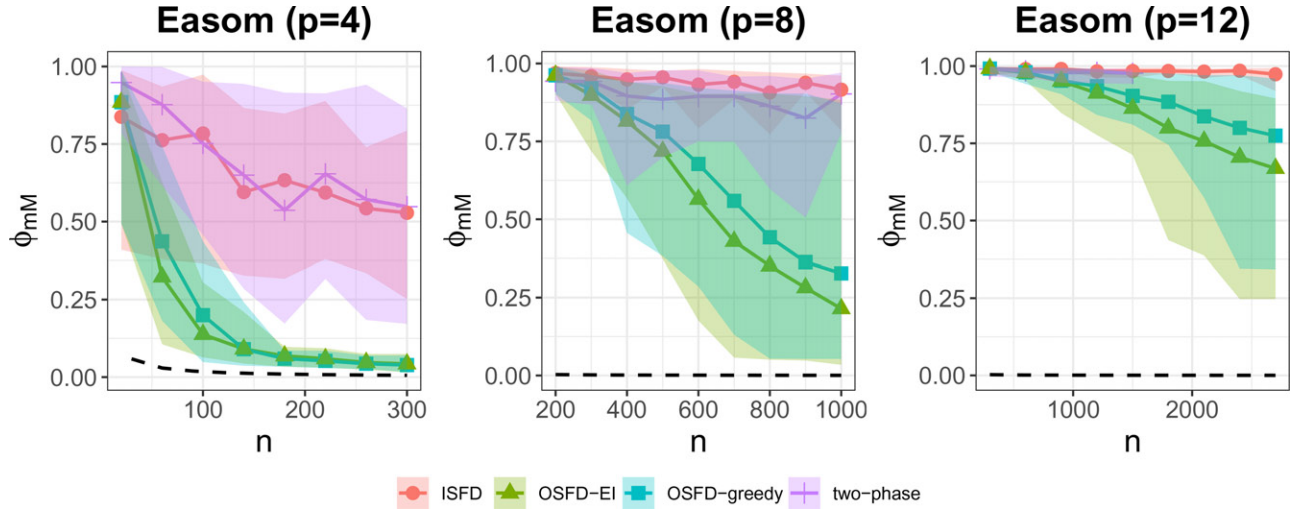
Figure 9. Fill distance for OSFD-EI, OSFD-greedy, random LHD, two-phase, and optimal minimax OSFD (black dashed line) against run size for the modified Easom function with $p = 4, 8, 12$ with initial design size of 20, 200, 300. Lines denote the average values and the shaded bands mark the 5th and 95th quantiles. The two-phase algorithm for $p = 12$ is terminated at $n = 1500$ due to high computational time and memory requirement.

coordinate axis of the plane. The location of the end of the robot arm (u, v) is

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

where $L_i \in [0, 1]$ and $\theta_i \in [0, 2\pi]$ for $i = 1, 2, 3, 4$. Clearly, the arm's range of motion is within a circle of radius 4.

Figure 10(a) shows that the outputs from the OSFD-greedy has a considerably better coverage of the full output space than the outputs generated by an ISFD (generated using the maximin LHD). To quantify the approximation error in an inverse design application, we generate 100,030 uniform points inside the circle shown in Figure 10(a) (left). The distances from each target y^* to the closest output point $\Delta = d_y(y^*, N^1(y^*))$ should be as small as possible. The nearest output point $N^1(y^*)$ is found from the outputs of OSFD and ISFD, respectively. From Figure 10(b), we can see that the distances from the target outputs produced

from OSFD are in general significantly lower than those from the ISFD.

