No. 2018-022

PREDICTION FOR BIG DATA THROUGH KRIGING: SMALL SEQUENTIAL AND ONE-SHOT DESIGNS

By

Jack P.C. Kleijnen, Wim C.M. van Beers

9 July 2018

ISSN 0924-7815
ISSN 2213-9532

Prediction for Big Data through Kriging: Small Sequential and One-Shot Designs

Jack P.C. Kleijnen and Wim C.M. van Beers

July 9, 2018

Abstract

Kriging or Gaussian process (GP) modeling is an interpolation method that assumes the outputs (responses) are more correlated, the closer the inputs (explanatory or independent variables) are. A GP has unknown (hyper)parameters that must be estimated; the standard estimation method uses the "maximum likelihood" criterion. However, big data make it hard to compute the estimates of these GP parameters, and the resulting Kriging predictor and the variance of this predictor. To solve this problem, some authors select a relatively small subset from the big set of previously observed "old" data; their method is sequential and depends on the variance of the Kriging predictor. The resulting designs turn out to be "local"; i.e., most design points are concentrated around the point to be predicted. We develop three alternative one-shot methods that do not depend on GP parameters: (i) select a small subset such that this subset still covers the original input space–albeit coarser; (ii) select a subset with relatively many—but not all—combinations close to the new combination that is to be predicted, and (iii) select a subset with the nearest neighbors (NNs) of this new combination. To evaluate these designs, we compare their squared prediction errors in several numerical (Monte Carlo) experiments. These experiments show that our NN design is a viable alternative for the more sophisticated sequential designs.

Keywords: Kriging; Gaussian process; big data; experimental design; nearest neighbor

JEL: C0, C1, C9, C15, C44

1 Introduction

Kriging or *Gaussian process* (GP) modeling is a popular statistical method for interpolation of input/output (I/O) data; these inputs are also called explanatory or independent variables, and these outputs are also called responses. Kriging is applied in many scientific disciplines; e.g., operations research / management science (OR/MS), engineering, machine learning, and geostatistics. Each discipline uses its own terminology; we use the terminology of OR/MS, especially, simulation. In this publication we assume that the I/O data are noise-free (as in deterministic simulation or computer experiments), so Kriging is an *exact* interpolator; i.e., the Kriging predictions for old input combinations or "points" are exactly equal to the previously observed outputs for these "old" combinations. Furthermore, we assume that the goal of Kriging is to predict the output for a '"new" point ; other goals (namely, validation, sensitivity analysis, optimization, and uncertainty analysis) are discussed in Kleijnen (2017).

Note: If the output data have noise (caused by pseudorandom numbers (PRNs) in discrete-event simulation or by observation error in real-life applications), then the Kriging predictor is not an exact interpolator. Hasty readers may skip paragraphs that start with "Note:".

Prediction through Kriging may be applied in *real-time* on-line decision-making. Kamiński (2015) observes that Kriging may be applied to simulation offered as a service over the Internet to end-users; e.g., a complex financial simulation predicts the output, given an investor's portfolio allocation decision. Such simulations can be rather time-consuming, whereas financial analysts prefer fast responses from the Internet; such fast responses can be provided by Kriging. Fouladinejad et al. (2017) also uses Kriging (besides neural nets) for a real-time driving simulator.

Big data is everywhere; see Xu et al. (2016). We distinguish between big data generated by *real systems* and by *simulation models* (of real systems, existing or planned). Big data on real systems are provided by electronic data capture via sensors etc.; e.g., supermarkets with point-of-sale systems (POSSs) capture massive data on their many stock keeping units (SKUs). Big data on simulated systems are collected if the simulation is "inexpensive"; i.e.,the simulation requires "little" computer time to obtain the output of the model—given the input data—and many input combinations are simulated; e.g., a simulation of a queueing systems with multiple servers, each with first-in-first-out (FIFO) priority rule (like a supermarket).

We might think that inexpensive simulations do not need Kriging, because we can quickly obtain a simulation response for a given input combinations. However, the queueing example has random output, so for a high traffic rate (which is the arrival rate divided by service rate) we need many replications. (If the simulation is expensive, then we assume that simulation does not create big data; e.g., a car-crash simulation may take a day to obtain the output for a given input combination.) Furthermore, we assume that the simulation is run on a traditional personal computer; i.e., we do not consider parallel computing architectures; such architectures are considered in Gramacy (2016), Gramacy et al. (2015), Guhaniyogi and Banerjee (2018), Guinness (2018), Gutiérrez de Ravé et al. (2014), Morgan et al. (2017), Van Stein et al. (2017), and Xu et al. (2016).

Note: Big data may also refer to a system (real or simulated) that has *many dimensions*. We, however, assume that the number of dimensions has been made manageable through "factor screening", which implies that we assume that among the many factors only relatively few factors are really important. Such screening is discussed in Kleijnen (2015, pp. 135–178) and Woods and

Lewis (2016) with its 117 references. Alternative methods for reducing the dimensionality use "moving least squares" (MLS), "partial least squares" (PLS), and "principal component analysis" (PCA): see Wang et al. (2011), Bouhlel et al. (2016, 2017), and Kajero et al. (2017), respectively.

Kriging is applied in many different scientific disciplines, which have their own terminology and standard mathematical symbols. We follow the terminology and symbols used in Kleijnen (2015). We limit our investigation of Kriging and big data to *noise-free* output data. We notice that Gramacy (2016) also allows output that has noise with a small constant "intrinsic" variance; in future research we may investigate intrinsic noise with big variances that vary with the input combinations; also see Section 6. We use a frequentist approach (instead of a Bayesian approach, which is used in Angelino et al. (2016), Damianou (2015), Guhaniyogi and Banerjee (2018), Gramacy (2016), Nickson et al. (2015), and Pronzato and Rendas (2017)). An advantage of such a frequentist approach is that we do not need to specify a prior distribution for the Kriging (hyper)parameters; we find it hard to specify such a prior distribution. For a further discussion of frequentist and Bayesian approaches we also refer to Efron (2015)

There are many sophisticated methods to analyze Kriging models for big data. Guhaniyogi & Banerjee (2018) states: “There is a burgeoning literature on approaches for analyzing large spatial datasets”; that article briefly discusses several approaches, and proposes a new Bayesian approach. We also refer to Bradley et al. (2016), Chilès and Desassis (2018), Damianou (2015), Guinness (2018), Meng and Ng (2016), Nickson et al. (2015), Sung et al. (2016), Tzeng and Huang (2018), and Van Stein et al. (2017). We, however, focus on the method that is detailed in Gramacy (2016) and Gramacy and Apley (2015), and compare our methods with Gramacy's method (Gramacy's method is also compared with a new method in Guhaniyogi and Banerjee (2018)). We observe that two NN variations are used in Gramacy (2016, Section 3.1) and Gramacy and Apley (2015, Fig. 2, Table 1, SM.1), but these publications use a Bayesian analysis and are less detailed.

Note: Meng and Ng (2016) models the global trend through so-called inducing points that sufficiently summarize the observed data points, and smooths out the local fluctuations around this trend; i.e., this model clusters the observed data points into groups based on their location and their outputs, where each group consists of a representative inducing point (usually the centroid of the group). Our approach is much simpler; e.g., we do not need a support vector machine (SVM) like Meng and Ng (2016) does or some other method (e.g., a Voronoi tessellation) to construct local areas; moreover, at the boundaries of these local regions the resulting Kriging model is discontinuous. Furthermore, we do not need to select the number of local areas. Finally, we do not use so-called latent outputs such as the average output within a local area; i.e., we use actually observed outputs. The induced points are similar to our subset of n points, explained in the next section. Van Stein et al. (2017) also fits a local Kriging model to each "cluster" of data; the number of clusters needs to be prespecified, and the selection of these clusters may use various methods.

Bradley et al. (2016) compares seven Kriging variants for spatial data. Sung et al. (2016) discusses two variants of the method detailed in Gramacy and Apley (2015); these variants are still much more complicated than our method is. A recent example of applying inducing points (in a Bayesian framework) is Yuan et al. (2017). Finally, Wang and Chen (2017) uses "pruning", which starts with the big set (of size N) and removes points, whereas other methods start with a small set and increase that set (until it has size $n \ll N$).

We organize the rest of this article as follows. Section 2 summarizes the basics of Kriging. Section 3 presents a one-shot design for Kriging in big data, using Latin hypercube sampling (LHS). Section 4 presents numerical experiments. Section 5 replaces LHS by NNs. Section 6 summarizes conclusions and possible topics for future research. Appendix 1 in the Online Appendix gives a list of abbreviations and symbols.

2 Kriging: basics

In this article we distinguish between the *size* of the big I/O dataset—denoted by N—and the size of a selected subset—denoted by n—where "size" means the number of input combinations, and the corresponding output. We assume a scalar or univariate output (so the output is not a vector or a multivariate random variable). There are k inputs x_j ($j = 1, ..., k$).

The so-called *ordinary Kriging* (OK) variant is

$$y(\mathbf{x}) = \mu + M(\mathbf{x}) \tag{1}$$

where μ is the constant mean $\mathrm{E}[y(\mathbf{x})]$ and $M(\mathbf{x})$ is a zero-mean stationary GP, so its covariances depend only on the distance between the input combinations $\mathbf{x}$ and $\mathbf{x}'$. We call $M(\mathbf{x})$ the extrinsic noise. This model gives the *linear predictor* (say) $\widehat{y}(\mathbf{x}_0)$ for the new input combination $\mathbf{x}_0 = (x_{0;j})$ that combines the N old outputs $\mathbf{w}_N$—or briefly $\mathbf{w}$—that are observed at the N old input combinations of the k inputs $\mathbf{x}_i = (x_{i;1}, ..., x_{i;k})$ with $i = 1, ..., N$ (which gives the $N \times k$ matrix $\mathbf{X}$):

$$\widehat{y}(\mathbf{x}_0, \boldsymbol{\lambda}) = \textstyle\sum_{i=1}^{N} \lambda_i w(\mathbf{x}_i) = \boldsymbol{\lambda}'\mathbf{w}. \tag{2}$$

This $\widehat{y}(\mathbf{x})$ is an *exact interpolator*; i.e., if $\mathbf{x}_0 = \mathbf{x}_i$, then $\widehat{y}(\mathbf{x}_i) = w(\mathbf{x}_i)$. Moreover, $\widehat{y}(\mathbf{x})$ minimizes the *mean squared prediction error* (MSPE), so the optimal weights $\boldsymbol{\lambda}$ in (2) are

$$\lambda_o' = [\sigma_M(\mathbf{x}_0) + \mathbf{1}\frac{1 - \mathbf{1}'\boldsymbol{\Sigma}_M^{-1}\sigma(\mathbf{x}_0)}{\mathbf{1}'\boldsymbol{\Sigma}_M^{-1}\mathbf{1}}]'\boldsymbol{\Sigma}_M^{-1} \tag{3}$$

assuming that a valid metamodel of the outputs $\mathbf{w}$ is a stationary GP with $\boldsymbol{\Sigma}_M = (\sigma_{i;i'}) = (\mathrm{Cov}(y_i, y_{i'}))$ $(i, i' = 1, ..., N)$ denoting the $N \times N$ matrix with the covariances between the metamodel's old outputs y_i, and $\sigma_M(\mathbf{x}_0) = (\sigma_{0;i}) = (\mathrm{Cov}(y_0, y_i))$ denoting the N-dimensional vector with the covariances between the metamodel's new output y_0 and the N old outputs y_i. Obviously, $\boldsymbol{\Sigma}_M$ is determined by the old I/O data, whereas $\sigma_M(\mathbf{x}_0$ varies with $\mathbf{x}_0$. Furthermore, λ_i decreases with the *distance* between $\mathbf{x}_0$ and $\mathbf{x}_i$. The optimal weights λ_o in (3) are determined by the *Kriging (hyper)parameters* $\psi = (\mu, \tau^2, \theta')'$ where τ^2 denotes $\mathrm{Var}(y_i) = \sigma_{i;i} = \sigma_i^2 = \sigma^2$, and θ denotes the k-dimensional vector with the parameters θ_j ($j = 1, ..., k$) of the Gaussian correlation function defined in (6) that we assume. Obviously, $\mathbf{\Sigma}_M = \tau^2\mathbf{R}$ where $\mathbf{R} = (\rho_{i;i'})$ denotes the corresponding correlation matrix; furthermore, furthermore, $\rho(\mathbf{x}_0) = \tau^{-2}\sigma_M(\mathbf{x}_0)$. Substitution of λ_o into (2) gives

$$\widehat{y}(\mathbf{x}_0, \psi) = \mu + \sigma_M(\mathbf{x}_0)'\mathbf{\Sigma}_M^{-1}(\mathbf{w} - \mu\mathbf{1}_N) \tag{4}$$

where $\mathbf{1}_N$ denotes the N-dimensional vector with all elements equal to 1. Then

$$\text{MSPE } [\widehat{y}(\mathbf{x}_0, \psi)] = \tau^2 - \sigma_M(\mathbf{x}_0)'\mathbf{\Sigma}_M^{-1}\sigma_M(\mathbf{x}_0) + \frac{[1 - \mathbf{1}_N'\mathbf{\Sigma}_M^{-1}\sigma_M(\mathbf{x}_0)]^2}{\mathbf{1}_N'\mathbf{\Sigma}_M^{-1}\mathbf{1}_N}. \tag{5}$$

Because $\widehat{y}(\mathbf{x}_0, \psi)$ is unbiased, we obtain MSPE $[\widehat{y}(\mathbf{x}_0, \psi)] = \mathrm{Var}[\widehat{y}(\mathbf{x}_0, \psi)]$. Finally, $\mathrm{Var}[\widehat{y}(\mathbf{x}_0, \psi)] = 0$ if $\mathbf{x}_0 = \mathbf{x}_i$.