5.2. Feature-Based Modeling

In this section, we present an exemplar case study of feature-based modeling within the field of material informatics based on Generale and Kalidindi (2021) in which a model-based linkage between virtually generated 5-harness satin (5HS) ceramic matrix composite (CMC) microstructures and their effective orthotropic thermal conductivity was developed. A GP-based model was trained through an active learning framework to reduce the computational burden inherent in performing finite element (FE) based thermal analyses. Subsequent microstructures selected for evaluation were identified through maximum posterior uncertainty and constrained to a preexisting microstructure ensemble. The input features for this model were extracted from n-point spatial correlations of the
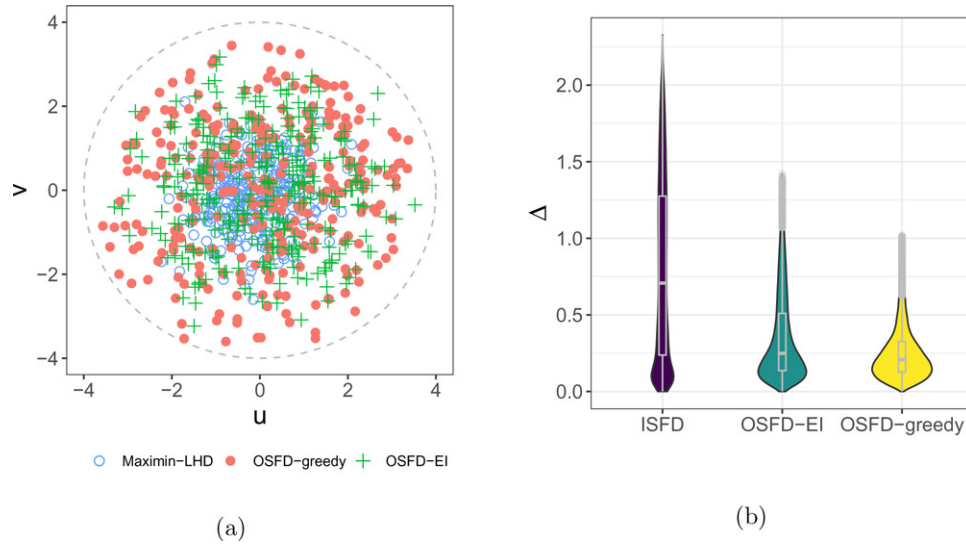
Figure 10. Robort arm application: (a) Scatterplot of the outputs from OSFD and Maximin-LHD; (b) The distances from the targets to the closest output point (Δ). Total number of targets is 100,030. Design size is 300. Initial design is generated by maximin LHD of size 30.

Table 1. Geometric parameters for CMC.

Microstructural dimension	Minimum	Nominal	Maximum
Tow major axis - t_w	788	985	1182
Tow minor axis - t_h	92	115	138
Tow spacing - t_s	880	1100	1320
Ply spacing - p_c	257	275	292
Matrix thickness - m_t	47	78	109

microstructure (also referred to as n-point statistics) (Torquato 2002). As a collective, n-point spatial correlations provide a hierarchy of increasingly complex descriptions of the material microstructure. One-point spatial correlations capture the probabilities of finding a specific local material state (i.e., microscale constituent) at any randomly selected voxel in a discretized representative volume element (RVE), more commonly referred to as the local material state's volume fraction. With increasing complexity, two-point spatial correlations define the probability of finding two specified local states at the head and tail of a randomly placed vector in the RVE. From solely the description of the first two n-point spatial correlations, it quickly becomes apparent that this representation of a materials microstructure is extremely high-dimensional in nature, with each dimension capturing a singular statistic related to the spatial arrangement of local states (Kalidindi 2015). In this case study, two-point spatial correlations were computed for an ensemble of virtually generated 5HS CMC RVEs, with principal component analysis (PCA) performed to extract a low-dimensional representation of microstructure. Each microstructure in this ensemble was generated through the open-source software package TexGen (Lin, Brown, and Long 2011), with five generating geometric parameters considered, as listed in Table 1.