There are several types of correlation functions; see (e.g.) Rasmussen and Williams (2006, pp. 80–104). In simulation, the most popular function is the *Gaussian correlation function* (which is also used in Gramacy 2015):

$$\rho(\mathbf{h}, \theta) = \prod_{j=1}^{k} \exp\ \left(-\theta_j h_j^2\right) = \exp\ (-\sum_{j=1}^{k} \theta_j h_j^2) \text{ with } \theta_j \geq 0 \tag{6}$$

with distance vector $\mathbf{h} = (h_j)$ where $h_j = |x_{g;j} - x_{g';j}|$ and $g, g' = 0, 1, ..., N$. The maximum likelihood estimator (MLE) $\widehat{\psi}$ of ψ is the solution of

$$\min_{\psi} \ln[|\tau^2\mathbf{R}(\theta)|] + (\mathbf{w} - \mu\mathbf{1}_N)'[\tau^2\mathbf{R}(\theta)]^{-1}(\mathbf{w} - \mu\mathbf{1}_N) \text{ with } \theta \geq \mathbf{0}. \tag{7}$$

where $|\mathbf{R}|$ denotes the determinant of $\mathbf{R}$. Solving (7) is a mathematical challenge; e.g., different solutions $\widehat{\psi}$ may result from different software packages or from different starting values for the same package; see Erickson et al. (2017).

Big data implies that N is so big that the computation of $\mathbf{R}^{-1}$ and $|\mathbf{R}|$ gives numerical problems. More precisely, these computations require $O(N^3)$ matrix decompositions; see Damianou (2015, p. 20), Gramacy (2016), Nickson et al. (2015), and Van Stein et al. (2017). Furthermore, the more space-filling (or clustered) the design is, the higher the *condition number* of $\mathbf{R}$ is; see Lim et al. (2017). To solve these big-data problems, we shall present several methods in Section 3.

Note: Because we use specific hardware and software in our experiments reported in Section 4, we now perform the following preliminary experiments. We use an Intel®i5 processor with a CPU speed of 3.30GHz, a memory of 4.00 GB, and Windows 7 Professional, 64-bit. For our GP computations we use DACE, which uses MATLAB (we use MATLAB's version 2017b); DACE is detailed in Lophaven et al. (2010). Our preliminary experiments with DACE show that N between 1,000 and 10,000 lead to an explosive growth of computing time; N = 9,500 and N = 10,000 cause "out of memory" stops. Details are available from the authors.

In practice, we simply *plug* $\widehat{\psi}$ into (4), and get

$$\widehat{y}(\mathbf{x}_0, \widehat{\psi}) = \widehat{\mu} + \widehat{\sigma}_M(\mathbf{x}_0)' \widehat{\mathbf{\Sigma}}_M^{-1}(\mathbf{w} - \widehat{\mu}\mathbf{1}_N). \tag{8}$$

So the Kriging predictor becomes nonlinear. We also plug-in $\widehat{\psi}$ into (5), to obtain the plug-in estimator of the predictor variance:

$$s^2[\widehat{y}(\mathbf{x}_0, \widehat{\psi})] = \widehat{\tau}^2 - \widehat{\sigma}_M(\mathbf{x}_0)' \widehat{\mathbf{\Sigma}}_M^{-1} \widehat{\sigma}_M(\mathbf{x}_0) + \frac{[1 - \mathbf{1}' \widehat{\mathbf{\Sigma}}_M^{-1} \widehat{\sigma}_M(\mathbf{x}_0)]^2}{\mathbf{1}' \widehat{\mathbf{\Sigma}}_M^{-1} \mathbf{1}}. \tag{9}$$

This estimator underestimates the ("true") Kriging variance; see Kleijnen (2015, pp. 191—197) and also Chevalier et al. (2014).

This $s^2[\widehat{y}(\mathbf{x}_0, \widehat{\psi})]$ is used in *efficient global optimization* (EGO), to guide the search for the optimal input combination. This $s^2[\widehat{y}(\mathbf{x}_0, \widehat{\psi})]$ is also used—combined with $\widehat{y}(\mathbf{x}_0, \widehat{\psi})$ (defined in (8))—to construct the following two-sided confidence interval (CI) with nominal coverage probability $1 - \alpha$:

$$\widehat{y}(\mathbf{x}_0, \widehat{\psi}) - z_{\alpha/2} s[\widehat{y}(\mathbf{x}_0, \widehat{\psi})] < w(\mathbf{x}_0) < \widehat{y}(\mathbf{x}_0, \widehat{\psi}) + z_{\alpha/2} s[\widehat{y}(\mathbf{x}_0, \widehat{\psi})]. \tag{10}$$

To obtain an actual coverage probability close to the nominal value $1 - \alpha$, we might replace $z_{\alpha/2}$ by a Student statistic with f degrees of freedom, denoted by t_f. An unsolved problem is the proper choice of f; we might try $f = n$ - k. The CI in (10) is related to the Bayesian "predictive distribution", which is used in Gramacy (2016, p. 3).

Instead of a single new point $\mathbf{x}_0$ we may consider a *test set* with n_0 new points, which defines the (say) $n_0 \times k$ matrix $\mathbf{X}_0$ (we distinguish between the test set and the "training set", which consists of the old I/O data). Once we have computed $\widehat{\mu}$, $\widehat{\tau}^2$, and $\widehat{\mathbf{\Sigma}}_M^{-1}$ from $\mathbf{X}_N$ and $\mathbf{w}_N$, the computations of $\widehat{y}$ and $s^2(\widehat{y})$ for $\mathbf{X}_0$ is fast; see (4) and (9).

OK uses the constant $\mu = \mathrm{E}(y)$, whereas *universal Kriging* (UK) uses a trend (e.g., $\mathrm{E}(y) = \beta'\mathbf{x}$); details on UK are found in Kleijnen (2015, pp. 197–198) and also in Chen et al. (2016) and Mukhopadhyay et al. (2016).

3 Designs for Kriging with big data

We discuss designs for prediction through Kriging in case of big data. We start with Gramacy (2016)'s rather sophisticated sequential designs; next we derive three types of one-shot designs:

1. Gramacy (2016)'s sequential designs (see Section 3.1)
2. LHS designs with uniform input distributions (Section 3.2)

3. LHS designs with triangular input distributions (Section 3.3)

4. NN designs (Section 3.5).

We assume that the k inputs are limited to a k-dimensional hypercube $[0,1]^k$, as the Kriging literature usually assumes. Our one-shot designs are not Bayesian. Because our designs do not use iterative searches for subset members, we need less computer time than Gramacy (2016)'s sequential designs do. We expect that our Kriging predictor is less accurate than Gramacy's Kriging predictors.

3.1 Gramacy (2016)'s designs

Gramacy (2016) defines *local Kriging* as the Kriging predictor—and its variance—that use only a subset with n I/O observations from the full set with N observations such that this subset is primarily comprised of data close to the new point $\mathbf{x}_0$; see (2). This subset is also called the set of *inducing points*, and the resulting metamodel is called the *sparse GP*; see Damianou (2015, pp. 20–23), Gal et al. (2014, p. 3), and Van Stein et al. (2017, p. 5).

The reason for choosing a subset with most data close to $\mathbf{x}_0$ is that the correlation $\rho\{y(\mathbf{x}_0), y(\mathbf{x}_i)\}$ (following from (6)) diminishes quickly as $\mathbf{x}_i$ moves away from $\mathbf{x}_0$; i.e., $y(\mathbf{x}_i)$ then has a vanishingly small influence on $y(\mathbf{x}_0)$. To obtain good estimates of the correlation parameters θ, it is desirable to have some spread in the subset. More specifically, Gramacy (2016) starts an iterative search using a small NN set of size n_0 ($n_0 \ll n$), and selects the next point sequentially—based on the prediction errors that use sequentially (re)estimated parameters of ψ. These errors are estimated using one of three different approximations of the Bayesian MSPE of the Kriging predictor for the currently available data (also see our frequentist $s^2[\widehat{y}(\mathbf{x}_0, \widehat{\psi})]$ in (9)). The last two approximations are simplifications of its predecessor, in order to save computer time when searching for $\mathbf{Z}_n$; so we expect the SPE to increase as the approximation becomes more simplified. For detailed derivations of these designs we refer to Gramacy (2016). In our numerical experiments with specific examples we shall present Gramacy's specific designs.

3.2 LHS with uniform input distributions

LHS is the most popular design type in Kriging for other goals than prediction. Actually, McKay et al. (1979) invented LHS—as an alternative for crude Monte Carlo sampling—for *risk analysis* or *uncertainty analysis* through deterministic simulation models that have (random) uncertain inputs (this analysis estimates the probability of the output exceeding a given threshold as a function of an uncertain input x_j; for details see Kleijnen (2015, pp. 218–222)). LHS assumes that an adequate metamodel is more complicated than a low-order polynomial, but LHS does not assume a specific metamodel (e.g., a Kriging or a linear regression model). Usually, risk analysis assumes that the k inputs are *independently* distributed (so their joint distribution is the product of the k individual marginal distributions); we also use this assumption.

In this subsection we detail LHS with *uniform* distributions (symbol U), so $x_j \sim \mathrm{U}(0,1)$. In the next subsection we shall detail LHS with non-uniform—namely, triangular—distributions with their modes at $x_{0;j}$ where $x_{0;j}$ denotes the j^{th} coordinate of $\mathbf{x}_0$ and $j = 1, ..., k$. For the resulting LHS design we find the NN in the given big data set. We use LHS to select n points from the given N points (with $n \ll N$) such that we may expect this sample to cover the same experimental space as the N points do (namely, $[0,1]^k$).

Note: Kleijnen (2015, p. 198) discusses LHS and alternative space-filling designs; e.g., orthogonal array, uniform, maximum entropy, minimax, maximin, integrated mean squared prediction error, and "optimal" designs. Kleijnen (2015, pp. 198–203) gives additional details on LHS. Van Stein et al. (2017) mentions taking n data points at random. Benková et al. (2015) discusses space-filling designs that may satisfy various criteria and input constraints (such that the input space is not a k-dimensional cube), including so-called bridge designs. Chen et al. (2016) shows that "there is substantial variation in prediction accuracy over equivalent designs".

Whatever the marginal distributions are, LHS with a sample size n defines n mutually exclusive and exhaustive subintervals (or classes) with equal probability (namely, $1/n$) for $x_{g;j}$ with $g = 1, .., n$ and $j = 1, ..., k$ (so LHS gives an $n \times k$ matrix or table). We denote these subintervals by $[l_{g;j},\, h_{g;j}]$; the standardization $0 \leq x \leq 1$ implies $l_{1;j} = 0$ and $h_{n;j} = 1$. Altogether, if F_j denotes the *cumulative distribution function* (CDF) of x_j, then

$$P(l_{g;j} \leq x_j \leq h_{g;j}) = F_j(h_{g;j}) - F_j(l_{g;j}) = F_j(\frac{g}{n}) - F_j(\frac{g-1}{n}) = \frac{1}{n} \qquad (11)$$

where $\min(g-1)/n = (1 - 1)/n = 0$, so $F_j((g-1)/n) = F_j(0) = 0$ because $\min x_j = 0$; likewise, $\max(g)/n = n/n = 1$, so $F_j(g/n) = F_j(1) = 1$ because $\max_j x = 1$. Hence, (11) implies that $x_j \sim \mathrm{U}(0,1)$ requires each interval to have length $1/n$. (however, a triangular distribution with mode $x_{0;j}$ requires subintervals $[l_{g;j},\, h_{g;j}]$ to be relatively short, compared with the other subintervals; see Section 3.3).

Whatever the marginal distributions are, LHS offers the following two *options*:

(i) x_j is *fixed* to the n *midpoints* (symbol m_j) of its n subintervals, so $x_{g;j} = m_{g;j} = (l_{g;j} + h_{g;j})/2$; e.g., if $x \sim \mathrm{U}(0,1)$, then these midpoints are equispaced with distance $1/n$ over the interval $[0,1]$ so these midpoints are $1/(2n)$, $3/(2n)$, ..., $1 - 1/(2n) = (2n-1)/(2n)$.

(ii) x_j is *sampled* within its subinterval, accounting for F_j.

MATLAB's function "lhsdesign" implements both options, using a parameter called "'smooth" that can be turned "off" or "on" where "off" produces points at the midpoints; the default is "on". We think that option (ii) (sampling instead of using midpoints) has the disadvantage that it may give two values—in two neighboring subintervals—that are very close together, so the two resulting outputs w are close together (assuming a smooth I/O function, as Kriging does) and give little new information (because these outputs have a high positive correlation). Moreover, we assume that the correlation function (6) is anisotropic (i.e., it is the product of k one-dimensional correlation functions); option (i) (midpoints) ensures n realizations of x_j that are "wide apart"—which we conjecture gives better estimates of the correlation parameters θ_j. Furthermore, this option never samples $x \downarrow 0$ or $x \uparrow 1$ ($x = 0$ or $x = 1$ is impossible because x is continuous; $x = \epsilon$ or $x = 1$ - ϵ is also undesirable); we wish to avoid these two extreme values because Kriging can use the output near $x = 0$ only to predict the output at $x > 0$ (not at $x < 0$), so Kriging can use $x = 0$ only in one direction; a similar argument holds for $x \uparrow 1$.

Option (i) (with midpoints) implies that x_j has a *discrete* PDF, so (11) becomes

$$P(x_{g;j} \ = m_{g;j}) = F_j(h_{g;j}) - F_j(l_{g;j}) = F_j(\frac{g}{n}) - F_j(\frac{g-1}{n}) = \frac{1}{n}. \tag{12}$$

Furthermore, LHS *samples without replacement*, so the midpoint $m_{j;g}$ is sampled only once (in the sample of size n). We denote the *inverse CDF* by F_j^{-1}, so $y = F_j(x)$ with $0 \leq y \leq 1$ implies $x = F_j^{-1}(y)$; we notice that $\mathrm{U}(0,1)$ and $\mathrm{T}(x_{0;j})$ ($\mathrm{T}(x_{0;j})$ will be discussed in section 3.3) imply that F_j is continuous. Altogether, $\mathbf{x}_j$ is a permutation of the n values $F_j^{-1}(0.5/n)$, $F_j^{-1}(1.5/n)$, ..., $F_j^{-1}(1 - 0.5/n)$.

Because MATLAB allows only a uniform PDF or a normal PDF (via "lhsnorm", which can give values outside $[0,1]$), we program the options (i) and (ii) for both $\mathrm{U}(0,1)$ and $\mathrm{T}(x_{0;j})$. To implement sampling without replacement for option (i) (which gives a hypergeometric distribution), we make $P(x_{j;g} = m_{j;g})$ $= 0$ as soon as we have sampled $x_{j;g} = m_{j;g}$, and for the values that are not yet sampled we increase $P(x_{j;g} = m_{j;g})$ to $1/n_g$ where n_g denotes the number of values that remain to be sampled; e.g., if $n_g = 1$, then the only remaining value is sampled with probability 1.

Note: If n is even, then $F_j^{-1}(0.5)$ equals the median $x_{(n/2);j}$ where a subscript within parentheses denotes an order statistic (such as the median). If the PDF of x_j is symmetric with mode $x_{0;j}$, then the mode and median coincide.

Option (ii) (sampling instead of midpoints) is also used in MATLAB's "lhsdesign" for $\mathrm{U}(0,1)$ (not $\mathrm{T}(x_0)$). This option first samples $r \sim \mathrm{U}(a,b)$ with $a =$ $(g-1)/n$ and $b = g/n$ where $g = 1, .., n$; next this option computes

$$x = F_j^{-1}(r) \text{ with } r \sim U(\frac{g-1}{n}, \frac{g}{n}). \tag{13}$$

Our algorithm 1 is a (pseudo)algorithm for LHS for option (i) (using midpoints) with n combinations of k inputs, which gives the $n \times k$ design matrix $\mathbf{X}_{\mathrm{L}}$ (the subscript L stands for LHS) (various algorithms for LHS are referenced in Kleijnen (2015, p. 200); recent algorithms are detailed in Dong and Nakayama (2017), and Le Guiban et al. (2017)):

Algorithm 1

1. Read n, k, F_j (j = 1, ..., k).
2. Initialize: $j = 1$.
3. Use F_j to divide the range of x_j into n mutually exclusive and exhaustive intervals of equal probability with midpoint $m_{j;g}$ (g = 1, ..., n), and find $\mathbf{x}_j = (m_{j;1}, m_{j;2}, ..., m_{j;n})'$.
4. Randomly permute the n elements of $\mathbf{x}_j$, and save the result as column j of $\mathbf{X}_{\mathrm{L}}$.
5. If $j < k$ then $j = j + 1$ and go to Step 3; else stop.

For option (ii) (sampling instead of midpoints), Step 3 becomes: Use F_j to divide the range of x_j into n mutually exclusive and exhaustive intervals of equal probability, and apply (13) to find $\mathbf{x}_j = (x_{j;1}, x_{j;2}, ..., x_{j;n})'$.

For both options, however, the *random* permutations in Step 4—for which we may use MATLAB'S "randperm"—may give a "bad" $\mathbf{X}_{\mathrm{L}}$. To decide on a "good" $\mathbf{X}_{\mathrm{L}}$, our algorithm needs a criterion. We decide to use the *maximin* criterion, which maximizes the minimum Euclidean distance between the n k-dimensional points in $[0, 1]^k$ (there are $n(n-1)/2$ distances; some distances may have the same value). This criterion is also the default in MATLAB's "lhsdesign". This criterion means that we perform these random permutations (say) M times, and select the design among the M candidates that maximizes the minimum distance between any two points $\mathbf{x}'_g$ and $\mathbf{x}'_{g'}$ with $g \neq g'$ (g' = 1, ..., n) and $\mathbf{x}'_g$ denoting the k-dimensional row vector $(x_{g;1}, ..., x_{g;k})$. We use MATLAB's default $M = 5$.

Note: Besides $M = 5$ we also experiment with several M values between 1 and 10^6, for option (i), U$(0, 1)$, $k = 2$ and $n = 25$. We find that $M = 10^6$ gives the maximin value 0.126. $M = 5$ gives between 0.089 and 0.113 for our $\mathbf{X}_{\mathrm{L}}$ and between 0.057 and 0.089 for MATLAB's $\mathbf{X}$, in six replications.

Note: Humans are excellent pattern recognizers, whereas computers are not. Consequently, computers may use one or more mathematical criteria, whereas we may view a plot and decide whether we accept the pattern as space-filling. However, if k is high, then our pattern recognition becomes questionable.

Note: At the start of this section we assumed that the k standardized inputs are limited to $[0, 1]^k$; however, we can easily adapt LHS for a *constrained* experimental region. For example, suppose there are $k = 2$ inputs; namely, $x_1 = p_1$ and $x_2 = p_2$ such that $p_1 + p_2 = 1$. We would then adapt our LHS with n input combinations, as follows. Select a value for p_1 within subinterval n, and sample p_2 within [0, 1 - p_1]. Place this p_2-value on the *tabu* list, which is the list with p_2 values that should not be sampled anymore. Next select p_1 within subinterval $n - 1$, and sample p_2 within [0, 1 - p_1]. Place this p_2-value on the *tabu* list. And so on, until we have selected p_1 within subinterval 1, and sampled p_2 within [0, 1 - p_1].

LHS does not impose a strict mathematical relationship between n and k (whereas a grid with s subintervals implies $n = s^k$; e.g., a grid with 10 values per

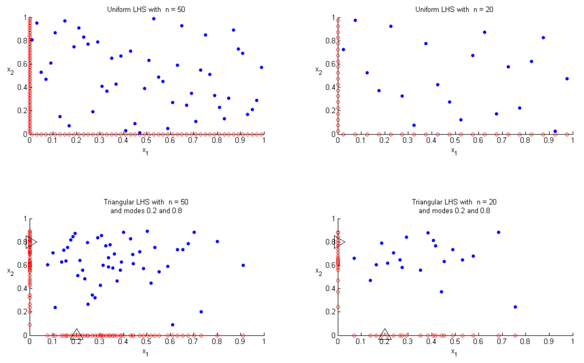

Figure 1: Effects of $n = 20$ versus $n = 50$ and $\mathrm{U}(0,1)$ versus $\mathrm{T}(0.2, 0.8)$ for LHS with midpoints

input has 10^k points). We assume that if $n \geq 10$, then we can estimate θ_j (with $j = 1$, ..., k) reasonably accurate, because LHS implies that projection of the n (k-dimensional) points onto the k individual axes gives n non-collapsing values per axis. Loeppky et al. (2009) recommends $n = 10k$ for LHS in Kriging aimed at sensitivity analysis (but our goal is prediction); this recommendation implies $n \geq 10$ if $k \geq 1$. Nevertheless, if LHS uses a "small" n ($= 10k$) and a "large" k, then LHS covers $[0, 1]^k$ sparsely (so there are only a few old points close to the new point) and the Kriging predictor is inadequate. Gramacy (2016) gives an example with $n = 50$ and $k = 2$, without further discussing the choice of n. We decide to experiment with several (n, k) combinations; namely, Gramacy's (n, k) $= (50, 2)$ and Loeppky et al.'s $(n, k) = (20, 2)$. Figure 1 shows empirical results; obviously, a higher n gives a denser coverage of the two-dimensional space; $\mathrm{U}(0, 1)$ gives points uniformly distributed over that space, whereas $\mathrm{T}(0.2, 0.8)$ clusters points around the point $(0.2, 0.8)$; the marginal distributions displayed on the two axes show that these distributions are indeed $\mathrm{U}(0, 1)$ and $\mathrm{T}(0.2, 0.8)$ where the modes are indicated by triangles on the two axes of the two lower panels.

3.3 LHS designs with triangular input distributions

"Big data" implies $n \ll N$ so we can use $10k < n \ll N$ I/O observations for the Kriging metamodel, which does not give numerical problems when computing $\mathbf{R}^{-1}$ and $|\mathbf{R}|$ (see Section 2). In case we use the Kriging metamodel in a real-time on-line *decision support system* (DSS), we may prefer $n = 10k$ to reduce computer time.

There is much *software* for LHS. For example, Microsoft's Excel spreadsheet software has add-ins that include LHS; see Oracle's Crystal Ball, Palisade's @Risk, and Frontline Systems' Risk Solver. LHS is also available in the MAT-

LAB Statistics toolbox, the R package, the Open TURNS software, and Sandia's DAKOTA software. We use MATLAB's "lhsdesign" to verify our own code for $\mathrm{U}(0,1)$.

Note: $\mathrm{U}(0,1)$ seems attractive if we are not interested in the prediction for one or more specific points, but in *sensitivity analysis* over the whole experimental area. Different goals of Kriging (and other types of metamodel) are discussed in Kleijnen (2017).

Kriging gives an inaccurate predictor $\widehat{y}(\mathbf{x}_0)$ in case of *extrapolation* (i.e., Kriging is meant for interpolation); see the discussion in Kleijnen (2015, p. 187). As we have already mentioned, our LHS assumes that the k inputs are independently distributed (so we do not have to select specific non-zero values for their correlation coefficients). Consequently, Algorithm 1 independently samples n values per input variable. To avoid extrapolation, we require $\mathbf{x}_0$ to lie inside the *convex hull* of the n points in $\mathbf{X}_{\mathrm{L}}$ (so $\mathbf{x}_0$ should not be a vertex of this hull or lie outside this hull). Actually we do not test $\mathbf{X}_{\mathrm{L}}$, but we test $\mathbf{X}_n$ which is the matrix with the $n < N$ simulated points $\mathbf{x}_i$ ($i = 1, ..., n$) that are the unique NN of $\mathbf{X}_{\mathrm{L}}$, as we shall see in Section 3.5. To check this "convex hull" condition, we can use one of the following two methods.

(i) We may formulate the *linear programming* (LP) problem

$$\begin{aligned} &\min_{a_i} \sum_{i=1}^{n} f_i a_i \\ &a(_i\mathbf{x}_i = \mathbf{x}_0 \\ &\sum_{i=1}^{n} a_i{=}1 \\ &a_i \geq 0 \end{aligned} \tag{14}$$

where we may specify any coefficients f_i (e.g., $f_i = 1$ or $f_i = 0$), because we want to know only whether the LP model has a *feasible* solution. To solve this problem, we use the MATLAB function "linprog", which may give an "exitflag" to explain why linprog stopped; if that flag is "-2", then "No feasible point was found".

(ii) The MATLAB function "convhulln($\mathbf{X}$)" returns the indices of the points in $\mathbf{X}$ that specify its convex hull (this function is based on Barber et al. (1996)). We require that $\mathbf{x}_0$ is not one of these points:

$$\mathrm{convhulln}(\mathbf{X}_n) = \mathrm{convhulln}(\mathbf{X}_n \cup \mathbf{x}_0) \tag{15}$$

Sub (i) and (ii): If $\mathbf{x}_0$ is not within the convex hull of $\mathbf{X}_n$, then we resample x_{LHS} which gives $\mathbf{X}_n$—until the hull condition is satisfied. Actually, we implement method (ii), and resample no more than 100 times to avoid unlimited computation time.

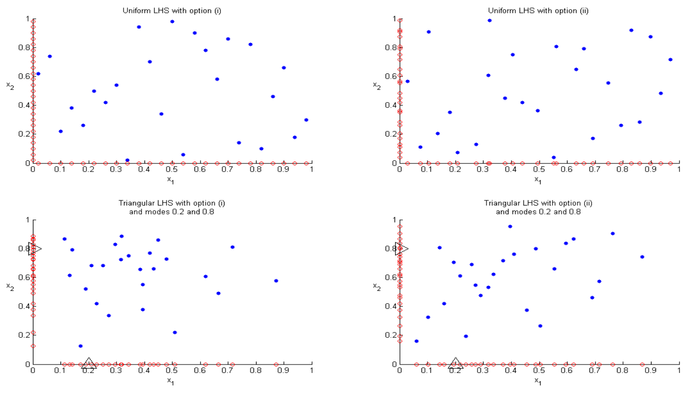

Figure 2: LHS with option (i) versus option (ii) for $\mathrm{U}(0,1)$ versus $\mathrm{T}(0.2, .0.8)$, given $n = 25$

Note: We impose the following condition for our LHS with option (ii) (sampling instead of midpoints):

$$\min_{1 \leq i \leq n} x_{n;\ i;j} < x_{0;j} < \max_{1 \leq i \leq n} x_{n;\ i;j} \ (j = 1, ..., k). \tag{16}$$

If n is "relatively" small, then option (i) (midpoints) cannot satisfy this condition; e.g. $\min_{1 \leq i \leq n} x_{n;\ i;j} = 1/(2n)$; e.g., if $n = 10$, then $\min_{1 \leq i \leq n} x_{n;\ i;j} = 1/20$ $= 0.05$, which may be higher than $x_{0;j}$ (likewise, $n = 10$ gives $\max_{1 \leq i \leq n} x_{n;\ i;j}$ $= 1 - 1/(2n) = 0.95$, which may be smaller than $x_{0;j}$). However, if $k > 1$, then the condition given in (16) is necessary but not sufficient.

3.4 LHS with uniform and triangular distributions

In the preceding subsection (especially (11) through (13)) we focused on the general CDF $F(x)$. Now we consider our two specific PDFs; namely $\mathrm{U}(0,1)$ and $\mathrm{T}(x_0)$. The CDF of $\mathrm{U}(0,1)$ is

$$F_{\mathrm{U};j}(x) = x \text{ if } 0 \leq x \leq 1. \tag{17}$$

This CDF and (12) imply that option (i) samples $\mathrm{U}(0,1)$ through

$$x_{j;g} = m_{g;j} \text{ if } l_{g;j} < x_{g;j} < \ h_{g;j} \ (g = 1, .., n)$$

We display a realization in the upper-left panel of Fig. 2. The CDF in (17) and the expression in (13) imply that option (ii) samples $\mathrm{U}(0,1)$ through the PRN $r \sim \mathrm{U}(0,1)$ and

$$x_{j;g} = l_{g;j} + r(h_{g;j} - l_{g;j}) \text{ if } l_{g;j} < x_{g;j} < \ h_{g;j} \ (g = 1, .., n)$$

This gives the upper-right panel of Fig. 2.