The first three geometric parameters listed controlled the generation of each ply within the RVE, defining the cross-sectional shape of the reinforcing tows through the tow major and minor axes, and the tow spacing within the woven architecture. The RVE was then assembled through the stacking of eight plies of the 5HS repeating unit cell (RUC) (Naik

1996). The ply spacing then defined the distance from ply to ply within this stack. With these four geometric parameters, completely dense voxelated microstructures, consisting of tow or matrix voxels, were output by TexGen (Lin, Brown, and Long 2011) with 100^3 total voxel count. Subsequently, matrix voxels were reassigned to pores by a threshold defined by the matrix thickness parameter, resulting in RVEs with three constituents. A Maximin LHD was then used to generate 3125 unique microstructures. It should be highlighted that for a microstructure of size 100^3 with three constituents, the collection of 2-point spatial correlations results in 3×100^3 dimensions, including two sets of auto-correlations, and one set of cross-correlations. A schematic demonstrating the overall process employed in their work can be seen in Figure 11, which outlines the feature engineering protocol from microstructure ensemble generation, to computation of two-point spatial correlations and through performing PCA to establish three-dimensional inputs into the predictive GP model. While it was demonstrated that the active learning framework significantly reduced the computational demand, a core limitation consisted of defining the geometric parameterized microstructure input space to be space-filling rather than the low-dimensional representation (i.e., the final input to the GP model), leading to suboptimal model building as coverage of the output space, in this case the principal components of the set of two-point spatial correlations, were poorly clustered. Subsequently, we present the benefits of employing an OFSD on their three-dimensional microstructure dataset.

The feature engineering protocol shown in Figure 11, can be described as a mapping $f(x) : \mathcal{X} \to \mathcal{E}$, where $\mathcal{X}$ is the input space of $(t_w, t_h, t_s, p_c, m_t)$ and $\mathcal{E}$ is the feature space of the three major features (e_1, e_2, e_3). In order to demonstrate the proposed utility of employing the OSFD algorithm on this dataset, 50 microstructures from the complete ensemble were selected as an initialization subset to estimate the principal component (PC) basis and lock it in place with sequential selection of microstructures in the feature space. The OSFD-EI algorithm was then applied to sequentially select an additional 100 microstructures
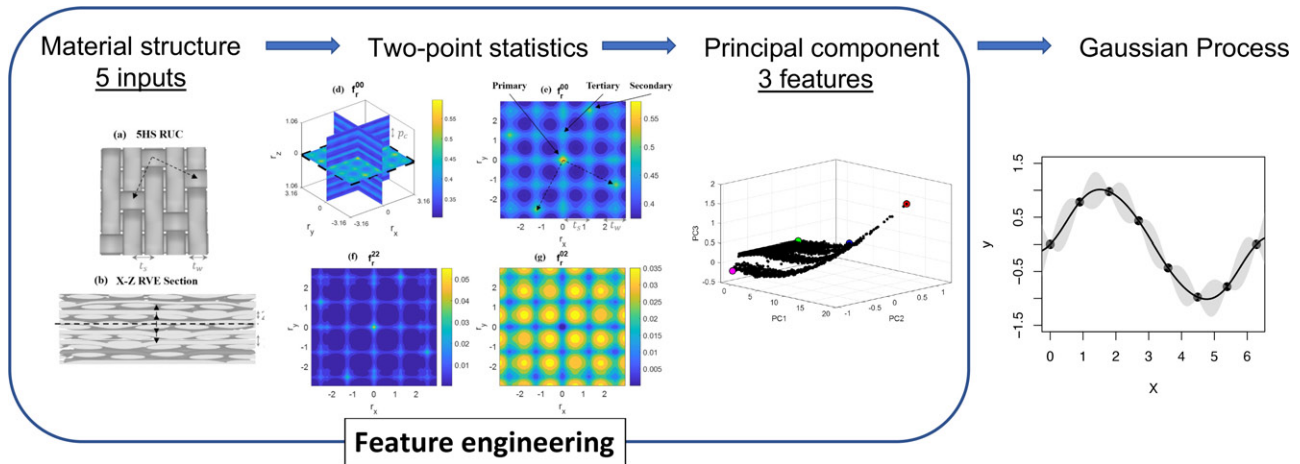
Figure 11. The process to build a reduced-order model for CMC thermal conductivity. Adapted from Generale and Kalidindi (2021).
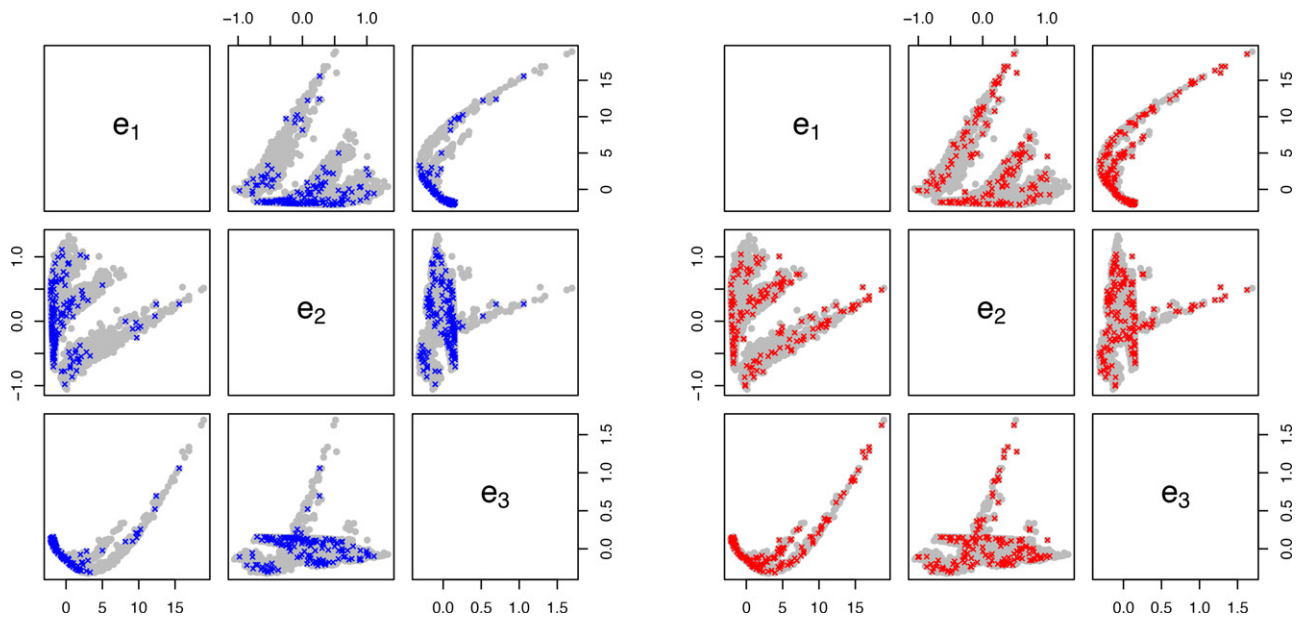