In risk analysis, LHS often assumes a specific *non-uniform* distribution for x_j. Inspired by Gramacy (2016) we consider $\mathbf{X}_{\mathrm{L}}$ with relatively many points close to the new point (to be predicted) $\mathbf{x}_0$; i.e., we replace $x_j \sim \mathrm{U}(0,1)$ by a continuous PDF with its mode at $x_{0;j}$ (and $0 \leq x \leq 1$). There are many PDFs that meet these requirements; see Law (2015, pp. 286–305). For example, *beta distributions* satisfy these requirements, provided we select the correct values for their (two) parameters; moreover, a different combination of these parameters gives a different variation around the mode; see Law (2015, pp. 295–297). We use a special case of the beta distributions; namely, a triangular distribution with its mode at x_0; we denote this distribution by $\mathrm{T}(x_0)$. The CDF of $\mathrm{T}(x_0)$ is (see Law (2015, pp. 304–305)):

$$F_{\mathrm{T};j}(x) = \frac{x^2}{x_{0;j}} \text{ if } 0 \leq x \leq x_{0;j} \tag{18}$$
$$F_{\mathrm{T};j}(x) = 1 - \frac{(1-x)^2}{1-x_{0;j}} \text{if } x_{;0j} \leq x \leq 1$$

which implies $F_{\mathrm{T};j}(0) = 0$, $F_{\mathrm{T};j}(1) = 1$, and $F_{\mathrm{T};j}(x_0) = x_{0;j}$,so this CDF has a *kink* at $x_{0;j}$. Combining this equation with (12), option (i) samples

$$x_{j;g} = \sqrt{\frac{x_{0;j}(2g-1)}{2n}} \text{ with } x_g \leq x_{0;j} \ (g = 1, ..., n) \tag{19}$$
$$x_{j;g} = 1 - \sqrt{\frac{(1-x_{0;j})[2n-(2g-1)]}{2n}} \text{ with } x_{j;g} \geq x_{0;j}$$

This gives the lower-left panel of Fig. 2 if $x_{0;1} = 0.2$ and $x_{0;2} = 0.8$; triangles on the axes denote $x_{0;1}$ and $x_{0;2}$. Option (ii) samples x_j (within the specific subinterval) via the first line of (18) if $h_{g;j} < x_{0;j}$. If $l_{g;j} > x_{0;j}$, then it samples x_j via the second line of (18). If $l_{g;j} < x_{0;j} < h_{g;j}$, then it first samples the PRN $r \sim \mathrm{U}(0,1)$; if $r < x_{0;j}$, then it samples x_j via the first line of (18); if $r > x_{0;j}$, then it samples x_j via the second line of (18). Altogether, this option samples

$$x_{j;g} = \sqrt{x_{0;j} r} \text{ if either } x_{0;j} > h_{g;j} \text{ or } l_{g;j} < x_{0;j} < h_{g;j} \text{ and } r < x_{0;j} \tag{20}$$
$$x_{j;g} = 1 - \sqrt{(1-x_{0;j})(1-r)} \text{ if either } x_{0;j} < l_{g;j} \text{ or } l_{g;j} < x_{0;j} < h_{g;j} \text{ and } r > x_{0;j}.$$

This gives the lower-right panel of Fig. 2.

3.5 Nearest neighbors in a "big data" set

Altogether we apply LHS with either uniform or triangular marginal PDFs to obtain the $n \times k$ design $\mathbf{X}_{\mathrm{L}}$. Next we select n points from the N points in $\mathbf{X}_N$ that are *closest* to the n points in $\mathbf{X}_{\mathrm{L}}$. We quantify this closeness through the Euclidean distance (say) d (besides this L_2 norm we use the L_1 norm $h_j =$

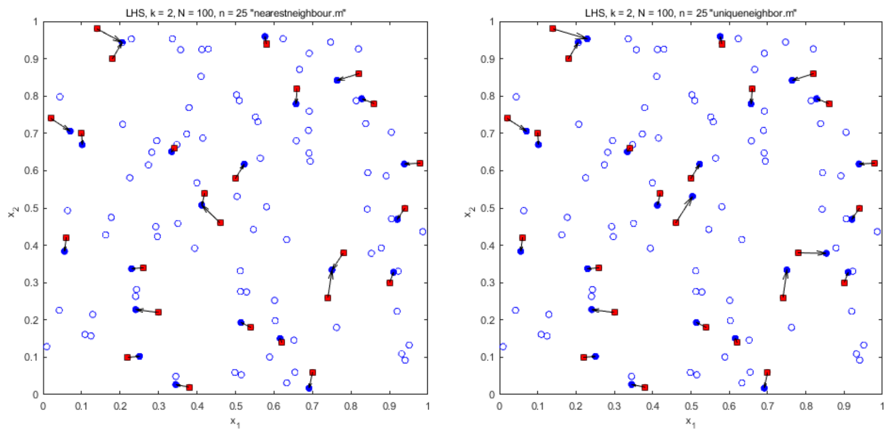

Figure 3: Brown's NN function versus our unique NN function; (blue) circles are big data, (red) squares are LHS data, and arrows point from LHS points to selected NN (colors blue and red are displayed only in PDF file)

$|x_{g;j} - x_{g';j}|$ below (6)). To solve the NN problem, we can choose among several algorithms; see

https://en.wikipedia.org/wiki/Nearest_neighbor_search (latest update 23 June 2017)

and also Guinness (2018).

Inspired by the MATLAB function "nearestneighbours.m"—developed by Richard Brown at Massey University in New Zealand—we develop our own MATLAB function "uniqueneighbor.m" (we use MATLAB throughout our investigation). Our function ensures that the n points in $\mathbf{X}_{\mathrm{L}}$ select n *unique* points in the big data set with N points. In this function we add the (constant) value 2 to each coordinate $x_{i;j}$ (with $0 \leq x_{i;j} \leq 1$) once a point $\mathbf{x}_i$ (with $i = 1$, ..., n) is selected. We illustrate the two MATLAB functions through an example with $k = 2$, $N = 100$, and $n = 25$ where we select the N points through uniform sampling and the n points through LHS option (ii) with uniform sampling. This example gives Fig. 3, where the results in the left-hand and right-hand panels are results of Brown's function and our function, respectively, and arrows point from a point in X_{L}—denoted by squares—to a point in X_N—denoted by circles (the PDF file uses the colors red and blue for X_{L} and X_N, respectively). This Figure shows that Brown's function implies that (for example) $(x_1, x_2) =$ (0.23550, 0.93305) in X_N is selected by two points in $\mathbf{X}_{\mathrm{L}}$, whereas our function does not.

Unfortunately, our method makes the final n points selected from $\mathbf{X}_N$ depend on the *order* in which we search the n points within $\mathbf{X}_{\mathrm{L}}$; see Fig. 4, which shows the results for forward selection (start with $i = 1$ and increase i until $i = n$) and backward selection (start with $i = n$ and decrease i until i =1). This Figure

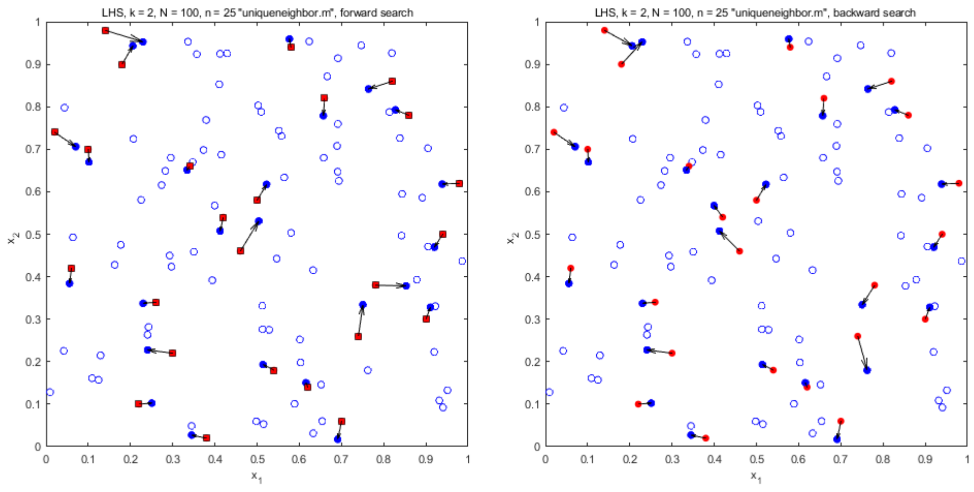

Figure 4: Forward versus backward selection of NN in our unique NN function; (blue) circles are big data, (red) squares are LHS data, and arrows point from LHS points to selected NN (colors blue and red are displayed only in PDF file)

shows that different starts give different selections for those points among the n points that select the same point among the N points when using Brown's function; see again Fig. 3 (left-hand panel). Which start gives the "optimal" n points among the N points is not obvious; i.e., we might estimate the variance of the Kriging predictor for each set of n points and select the set with minimum variance. However, in our experiments we limit our search to starting with $i = 1$.

The *triangular* distribution gives an $\mathbf{X}_n$ with more points close to $\mathbf{x}_0$ than the uniform distribution does; see Figure 5 where x_1 and x_2 have the modes 0.2 and 0.8, respectively. The question is whether the triangular distribution gives a "better" Kriging predictor; this question shall be addressed in Section 4.

4 Numerical experiments

To quantify the performance of alternative Kriging methods and designs, the literature often uses explicit mathematical functions with outputs (dependent variables) that can be quickly computed; these functions may have complicated response surfaces with known local minima. For details we refer to Dixon and Szego (1978), Floudas et al. (1999), Jamil and Yang (2013), Surjanovic and Bingham (2016), and Van Stein et al. (2017, p.14).

In Section 4.1 we discuss how we quantify the performance of the Kriging predictor. Next we present experimental results for the following four specific test functions. In Section 4.2 and Appendix 2 we use the example in Xiong

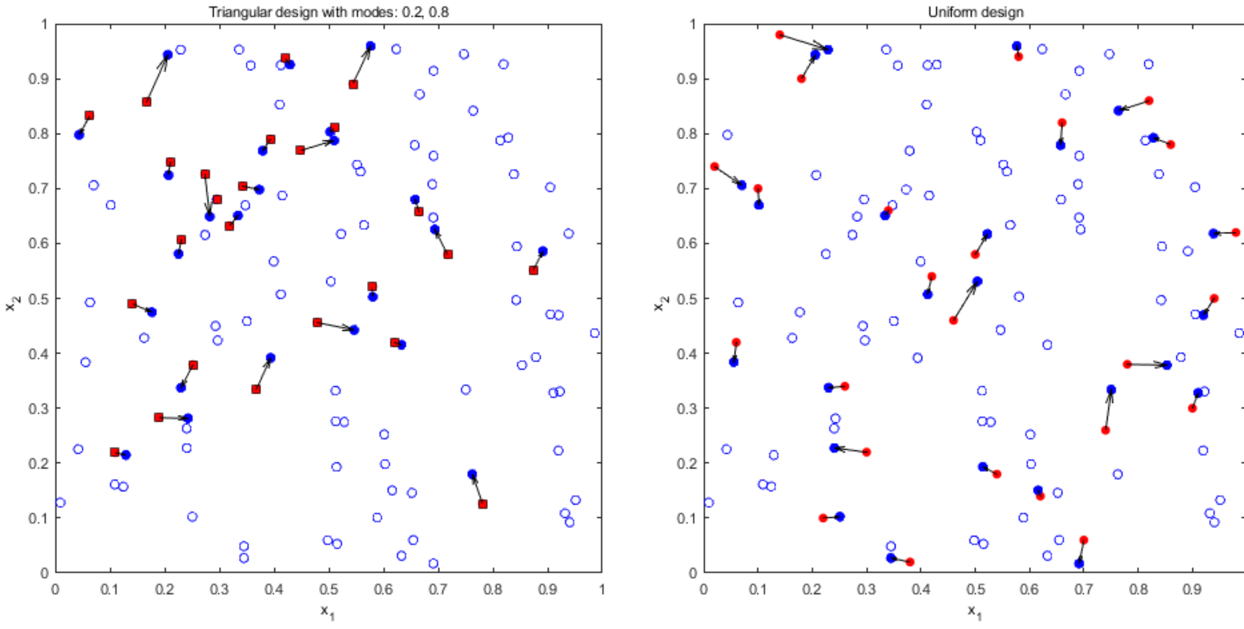

Figure 5: Unique NN for T$(0.2, 0.8)$ versus U$(0, 1)$, given $n = 25$ and N =100; (blue) circles are big data, (red) squares are LHS data, and arrows point from LHS points to selected NN (colors blue and red are displayed only in PDF file)

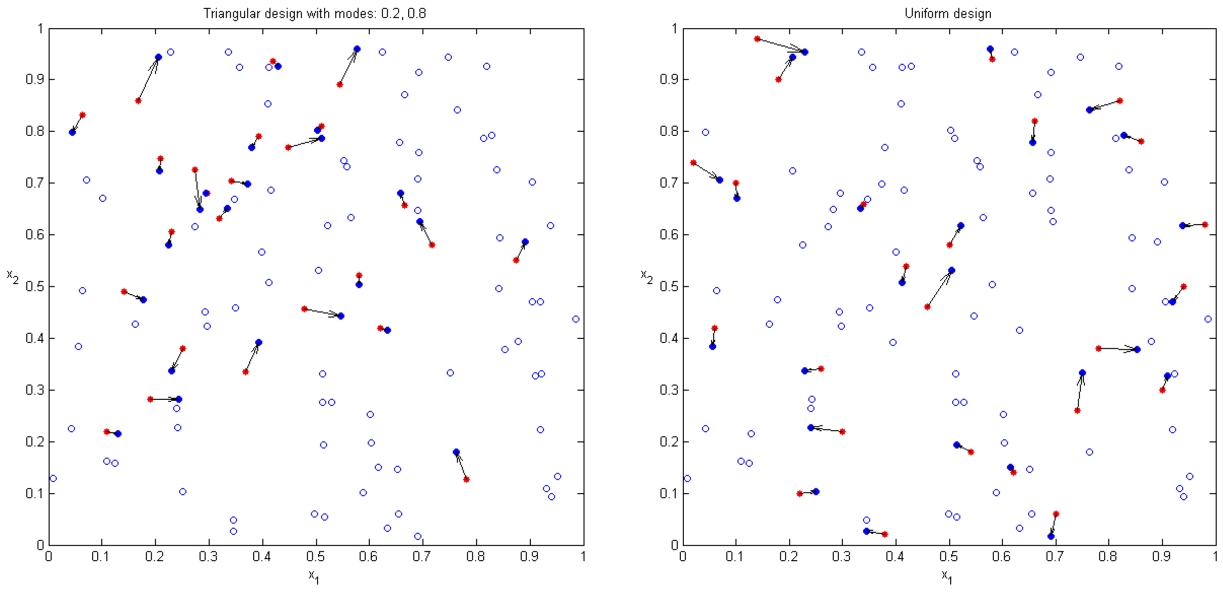

Figure 6: Unique NN for T$(0.2, 0.8)$ versus U$(0, 1)$, given $n = 25$ and N =100

et al.(2007) with $k =$ 1 input and many local optima; such an example with a single input makes it easy to plot the results of the experiment. In Section 4.3 and Appendix 3 we give another example with $k = 1$ input that is inspired by the M/M/1 model; this model is popular in management science/operations research (MS/OR); for readers not familiar with the M/M/1 model we define this model in Appendix 3. In Section 4.4 we give the example in Gramacy (2016) with $k = 2$ inputs and a multimodal output function. In Section 4.5 we give the popular borehole engineering example with $k = 8$ inputs.