Figure 12. CMC application: features from the whole dataset (gray solid circles), maximin LHD (blue crosses on the left), and OSFD-EI (red crosses on the right).

from the ensemble best filling this feature space, with the results displayed in Figure 12. The identified OSFD is shown overlaid against the PCA representation of the complete ensemble in gray, demonstrating the impressive coverage of this input feature space through a minimal collection of all available microstructures. For comparison purposes, the left panel of Figure 12 displays the results from selecting 150 microstructures using an ISFD, as generated through a 150-run Maximum LHD and selecting the nearest neighboring point. Quantitatively, the fill distance for the features generated by ISFD is 3.43 while that produced by OSFD-EI is 0.67 and is 0.98 by OSFD-greedy (scatterplot is shown in the appendix). This direct comparison clearly displays the poor coverage offered through selecting model-building inputs with an ISFD in comparison to the proposed OSFD for this application.

There are several benefits of employing an OSFD instead of an ISFD in this application. Most importantly, the coverage provided through 150 points generated through the OSFD algorithm is nearly as good as the complete ensemble of 3125 points, conventionally generated through an ISFD. This fact has impor-

tant implications for the overall cost of training the final GP-based structure-property linkage, as microstructure generation and the application of PCA to the complete ensemble with size $3125 \times 3 \times 100^3$ is computationally expensive and represents overhead which can be substantially reduced through the use of an OSFD. The use of an OSFD could also be leveraged to generate additional structures in regions of the input space with insufficient coverage, clearly visualized in Figure 12. The use of such additional points may lead to a more robust final model, as GP-based models are well known to extrapolate poorly.