In these examples we compare LHS with either midpoints of subintervals (option i) or sampling within subintervals (option ii)—see Section 3—and LHS with either the uniform PDF or triangular PDFs with modes at $\mathbf{x}_0$—see Section 3.4—so we have $2 \times 2 = 4$ experiments with LHS per example. In these four experiments we select the unique NNs of the n points sampled by LHS. These n points differ in different macroreplications, except for $k = 1$ and option (i) (using either T$(\mathbf{x}_0)$ or U$(0, 1)$). To evaluate the performance of Kriging, we select n_0 new points $\mathbf{x}_{0;t}$ $(t = 1, ..., n_0)$ using LHS with U$(0, 1)$ and option (ii) (sampling instead of midpoints); we define the $n_0 \times k$ test matrix $\mathbf{X}_0$.

Note: Our artificial examples are explicit mathematical functions that we can quickly evaluate for a small set of n_0 points in $\mathbf{X}_0$. In practical "computationally expensive" simulation models, the computation of the simulation output $w(\mathbf{x}_0)$ takes hours or days. Practical applications may also concern real-world data (instead of simulated data), so $w(\mathbf{x}_0)$ is available only if $\mathbf{x}_0 \in \mathbf{X}_N$. Big data implies that $n \ll N$, so we can use N - n points to test the validity of the Kriging model that we estimated from n points only. (If we had no "big data", then we might use cross-validation with only $n = N$ points, so we would delete one point and estimate the Kriging model from n - 1 points, etc.; see Kleijnen (2015).) We then select n_0 points through LHS with uniform PDFs for the k inputs (no midpoints), and use our NN function (defined in Section 3.5) to select the n_0 unique NN in $\mathbf{X}_N$; obviously, the resulting points in $\mathbf{X}_0$ are uniformly spread over $[0,1]^k$. In practice, the N points in $\mathbf{X}_N$ and $\mathbf{w}_N$ are given, and are the "true" I/O data. In our experiments we proceed as if we do not know $\mathbf{X}_0$ when we select $\mathbf{X}_n$; otherwise we would select $\mathbf{X}_0 = \mathbf{X}_n$.

A Kriging metamodel treats the simulation model as a *black box*; i.e., Kriging uses only the simulation model's I/O data $(\mathbf{x}_i, w_i)$ with $\mathbf{x}_i = (x_{i;1}, ..., x_{i;k})'$ and $i = 1, ..., N$. If $k = 1$, then we can present (x_i, w_i) as a scatterplot, and use this plot to detect a pattern in the I/O function. If $k > 1$, then it is harder to plot the N points; also see the plots in the next subsections.

4.1 Performance measures for experiments

To quantify the performance of Kriging with various designs in our numerical experiments, we use the MSPE because the Kriging predictor $\widehat{y}$ uses this criterion (as we mentioned above (3)). We might also use the square root of this MSPE; this monotonic transformation gives the same ranking—but decreases (increases, respectively) the magnitude of the difference between two MSPEs if the MSPEs are higher (smaller, respectively) than the value one—and may be used in a CI for $\widehat{y}$ (also see the CI in (10), which uses $s(\widehat{y})$, not $s^2(\widehat{y})$. We point out that MSPE defined in (5) assumes that the Kriging model is a valid metamodel of the simulation model. We, however, define the *prediction error* (PE) in (21) below.

Note: It would be inconsistent to use criteria such as the mean absolute error $\mathrm{E}(|\widehat{y}(\mathbf{x}_0) - w(\mathbf{x}_0)|)$, the mean absolute relative error $\mathrm{E}(\left|[\widehat{y}(\mathbf{x}_0, \widehat{\psi}) - w(\mathbf{x}_0)]/w(\mathbf{x}_0)\right|)$, or the coverage probability of the CI defined in (10). These criteria are also discussed in Kleijnen (2015, p. 120). Coverage is used in Guhaniyogi and Banerjee (2018, Fig. 2).)

More precisely, we define the *prediction error* (PE) as

$$e(\mathbf{x}_0, \mathbf{X}_n) = \widehat{y}(\mathbf{x}_0, \mathbf{X}_n) - w(\mathbf{x}_0), \tag{21}$$

where (unlike most publications on Kriging) we explicitly distinguish between the output $\widehat{y}$ of the estimated metamodel, and the output w of the simulation model. This PE gives the *squared prediction error* (SPE) $e^2(\mathbf{x}_0, \mathbf{X}_n)$. To decrease the effects of the randomness in $\mathbf{X}_n$ when using LHS, we obtain (say) $m >$ 1 *independently and identically distributed* (IID) *macroreplications* (m should not be confused with $m_{j;g}$). Macroreplication r gives $\mathbf{X}_{n;r}$ (the subscript r should not be confused with the PRN r). To estimate our performance measure $\mathrm{E}[e^2(\mathbf{x}_0, \mathbf{X}_n)]$, we compute the *average SPE* (ASPE) of these m macroreplications:

$$\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n) = \frac{\sum_{r=1}^{m} e^2(\mathbf{x}_0, \mathbf{X}_{n;r})}{m} = \overline{e^2}(\mathbf{x}_0, \mathbf{X}_n). \tag{22}$$

Note: These macroreplications use m non-overlapping PRN streams, while all other experimental factors remain the same; e.g., $\mathbf{X}_N$ and n do not change across macroreplications, while $\mathbf{X}_n$ does change as $\mathbf{X}_{\mathrm{L}}$ changes. To guarantee statistical independence, we can choose between the following two implementations: (i) generate m matrixes $\mathbf{X}_{\mathrm{L}}$, and store these matrixes; (ii) save the last PRN of macroreplication r = 1, and start the next macroreplication r + 1 with this PRN, etc.; MATLAB enables this option, as Kleijnen and Shi (2017) explains. Actually we choose implementation (i). We observe that all m macroreplications give the same result (so no randomness occurs) if k = 1 and LHS uses the midpoints $m_{j;g}$ for $\mathrm{T}(\mathbf{x}_0)$ or $\mathrm{U}(0,1)$.

To quantify the statistical accuracy of $\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)$ defined in (22), we compute its *standard error* (SE):

$$\mathrm{SE}(\mathbf{x}_0, \mathbf{X}_n) = \sqrt{\frac{\sum_{r=1}^{m}[e^2(\mathbf{x}_0, \mathbf{X}_{n;r}) - \overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)]^2}{(m-1)m}}. \tag{23}$$

Obviously, this SE decreases as m increases. Using $\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)$ and its $\mathrm{SE}(\mathbf{x}_0, \mathbf{X}_n)$—defined in (22) and (23), respectively—and the Student t-statistic with m - 1 degrees of freedom denoted by t_{m-1}, we compute the following $(1-\alpha)$ CI:

$$E[e^2(\mathbf{x}_0, \mathbf{X}_n)] \in \overline{e^2}(\mathbf{x}_0, \mathbf{X}_n) \pm t_{m-1;\alpha/2}\mathrm{SE}(\mathbf{x}_0, \mathbf{X}_n). \tag{24}$$

We can use this CI to compare the MSPEs for two different design types if one

type has no randomness (as is the case if k = 1 and the design uses midpoints) or if $k \geq 1$ and the design is given (e.g., the design is given by Gramacy (2016)'s sequential design).

We may compute $\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)$ for various $\mathbf{x}_{0;t}$. It may then turn out that $\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)$ is relatively high if the new point $\mathbf{x}_0$ is not surrounded by many old points. We may compute the *overall average* of all n_0 new points $\mathbf{x}_{0;t}$ in $\mathbf{X}_0$:

$$\overline{\overline{e^2}}(\mathbf{X}_0, \mathbf{X}_n) = \frac{\sum_{t=1}^{n_0} \overline{e^2}(\mathbf{x}_{0;t}, \mathbf{X}_{n;t})}{n_0} \tag{25}$$

where $\mathbf{X}_{n;t}$ is the design with n old points used to estimate the output for the new point $\mathbf{x}_{0;t}$.

Note: Actually, (25) is the square of the estimated *root mean squared prediction error* (RMSPE), which Gramacy (2015) uses. However, this RMSPE changes the measurement unit. Moreover, the SE of $\mathrm{RM\widehat{S}PE}$ is hard to derive.

The SE of $\overline{\overline{e^2}}(\mathbf{X}_0, \mathbf{X}_n)$ simply follows from (23) and (25):

$$\mathrm{SE}(\mathbf{X}_0, \mathbf{X}_n) = \sqrt{\frac{\sum_{t=1}^{n_0} \mathrm{SE}^2(\mathbf{x}_{0;t}, \mathbf{X}_{n;t})}{n_0^2}}, \tag{26}$$

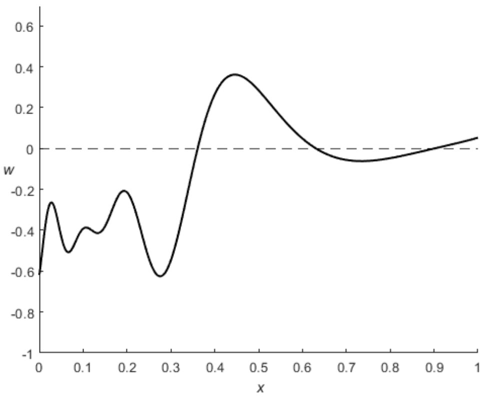

Figure 7: Xiong et al. (2007)'s example observed at 10,000 points

provided we do not use *common random numbers* for the n_0 points. Analogously to (24), we can use this $\mathrm{SE}(\mathbf{X}_0, \mathbf{X}_n)$ to compute the following $(1-\alpha)$ CI for $\mathrm{E}(\mathrm{S}\overline{\overline{\mathrm{P}}}\mathrm{E}(\mathbf{X}_0, \mathbf{X}_n))$:

$$\mathrm{E}(\overline{\overline{e^2}}(\mathbf{X}_0, \mathbf{X}_n) \in \overline{\overline{e^2}}(\mathbf{X}_0, \mathbf{X}_n) \pm z_{\alpha/2}\mathrm{SE}(\mathbf{X}_0, \mathbf{X}_n). \tag{27}$$

Note: Altogether, macroreplication r gives the LHS design matrix $\mathbf{X}_{\mathrm{L};r}$, which gives the NN matrix with $n \ll N$ simulated points $\mathbf{X}_{n;r} = (\mathbf{x}_{i;r})$ with $i = 1, ..., n$. For this $\mathbf{X}_{n;r}$ we compute the corresponding simulated outputs $\mathbf{w}_r = (w_{i;r})$ with $w_{i;r} = f(\mathbf{x}_{i;r})$ where f denotes the simulation I/O function. Using these $(\mathbf{X}_{n;r}, \mathbf{w}_r)$, DACE computes $\widehat{\psi}_r$. Using this $\widehat{\psi}_r$ and $(\mathbf{X}_{n;r}, \mathbf{w}_r)$, DACE computes $\widehat{y}(\mathbf{x}_0, \widehat{\psi}_r)$; see (8). Using these $\widehat{y}(\mathbf{x}_0, \widehat{\psi}_r)$ and $w(\mathbf{x}_0)$, we (not DACE) compute SPE_r. A different software package may give a different $\widehat{\psi}_r$ (see Section 2), which gives a different $\mathrm{S}\overline{\mathrm{P}}\mathrm{E}(\mathbf{x}_0, \mathbf{X}_n)$. We use rather old software; namely, DACE.

4.2 Xiong et al. (2007): one input and many local extrema

In this example our "big data" set is (x_i, w_i) with $N = 1{,}000$ uniformly distributed x_i; these (x_i, w_i) are plotted in Fig. 7.This plot looks like a "smooth" function (displayed in black in the PDF file). Actually, (x_i, w_i) is computed through Xiong et al. (2007)'s function (where we use w instead of Xiong et al.'s y):

$$w(x) = sin[30(x-0.9)^4]cos[2(x-0.9)] + \frac{x-0.9}{2} \text{ with } 0 \leq x \leq 1. \tag{28}$$

This function has both global and local extrema, and $\mathrm{E}[w(x)]$ with $0 \leq x \leq 0.4$ is much smaller than $\mathrm{E}[w(x)]$ with $0.4 \leq x \leq 1$.

In our experiment we have the following factors (the first three factors are controlled by the Kriging analysts, whereas the last factor is determined by the clients of these analysts):

- $\mathrm{T}(\mathbf{x}_0)$ versus $\mathrm{U}(0,1)$: we conjecture that $\mathrm{T}(\mathbf{x}_0)$ performs better (see Section 3.4, paragraph 2).

- Midpoints versus sampled points (within each of the n classes); we conjecture that midpoints perform better (see the text preceding (12)).

- Number of learning ("old") points; namely, n is either $10k = 10$ (see Loeppky et al. (2009)) or 100; we conjecture that a higher n gives better performance (to select n *specific* points, we use LHS).

- n_0 locations of $\mathbf{x}_0$; we conjecture that an "extreme" $\mathbf{x}_0$ performs worse (because Kriging is a bad extrapolator); we consider a fixed number of "new" points; namely, $n_0 = 10$. Because Fig. 7 shows more frequent oscillations when $0 \leq x \leq 0.3$, we select $n_0/2 = 5$ new points x_0 in $0.0 \leq x \leq 0.3$, so the other 5 points are within $0.3 \leq x \leq 1.0$.

Table 1 gives the *exact* values of the *average SPE* (ASPE) and their SEs (in parentheses); some values are displayed after multiplying them with 10^c—where c depends on the magnitudes of the results—so it is easier to compare results across rows (with different x_0). Appendix 2 gives Fig.16, which displays *box-and-whisker plots* for the estimated SPE if not using midpoints, but sample points within each of the n classes; we use the MATLAB function "boxplot" with the default whisker length of at most 1.5 times the interquartile range. We present the following conclusions.

(i) The bottom line of Table 1 clearly shows that the *overall* ASPE (defined in (25)) is minimal for $\mathrm{T}(\mathbf{x}_0)$ with midpoints. We prefer this measure because the analysts do not know which new points will need to be predicted.

(ii) Table 1 clearly shows that if we use $\mathrm{T}(\mathbf{x}_0)$ with midpoints and $0 \leq x_0 \leq 0.3$, then an individual new point x_0 gives an $\overline{e^2}(x_0, \mathbf{X}_n)$ that is relatively high because the I/O function wiggles in this subarea); if we use $\mathrm{U}(0,1)$ (which does not depend on x_0), then the behavior of $\overline{e^2}(x_0, \mathbf{X}_n)$ is more irregular.

(iii) Comparing $\overline{e^2}(x_0, \mathbf{X}_n)$ for $n = 10$ and $n = 100$ in Table 1, we conclude that the analysts should use the highest possible value for n ($\ll N$)—given the computer's memory size (the required size is of order n^2 because of the need to store the $n \times n$ matrixes $\widehat{\mathbf{\Sigma}}_M$ and $\widehat{\mathbf{\Sigma}}_M^{-1}$) and speed (which is of order n^3, because of the need to compute $\widehat{\psi}$, $\widehat{y}(\mathbf{x}_0, \widehat{\psi})$, and $s^2(\mathbf{x}_0, \widehat{\psi})$).

(iv) If we use sampled points (instead of midpoints), then the box-and-whisker plots in Appendix 2 clearly show that the statistical distribution of $e^2(x_0, \mathbf{X}_n)$ has a long right-hand tail, especially if $0 \leq x_0 \leq 0.3$ (this tail implies that relatively high $e^2(x_0, \mathbf{X}_n)$ occur, which we expect because the $e^2(x_0, \mathbf{X}_n)$ is sensitive to outliers).

$n = 10$						
x_0	T(x_0)			U$(0,1)$		
	midpoints	sampled points		midpoints	sampled points	
	$\times 10^{-3}$	$\times 10^{-3}$	$(\times 10^{-3})$			$(\times 10^{-3})$
0.0579	14.173250	387.259219	(54.259719)	0.002051	0.326645	(67.192776)
0.1197	13.405614	62.247749	(12.012896)	0.000823	0.326670	(81.643325)
0.1675	1.132151	9.712443	(1.841269)	0.014015	0.040278	(7.966081)
0.2178	0.087451	0.156941	(0.031959)	0.051322	0.073233	(10.264948)
0.2590	0.010744	0.139421	(0.018511)	0.002499	0.021036	(2.600499)
0.3183	0.001152	0.078622	(0.019618)	0.009453	0.013668	(2.167676)
0.5274	0.000024	0.000468	(0.000072)	0.000253	0.000971	(0.267078)
0.6182	0.000006	0.000076	(0.000019)	0.000290	0.000721	(0.180870)
0.7397	0.000000	0.000021	(0.000006)	0.000070	0.001750	(0.412430)
0.9045	0.000350	0.000734	(0.000168)	0.020472	0.045283	(11.586822)
Overall	2.8811E-03	45.9596E-03	(5.5604E-03)	0.010125	0.085026	(10.7216E-03)
$n = 100$						
	$\times 10^{-3}$	$\times 10^{-3}$	$(\times 10^{-3})$			$(\times 10^{-3})$
0.0579	0.1217324	0.874386	(0.018490)	0.006790	0.006489	(0.113602)
0.1197	0.1192415	0.405315	(0.012380)	0.000694	0.000633	(0.026817)
0.1675	0.0778101	0.221054	(0.004841)	0.002444	0.002353	(0.042927)
0.2178	0.0045075	0.127653	(0.004020)	0.003927	0.003784	(0.043876)
0.2590	0.0022729	0.127987	(0.004991)	0.001551	0.001321	(0.019813)
0.3183	0.0001789	0.036557	(0.001663)	0.000016	0.000042	(0.002896)
0.5274	0.0000084	0.000060	(0.000007)	0.000253	0.000222	(0.004266)
0.6182	0.0004973	0.002232	(0.000069)	0.000116	0.000108	(0.002508)
0.7397	0.0011572	0.000017	(0.000002)	0.000009	0.000008	(0.000453)
0.9045	0.0000128	0.000311	(0.000015)	0.000020	0.000032	(0.000742)
Overall	0.0327E-03	0.1796E-03	(2.3715E-06)	0.001582	0.001499	(1.3349E-05)

Table 1: Average SPE (ASPE) and its SE in parentheses, for Xiong et al.'s example

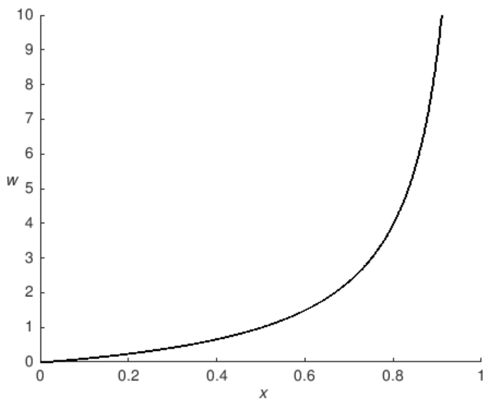

Figure 8: M/M/1–inspired example observed at 10,000 points

4.3 "M/M/1": one input and "exploding" output

In Appendix 3 we define the M/M/1 model. The typical output of this model is the steady-state mean waiting-time. We do not use discrete-event simulation of this model, because such a simulation requires both a Kriging metamodel that accounts for the so-called intrinsic noise besides the extrinsic noise $M(\mathbf{x})$ in (1), and a simulation with a correct choice of the starting state (e.g., the "empty" state, which has no customers waiting) and ending state (e.g., enough customers are simulated to reach the steady state). Instead—inspired by the analytical solution of this model—we define

$$w(x) = \frac{x}{1-x} \text{ with } 0 < x < 1. \tag{29}$$

As we did for Xiong et al. (2007)'s example, we select N =1,000 and $n = 10$ old points. Fig. 8 with (x_i, w_i) and $i = 1, ..., 1{,}000$ shows that $w(x)$ is a *monotonic* function; when x exceeds (say) 0.6, then $w(x)$ strongly increases, and when x approaches 1, then $w(x)$ *explodes.* Therefore, we select $n_0/2$ new points in the interval $0.2 < x_0 < 0.6$ and $n_0/2$ new points in $0.6 < x_0 < 0.8$.

Table 2 gives *average SPE*s (ASPEs) and SEs in parentheses; Appendix 3 gives Fig.17 with box-and-whisker plots for these ASPEs if using sampled points. We present the following conclusions, which closely resemble our conclusions for the preceding example.

(i) The bottom line of Table 2 clearly shows that the *overall* ASPE is minimal for $\mathrm{T}(\mathbf{x}_0)$ with midpoints.

(ii) Table 2 clearly shows that if we use n =10 or n = 100 old points and $\mathrm{T}(\mathbf{x}_0)$ with midpoints, then the $n_0 =$ 10 new points $x_{0;t}$ (t = 1, ..., 10) show nonmonotonic behavior of ASPE—even though $w(x)$ is a monotonic function.

(iii) Comparing ASPEs for n = 10 and n = 100 clearly shows that the analysts should use the highest possible value for n ($\ll N$) (even though—

$n = 10$						
x_0	T(x_0)			U$(0,1)$		
	midpoints	sampled points		midpoints	sampled points	
	$\times 10^{-6}$	$\times 10^{-6}$	$(\times 10^{-5})$			
0.2196	0.000129	0.000861	(0.000027)	0.000282	0.049780	(0.027417)
0.3244	0.000058	0.000655	(0.000016)	0.000056	0.003158	(0.002156)
0.4008	0.000141	0.002913	(0.000145)	0.000074	0.063410	(0.060149)
0.4683	0.000090	0.001338	(0.000024)	0.000021	0.019794	(0.014937)
0.5433	0.000009	0.006722	(0.000197)	0.000004	0.172797	(0.116642)
0.6154	0.000801	0.051510	(0.002015)	0.000158	0.008889	(0.003881)
0.6721	0.002392	0.137272	(0.003087)	0.000216	0.146564	(0.121277)
0.7100	0.023656	1.411934	(0.056152)	0.000908	0.019522	(0.009742)
0.7460	0.002326	1.766289	(0.053381)	0.000023	0.265368	(0.163860)
0.7811	0.510245	15.515917	(0.620607)	0.004364	0.740905	(0.559372)
Overall	0.0540E-06	1.8895E-06	(6.2544E-07)	0.000611	0.149019	(0.061055)
$n = 100$						
	$\times 10^{-6}$	$\times 10^{-6}$	$(\times 10^{-4})$			
0.2196	0.000278	0.009585	(0.000026)	0.041164	1.502790	(0.492937)
0.3244	0.001267	0.011808	(0.000062)	0.085950	1.310875	(0.307199)
0.4008	0.001557	0.014305	(0.000048)	0.000096	0.154568	(0.052656)
0.4683	0.011091	0.094452	(0.000348)	0.208198	13.016071	(2.615860)
0.5433	0.000000	0.242660	(0.001222)	1.354021	21.034320	(5.157925)
0.6154	0.000000	1.703808	(0.006742)	1.788633	48.685013	(10.524255)
0.6721	0.000010	7.294086	(0.019636)	0.508395	3.765382	(0.938038)
0.7100	0.000000	16.521462	(0.038523)	3.319421	62.947899	(14.680000)
0.7460	0.000641	21.705865	(0.037815)	0.049484	0.955395	(0.315473)
0.7811	0.003736	150.437159	(0.399877)	5.722450	129.885922	(27.381689)
Overall	0.0019E-06	19.8035E-06	(4.0404E-06)	1.307781	28.325824	(3.332845)

Table 2: Average SPE (ASPE) and its SE in parentheses, for M/M/1-inspired example

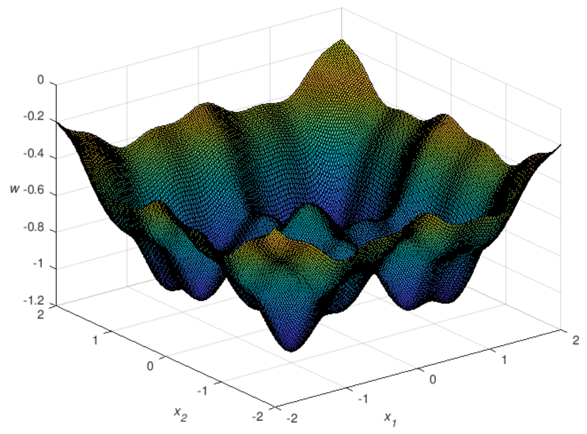

Figure 9: Gramacy (2015)' s example observed at 40,401 points

compared with n = 10—n = 100 gives a higher ASPE for the smallest x_0—namely 0.2196).

(iv) If we use sampled points for either $\mathrm{T}(x_0)$ or $\mathrm{U}(0, 1)$, then the box-and-whisker plots in Appendix 3 clearly show that the distribution of ASPEs may have a long right-hand tail.

4.4 Gramacy (2016): two inputs and multimodal output

Following Gramacy (2016), we plot $(\mathbf{z}_i, w_i)$ with $\mathbf{z}_i = (z_{i;1}, z_{i;2})$, $i = 1, ..., N$, and N = 40,401 defined by a 201 × 201 grid for $(z_{i;1}, z_{i;2})$ with increments of size 0.02; see Fig. 9. Obviously, this plot has many hilltops and fast-moving changes. Actually, $(\mathbf{z}_i, w_i)$ (i = 1, ..., 40,401) in this plot follow from

$$w(z_1, z_2) = -f(z_1)f(z_2) \text{ with } -2 \leq z_1, z_2 \leq 2 \text{ and}$$
$$f(z_j) = e^{-(z_j-1)^2} + e^{-0.8(z_j+1)^2} - 0.05\sin(8(z_j + 0.1)) \text{ with } j = 1,\ 2. \quad (30)$$

We observe that z_j (j = 1,2) is not standardized; the linear transformation x_j

= 0.5 + 0.25z_j makes x_j standardized such that $0 \leq x_j \leq 1$.