6. Conclusions

It is common to use a space-filling design in the input space to generate the computer model outputs and develop the input–output relationship. In this article we have demonstrated that for several applications, and contrary to conventional model building workflows, filling the output space is more desirable than filling the input space. We have proposed a sequential design that identifies the largest gap in the output space and

generates an input point to fill-in that gap. Two versions of the sequential design are proposed: a greedy algorithm and an expected improvement-based algorithm. The greedy algorithm is faster than the EI algorithm, but the EI algorithm gives a more robust performance. We have demonstrated the usefulness of the proposed method on two applications involving inverse design and feature-based modeling.

Although we have used nearest neighbor method in our sequential design algorithms because of its computational speed, other surrogate modeling techniques can be used as long as they can be trained quickly with large amounts of data. The traditional GP model did not perform well in our applications, but we believe nonstationary GP models that scale well with the size of data such as laGP (Gramacy and Apley 2015) or twinGP (Vakayil and Joseph 2023) can further improve the performance of OSFD. We leave this as a topic for future research.

Supplementary Materials

Algorithms and additional results: The statements of Algorithms 1–5 are given in the file appendix.pdf along with some additional results.
Codes: The R codes to reproduce Figure 7 and an R Markdown tutorial are included in a zip file. The implementation of the algorithms is available in the R package OSFD (Wang and Joseph 2023).

Disclosure Statement

The authors report there are no competing interests to declare.

Funding

References

An, J., and Owen, A. (2001), "Quasi-Regression," *Journal of Complexity*, 17, 588–607. [72]

Batra, R., Tran, H. D., Kim, C., Chapman, J., Chen, L., Chandrasekaran, A., and Ramprasad, R. (2019), "General Atomic Nighborhood Fingerprint for Machine Learning-based Methods," *Journal of Physical Chemistry C*, 123, 15859–15866. [66]

Fasshauer, G. E. (2007), *Meshfree Approximation Methods with MATLAB* (Vol. 6), Singapore: World Scientific. [66]

Garnett, R. (2023), *Bayesian Optimization*, Cambridge, UK: Cambridge University Press. [69,72]

Geisser, S., and Eddy, W. F. (1979), "A Predictive Approach to Model Selection," *Journal of the American Statistical Association*, 74, 153–160. [70]

Generale, A. P., and Kalidindi, S. R. (2021), "Reduced-Order Models for Microstructure-Sensitive Effective Thermal Conductivity of Woven Ceramic Matrix Composites with Residual Porosity," *Composite Structures*, 274, 114399. [73,75]

Gramacy, R. B. (2020), *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*, Boca Raton, FL: Chapman and Hall/CRC. [66]

Gramacy, R. B., and Apley, D. W. (2015), "Local Gaussian Process Approximation for Large Computer Experiments," *Journal of Computational and Graphical Statistics*, 24, 561–578. [76]

Iyer, A., Yerramilli, S., Rondinelli, J. M., Apley, D. W., and Chen, W. (2022), "Descriptor Aided Bayesian Optimization for Many-Level Qualitative Variables With Materials Design Applications," *Journal of Mechanical Design*, 145, 031701. [65]

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990), "Minimax and Maximin Distance Designs," *Journal of Statistical Planning and Inference*, 26, 131–148. [65,66]

Jones, D. R., Schonlau, M., and Welch, W. J. (1998), "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, 13, 455–492. [69]

Joseph, V. R. (2006), "Limit Kriging," *Technometrics*, 48, 458–466. [69]

——— (2016), "Space-Filling Designs for Computer Experiments: A Review," *Quality Engineering*, 28, 28–35. [65,66]

Kalidindi, S. R. (2015), *Hierarchical Materials Informatics: Novel Analytics for Materials Data*, Amsterdam, Netherlands: Elsevier. [74]

Krishna, A., Craig, S. R., Shi, C., and Joseph, V. R. (2022), "Inverse Design of Acoustic Metasurfaces Using Space-Filling Points," *Applied Physics Letters*, 121, 071701. [65,66,72]