Gramacy (2016) searches for a small design $\mathbf{Z}_n$ for the new point $\mathbf{z}_0 = (-1.725, 1.725)'$ or $\mathbf{x}_0 = ((0.06875, 0.93125)'$, which lies in the upper-left corner of the standardized experimental area $[0.1]^2$. This search starts with $n_0 =\ 6$ NNs of this $\mathbf{z}_0$. The search uses one of three criteria, which gives three selections for $\mathbf{Z}_n$; the last two criteria are simplifications of its predecessor, in order to save computer time when searching for $\mathbf{Z}_n$; so we expect the RMSPE to increase as the criterion becomes more simplified. The search is stopped when n = 50. It turns out that each of these three selections of $\mathbf{Z}_n$ contains a cluster of points

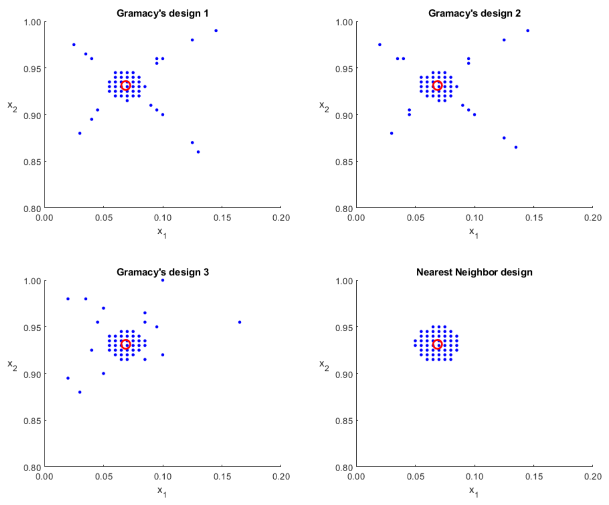

Figure 10: Gramacy (2015) example: Gramacy's three designs and our NN design

around $\mathbf{z}_0$ plus points along rays from $\mathbf{z}_0$; all n points are located in a relatively small subspace of the space formed by the N point; see Fig. 10 where the plot in the lower-right corner will be discussed in Section 5.1. This Figure uses Table 4 in Appendix 4, where the first six NNs of this $\mathbf{z}_0$ are displayed in the first six rows. This search stops when $n = 50$. Three different criteria give three selections for $\mathbf{Z}_n$. It turns out that each of these three selections of $\mathbf{Z}_n$ contains a cluster of points around $\mathbf{z}_0$ plus points along rays from $\mathbf{z}_0$; all n points are located in a relatively small subspace of the space formed by the N points; also see Gramacy (2016, Figs. 2 and 8) and Table 4 in Appendix 4, which shows the order in which the $n = 50$ points of the three designs are selected. (We give these exact values, for the sake of reproducible experimentation; also see Uhrmacher et al. (2016)).

The search for $\mathbf{Z}_n$ starts with n_0 NNs and $\theta = 0.1$. Next it selects the following point using one of the three criteria. The predictor $\widehat{y}(\mathbf{z}_0)$ uses the MLE $\widehat{\theta}$ estimated from the I/O data set with n observations. This $\widehat{\theta}$ changes with $\mathbf{z}_0$, because changing $\mathbf{z}_0$ changes $\mathbf{Z}_n$. This change mitigates the *stationary* assumption of the Kriging model, so $\mathbf{Z}_n$ may improve the predictor compared with $\mathbf{Z}_N$. (Chilès and Desassis (2018), Martinez-Cantin (2016), and Pronzato and Rendas (2017) also discuss this stationarity assumption, albeit in a Bayesian framework.)

Unlike Gramacy (2016), we select $\mathbf{X}_n$ through one-shot LHS (instead of a sequential design) with $\mathrm{T}(\mathbf{x}_0)$ and midpoints; i.e., given the experimental results for the preceding two examples we assume that $\mathrm{U}(0, 1)$ or sampled points are inferior. Our $\mathbf{X}_n$ does not depend on the specific simulation model that

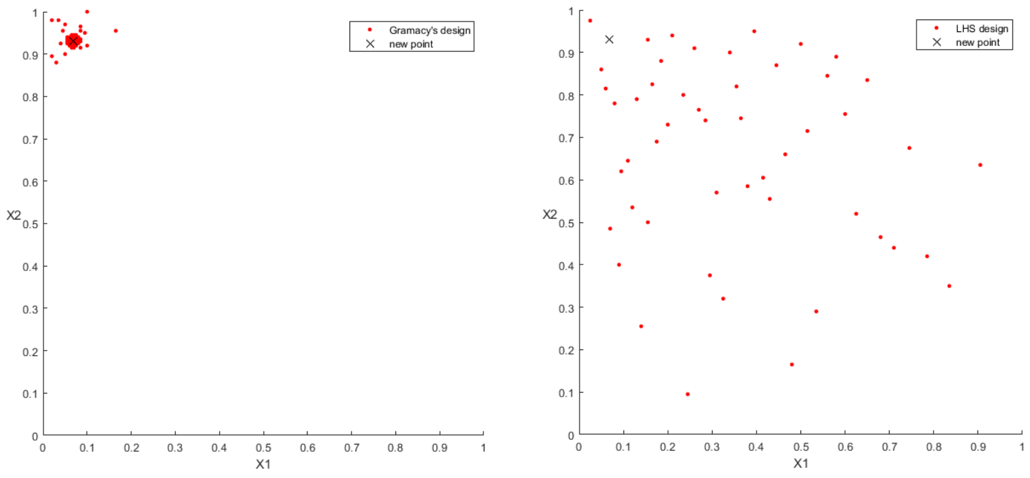

Figure 11: Gramacy's design #3 and macroreplication 1 of our LHS design with triangular distributions and midpoints

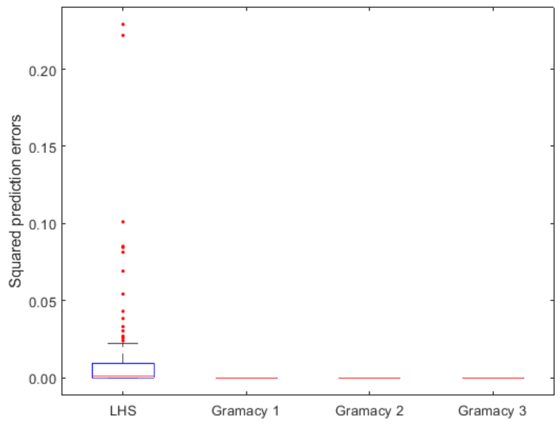

Figure 12: Boxplots for SPE in LHS design with 100 macroreplications, and in Gramacy (2015)'s three designs for Gramacy's two-dimensional example

is approximated by a Kriging metamodel (whereas Gramacy's design is "customized"). Fig. 11 displays a plot for Gramacy's design 3 (specified in Appendix 4 that uses the simplest criterion) and our design in macroreplication 1 (remember that $k > 1$ gives different LHS designs, even if midpoints are used; see Algorithm 1). Gramacy's $\mathbf{Z}_n$ and our $\mathbf{X}_n$ enable us to compute the corresponding output $\mathbf{w} = (w_i)'(i = 1, ..., n)$, so we can compute the estimated Kriging hyperparameters $\widehat{\psi}$ and $\widehat{\psi}_r$ $(r = 1, ..., m)$, respectively. For this computation we use DACE (remember that different $\widehat{\psi}$ may result from different software packages or from initializing the same package differently; see the text below (7)). The SPE of the Kriging predictor for the new point $\mathbf{z}_0 = (-1.725, 1.725)'$ or $\mathbf{x}_0 = ((0.06875, 0.93125)'$ depends not only on $\widehat{\psi}$ or $\widehat{\psi}_r$, but also on the n old points ($\mathbf{Z}_n$ or $\mathbf{X}_n$). The resulting SPEs are given in Fig. 12, which displays box-and-whisker plots. We present the following conclusions.

(i) The sample median $\text{SPE}_{(0.5m)}$ of the $m = 100$ SPEs in our design is 0.001453, whereas the SPEs of Gramacy design 1, 2, and 3—denoted by SPE_{Gram}—are 0.000005597, 0.000005598, and 0.000005597, so our design clearly gives a significantly higher SPE; we test this significance through the sign test. (Compared with the sample mean, the sample median is a more robust estimator of the mean; the ASPE $\overline{e^2}(\mathbf{x}_0, \mathbf{X}_n)$ is 0.01402649 with a SE of 0.00363959, so $\overline{e^2}(\mathbf{X}_0, \mathbf{X}_n)$ is significantly higher than Gramacy's SPEs.)

(ii) The boxplot shows that our design may give very high outliers (hence, $\overline{\overline{e^2}}(\mathbf{X}_0, \mathbf{X}_n) \gg \text{SPE}_{(0.5m)}$; see (i)).

(iii) Seven of our 100 SPEs (not displayed) are smaller than Gramacy's SPEs; e.g., our lowest SPE is 0.00000001 ($\ll$ 0.000005597).

(iv) In general, an estimated difference (like $\text{SPE}_{(0.5m)}$ - SPE_{Gram})) may be statistically significant, but practically unimportant. However, $\sqrt{\text{SPE}_{(0.5m)}}/\sqrt{\text{SPE}_{\text{Gram}}}$ is roughly as high as 16 (the square root gives the same measurement unit as the unit used by $\widehat{y}$). In practice, we must decide whether we accept such a big relative increase of the SPE caused by our LHS design.

In the next example, we shall examine whether our conclusions also hold if there are more inputs (namely, eight instead of two) and more new input combinations $\mathbf{x}_0$ (namely, ten instead of a single—rather extremely located—combination). Moreover, in Section 5 we shall present a NN design as an alternative for our LHS design.

4.5 Borehole: eight inputs

In the real (non-simulated) world, a "borehole" is defined in Wikipedia

(https://en.wikipedia.org/wiki/Borehole)

as "a narrow shaft bored in the ground ... for the extraction of water, other liquids (such as petroleum) or gases (such as natural gas)". An analytical model of a borehole is used in many publications on simulation methodology; e.g., Erickson et al. (2017), Gramacy (2016), Gramacy and Apley (2015), and Santner et al. (2018. p. 222). This model has the output "water flow rate" w

Name of original input z	Symbol	Unit	Range $[l, u]$
Radius of borehole	r_w	m	0.05, 0.15
Radius of influence	r	m	100, 50000
Transmissivity of upper aquifer	T_u	m^2/yr	63070, 115600
Potentiometric head of upper aquifer	H_u	m	990, 1110
Transmissivity of lower aquifer	T_l	m^2/yr	63.1, 116
Potentiometric head of lower aquifer	H_l	m	700, 820
Length of borehole	L	m	1120, 1680
Hydraulic conductivity of borehole	K_w	m/yr	9855, 12045

Table 3: Borehole inputs

(measured in m^3/yr) and the $k = 8$ original inputs—which we denote by the general symbol z_j (with $j = 1, ..., 8$)—and their ranges $[l_j, u_j]$ listed in Table 3 (the range of one of these inputs may be changed to create a more nonlinear and non-additive function, as Xiong et al. (2013) does):

$$w = \frac{2\pi T_u(H_u - H_l)}{\ln(r/r_w)(1 + \frac{2LT_u}{\ln(r/r_w)r_w^2 K_w} + \frac{T_u}{T_l})} \tag{31}$$

This model is coded in R and MATLAB; see
https://www.sfu.ca/~ssurjano/borehole.html

To obtain the standardized inputs $0 \leq x_j \leq 1$, we use Table 3 with $l_j \leq z_j \leq h_j$ and apply the linear transformations $x_j = a_j + b_j z_j$ with

$$a_j = \frac{1}{2} - \frac{(u_j + l_j)/2}{(u_j - l_j)} \text{ and } b_j = \frac{1}{u_j - l_j},$$

so $z_j = (x_j - a_j)/b_j = x_j/b_j - a_j/b_j$; i.e., we apply

$$z_j = l_j + (u_j - l_j)x_j \text{ or } x_j = \frac{z_j - l_j}{u_j - l_j} \tag{32}$$

so $x_j = 0$ implies $z_j = l_j$ and $x_j = 1$ implies $z_j = u_j$.

Whereas Gramacy (2016) uses N = 100,000 and $n = 50$ or $n = 200$, we now use N = 1,000,000 and $n = 60$—for each of the $n_0 = 10$ new locations $\mathbf{x}_{0;t}$ (with $t = 1, ..., n_0 = 10$). To sequentially select $\mathbf{X}_n$, only one of the three criteria—namely, the so-called ALC criterion—is used; we can provide interested readers with the n_0 =10 Excel sheets, each with $n \times k = 60 \times 8$ numbers that we received from Gramacy in personal communication. We use MATLAB's "lhsdesign" to sample N = 1,000,000 old combinations, so there is zero probability of a new combination $\mathbf{x}_{0;t}$ coinciding with one of the N = 1,000,000 old combinations (as we confirmed empirically via MATLAB's function "intersect").

Note: Gramacy (2016, p. 20) samples $N + n_0$ combinations through LHS with U(0,1) and random sampling within the subintervals, using R's function randomLHS. Unlike MATLAB's "lhsdesign", R's "randomLHS" does not try to

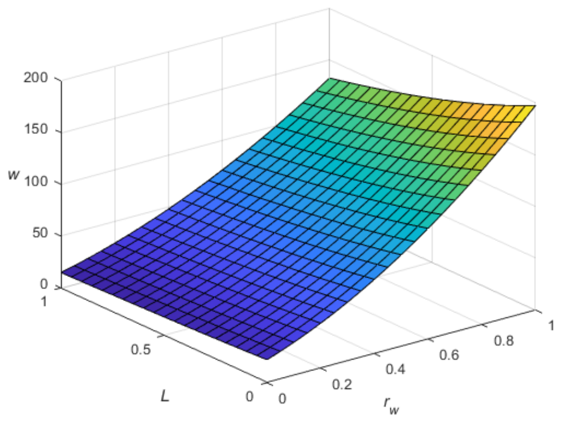

Figure 13: Borehole example: I/O function with 21×21 standardized input combinations (r_w, L) and output w

optimize the design through a criterion such as maximin, so it functions like "lhsdesign" with $M = 1$ (instead of the default $M = 5$; see Section 3.2). Sampling $N + n_0$combinations guarantees that none of these n_0 locations coincides with any of these N locations.

Note: Our N = 1,000,000 combinations are not identical to Gramacy's N = 1,000,000 combinations, because these combinations are sampled through MATLAB's "lhsdesign" and R's "randomLHS", respectively. However, the two resulting sets $\mathbf{X}_N$ have the same statistical distribution, so we ignore the difference between these two sampled sets.

Whereas the preceding examples have $k = 1$ or $k = 2$ inputs, we now have $k = 8$ inputs so it is hard to plot I/O points. Actually we make 3D plots like Fig. 9; i.e., we plot w_i (i = 1, ..., N) versus two inputs while keeping the other six inputs constant at the midpoints of their ranges in the experiment; see $[l, u]$ in Table 3. Santner et al. (2018, Fig 7.10 and Table 7.11) shows that r_w is the most important input, and the three inputs L, H_l, and H_u appear to have approximately equally important effects (quantified through their estimated main effects and total sensitivity indices). Therefore we make three plots; namely, w versus (r_w, L), (r_w, H_l), and (r_w, H_u), respectively. These three plots look very similar, so we present a single plot; namely, Fig. 13. To generate this plot, we use a grid of 21×21 input combinations (r_w, L) while keeping the other six inputs constant (finer grids give very similar plots); the plot uses standardized input values. These input combinations give outputs w that follow from the analytical model (31). The plot confirms that r_w is the most important input. Moreover, the plot suggests a *monotonic* I/O function; i.e., the borehole model looks complicated, but gives a simpler I/O function than Xiong et al.'s example in Section 4.2 and Gramacy's example in 4.4.