Krishna, A., Tran, H., Ramprasad, R., Huang, C., and Joseph, V. R. (2023), "Adaptive Exploration and Optimization of Materials Crystal Structures," *INFORMS Journal on Data Science*. DOI:10.1287/ijds.2023.0028. [65,66]

Lin, H., Brown, L. P., and Long, A. C. (2011), "Modelling and Simulating Textile Structures Using TexGen," *Advanced Materials Research*, 331, 44–47. [74]

Lu, L., and Anderson-Cook, C. M. (2021), "Input-Response Space-Filling Designs," *Quality and Reliability Engineering International*, 37, 3529–3551. [65,66,70]

Mak, S. (2016), *minimaxdesign: Minimax and Minimax Projection Designs,* R package version 0.1.5. [71]

Mak, S., and Joseph, V. R. (2018), "Minimax and Minimax Projection Designs Using Clustering," *Journal of Computational and Graphical Statistics*, 27, 166–178. [67]

Mak, S., Sung, C.-L., Wang, X., Yeh, S.-T., Chang, Y.-H., Joseph, V. R., Yang, V., and Wu, C. J. (2018), "An Efficient Surrogate Model for Emulation and Physics Extraction of Large Eddy Simulations," *Journal of the American Statistical Association*, 113, 1443–1456. [65]

Molesky, S., Lin, Z., Piggott, A. Y., Jin, W., Vucković, J., and Rodriguez, A. W. (2018), "Inverse Design in Nanophotonics," *Nature Photonics*, 12, 659–670. [72]

Morris, M. D., and Mitchell, T. J. (1995), "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, 43, 381–402. [66]

Naik, R. (1996), "Analysis of Woven and Braided Fabric-Reinforced Composites," in *Composite Materials: Testing and Design: Twelfth Volume*, eds. R. Deo and C. Saff, pp. 239–263, West Conshohocken, PA: ASTM International. [74]

Ren, Z., Tian, S. I. P., Noh, J., Oviedo, F., Xing, G., Li, J., Liang, Q., Zhu, R., Aberle, A. G., Sun, S., et al. (2022), "An Invertible Crystallographic Representation for General Inverse Design of Inorganic Crystals with Targeted Properties," *Matter*, 5, 314–335. [72]

Rhee, C.-H., Zhou, E., and Qiu, P. (2017), "Space-Filling Design for Nonlinear Models," arXiv preprint arXiv:1710.11616. [66,68,70]

Roustant, O., Ginsbourger, D., and Deville, Y. (2012), "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-based Metamodeling and Optimization," *Journal of Statistical Software*, 51, 1–55. [70]

Santner, T. J., Williams, B. J., and Notz, W. I. (2019), *The Design and Analysis of Computer Experiments*, New York, NY: Springer. [65]

Solteiro Pires, E., Tenreiro Machado, J., de Moura Oliveira, P., Boaventura Cunha, J., and Mendes, L. (2010), "Particle Swarm Optimization with Fractional-Order Velocity," *Nonlinear Dynamics*, 61, 295–301. [70]

Striegel, C., Biehler, J., Wall, W. A., and Kauermann, G. (2022), "A Multi-fidelity Function-on-Function Model Applied to an Abdominal Aortic Aneurysm," *Technometrics*, 64, 279–290. [65]

Sun, F., and Gramacy, R. B. (2021), *maximin: Space-Filling Design under Maximin Distance,* R package version 1.0-4. [70]

Torquato, S. (2002), *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*, New York, NY: Springer. [74]

Vakayil, A., and Joseph, V. R. (2023), "A Global-Local Approximation Framework for Large-Scale Gaussian Process Modeling," arXiv preprint arXiv:2305.10158. [76]

Wang, S., and Joseph, V. R. (2023), *OSFD: Output Space-Filling Design,* R package version 0.1.0. [76]

Zhang, N., and Apley, D. W. (2014), "Fractional Brownian Fields for Response Surface Metamodeling," *Journal of Quality Technology*, 46, 285–301. [70]