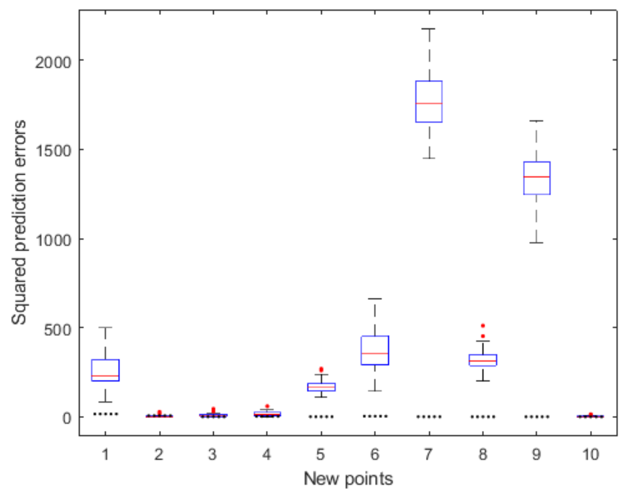

Figure 14: Boxplots for SPE in n_0 = 10 LHS designs with 25 macroreplications, and SPE in Gramacy (2015)'s n_0 = 10 designs for the borehole example

We do not know whether Gramacy's design for the borehole example with k = 8 gives a specific pattern that resembles the pattern with rays in Fig. 10 for the example with k = 2.

To save considerable computer time, we use only m = 25 macroreplications for each of the n_0 = 10 new points for the borehole example (instead of 100, as we did for the preceding examples; most computer time is needed to search for the NN among as many as N = 1,000,000 points in the borehole example). It turns out that m = 25 macroreplications give enough information; i.e., the signal/noise ratio is strong enough. The five horizontal dots in Fig. 14 display the SPEs for Gramacy's n_0 =10 designs, and the boxplots show the SPEs in the m = 25 macroreplications of our design sampled from $\mathrm{T}(\mathbf{x}_{0;t})$. We present the following conclusions.

(i) For six of the ten new points, our design gives much higher SPEs than Gramacy's design does; no further statistical analysis is needed.

(ii) For three other new points, our design may give SPEs that are lower than the SPE of Gramacy's design for the specific new point. More specifically, for the second new point, 22 of the 25 macroreplications for our design give smaller SPEs than Gramacy's design does. For the third new point, 2 macroreplications give smaller SPEs. For the fourth new point, 3 macroreplications give smaller SPEs. Note that for the tenth new point, no macroreplication gives a smaller SPE.

(iii) The overall ASPE *computed from* all ten new points is 426.650537 for LHS with $\mathrm{T}(\mathbf{x}_{0;t})$, 2.785240 for Gramacy's design, and 1.556274 for our NN design.

We conclude that our type of design may give relatively high SPEs, so we do not recommend this type. Therefore we consider the next type of one-shot design, as an alternative for Gramacy's sequential design.

5 One-shot design with n nearest neighbors

As we described above, Gramacy (2016) selects a "small" design matrix $\mathbf{X}_n$, which specifies n points from the "big" input data matrix $\mathbf{X}_N$ with $n \ll N$. Furthermore, this selection starts with a "relatively small" subset with $n_0 < n$ unique NNs of $\mathbf{x}_0$, and selects the next (n - n_0) points sequentially. It turns out that most points in $\mathbf{X}_n$ are close to $\mathbf{x}_0$. In the present section we select n unique NNs of $\mathbf{x}_0$ in "one shot", using the MATLAB function "nearestneighbours.m" (this function selects the same n NNs of the single new combination $\mathbf{x}_0$ as our own MATLAB function "neighbor.m", which we developed for selecting the n unique NNs of the $n > 1$ points in $\mathbf{X}_\mathrm{L}$—see Section 3.5—and takes more computer time than "nearestneighbours.m" does). Obviously, we do not need $m > 1$ macroreplications for our design.

5.1 Gramacy (2016)'s example revisited

We return to Gramacy (2016)'s example with N = 40,401 and n = 50, for one specific new input combination; namely, $\mathbf{x}_0 = (0.06875, 0.93125)'$ (which lies in the upper-left corner of the area of interest $[0.1]^2$); see again Section 4.4. Fig. 10 displays Gramacy's three designs and our design. Gramacy's designs start with n_0 = 6 NN, and sequentially add combinations that turn out to be either NNs or combinations close to the four rays (see again Section 4.4); by definition, our design selects only NNs. Note that in this Figure the x-axis and the y-axis extend only from 0.00 to 0.20 (not 1.00) and from 0.80 (not 0.00) to 1.00, respectively (because $\mathbf{x}_0 = (0.06875, 0.93125)'$).

Note: After we have selected our NN design, we—like Gramacy—use OK with the Gaussian correlation function to predict he output for a new input combination.

These four designs give the following SPEs for Gramacy's three designs and our design: 2.5246×10^{-15}, 2.6565×10^{-14}, 2.6535×10^{-15}, and 5.8423×10^{-13}; i.e., these SPEs turn out to be virtually zero. These SPEs are so small because w_0 = -0.372451 and $\widehat{y}_0$ = -0.372451 for the three Gramacy's designs and -0.372450 for our design.

5.2 The borehole example revisited

We return to the borehole example in Section 4.5, which uses N = 1,000,000 and only one of Gramacy (2016)'s designs with n = 60 for ten new locations in the input space with k = 8 dimensions. Fig. 15 displays the SPEs for Gramacy's design and our design with the same n (= 60). This Figure shows that

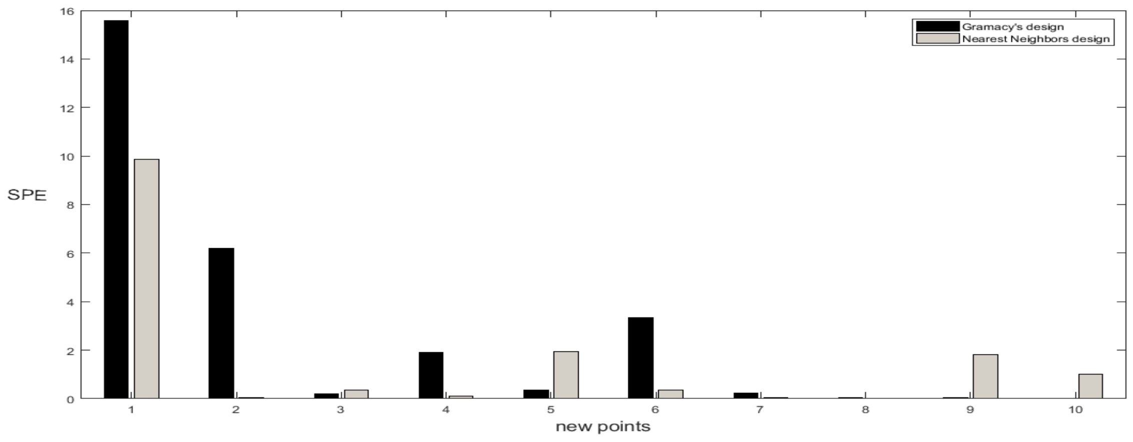

Figure 15: SPEs for ten new input combinations, for Gramacy's design and our NN design, in the borehole example

our design gives smaller SPEs for six of the ten new points; the Figure clearly shows this result for the first six new points, while for the last four new points the exact values of Gramacy's design versus our design are 0.2283 versus 0.0439, 0.0385 versus 0.0005, 0.0385 versus 1.8076, 0.0014 versus 0.9979. The overall ASPE (of the ten new points) are 2.7852 versus 1.5563. (The SPEs for our LHS with triangular distributions are much higher, so we do not display these SPEs in the Figure.) We conjecture that our one-shot NN design performs relatively well because Gramacy's customized design requires the specification of a Kriging metamodel (e.g., an OK model with Gaussian correlation function) plus the estimation of the Kriging hyperparameters (e.g., $\psi = (\mu, \tau^2, \theta')'$) to sequentially select new points after the initial (small) set of n_0 ($< n$) NN points. (Our explanation is inspired by the popular conjecture that OK performs relatively well because UK requires the estimation of additional parameters in the polynomial for the mean; see again the references at the end of Section 2.)

6 Conclusions and future research

Kriging for prediction in big data with N observations is problematic because Kriging requires the estimation of its (hyper)parameters and the resulting Kriging predictor and Kriging variance. To solve this problem, Gramacy (2016) selects a small subset of size n from the N previously observed "old" data. This subset is selected sequentially, using the sequentially re-estimated Kriging variance. The resulting designs turn out to be "local"; i.e., most design points are concentrated around the "new" point to be predicted. We develop three alternative one-shot methods that do not depend on the Kriging model and its parameters: (i) use LHS with uniform marginal input distributions to select a small subset such that this subset still covers the original input space–albeit coarser; (ii) use LHS with triangular input distributions to select a subset with relatively many—but not all—combinations close to the new combination that is to be predicted, and (iii) select a subset with the NNs of the new combination. To evaluate these designs, we compare their SPEs in several numerical experiments. These experiments show that our NN design is a viable alternative for Gramacy (2016)'s more sophisticated sequential design. .

Expensive simulation implies that the number of observations N is relatively small compared to big data situations. Consequently, in expensive simulations we try to find the smalles tN that still gives statistically accurate results. In big data, however, N is such that we must select a subset of size n ($\ll N$). In practice, the simulation analysts should use the highest possible value for n—given the computer's memory size and speed.

In future research we may investigate Kriging and big data, for *random* output. This randomness may result from measurement error in the output data of a *real* (non-simulated) system with big data. This error is modeled through the so-called nugget effect; see Kleijnen (2015, pp. 206—207). The nugget effect in Kriging metamodels of deterministic simulation models is also discussed in Chen et al. (2016), Mukhopadhyay et al. (2016), and Peng and Wu (2014).

Random output also results in *random simulation*, such as discrete-event simulation and agent-based simulation. Deterministic simulation also gives random output if the values of the inputs (parameters) are uncertain so the input values are sampled from a prior input distribution: so-called "uncertainty propagation".

Random simulation has so-called "intrinsic" variances that may vary with the input combinations (in machine learning, intrinsic variances turn the outputs into "latent" variables). Intrinsic variances may be analyzed through either "stochastic Kriging" or "hetGP"; see Meng and Ng (2016) and Binois et al. (2015) respectively. Constant intrinsic variances are discussed in Damianou (2015) and Nickson et al. (2015). Correlated intrinsic noise is created by common random numbers, and deserves attention too; see Kleijnen (2017).

References

Angelino, E., M.J. Johnson, and R.P. Adams (2016), Patterns of scalable Bayesian inference. arXiv preprint arXiv:1602.05221, 2016

Barber, C.B., D.P. Dobkin, and H. Huhdanpaa (1996), The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22, no. 4, pp. 469–483

Benková, E., R. Harman, and W.G. Müller (2015), Privacy sets for constrained space-filling. *arXiv*:1510.05284

Binois, M., R.B. Gramacy and M. Ludkovskiz (2016), Practical heteroskedastic Gaussian process modeling for large simulation experiments. *ArXiv*, 17 Nov 2016

Bouhlel, M.A., N. Bartoli, A. Otsmane, and J. Morlier (2016), Improving Kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53, no. 5, pp. 935–952

Bouhlel, M.A., N. Bartoli, R.G. Regis, A. Otsmane, and J. Morlier (2017), Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. *Optimization Engineering*, in press

Bradley, J.R., N. Cressie, and T. Shi (2016). A comparison of spatial predictors when datasets could be very large. *Statistics Surveys*, 10, pp. 100–131

Chen, H., J.L. Loeppky, J. Sacks, and W.J. Welch (2016), Analysis methods for computer experiments: how to assess and what counts? *Statistical Science*, 31, no. 1, pp. 40-60

Chevalier, C., X. Emery, and D. Ginsbourger (2014), Fast update of conditional simulation ensembles. <hal-00984515>

Chilès, J.-P. and N Desassis (2018), Fifty years of Kriging. *Handbook of Mathematical Geosciences; fifty years of IAMG*, edited by B. S. Daya Sagar, Q. Cheng, and F. Agterberg, Springer, pp. 589–612

Damianou, A. (2015), *Deep Gaussian Processes and Variational Propagation of Uncertainty*. Doctor of Philosophy dissertation, Department of Neuroscience, University of Sheffield, U.K. (https://arxiv.org/abs/1510.07965)

Dixon, L.C.W. and G.P. Szego (1978), The global optimisation problem: an introduction. In: Towards global optimisation, volume 2, eds. Dixon, L.C.W. and G.P. Szego, North-Holland, New York, pp. 1-15

Dong, H. and M.K. Nakayama (2017), Quantile estimation with Latin hypercube sampling. *Operations Research*, 65, no. 6, pp. 1678–1695

Efron, B. (2015), Frequentist accuracy of Bayesian estimates. Royal Statistical Society, Series B, 77, no. 3, pp. 617-646

Erickson, C.B., B.E. Ankenman, S.M. Sanchez (2017), Comparison of Gaussian process modeling software. *European Journal of Operational Research*, accepted

Floudas, C.A. et al. (1999), *Handbook of Test Problems in Local and Global Optimization*. Dordrecht, The Netherlands: Kluwer Academic Publishers

Fouladinejad, N., N. Fouladinejad, M.K.A. Jalil, and J.M. Taib (2017), Decomposition-assisted computational technique based on surrogate modeling for real-time simulations. *Complexity*, https://doi.org/10.1155/2017/1686230

Gal, Y., M. van der Wilk, and C.E. Rasmussen (2014), Distributed variational inference in sparse Gaussian process regression and latent variable models. *arXiv*:1402.1389v2 [stat.ML] 29 Sep 2014

Gramacy, R.B. (2016), laGP: large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, 72, no. 1, pp. 1–46

Gramacy, R. B. and D. W. Apley (2015), Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24, no. 2, pp. 561–578

Gramacy, R. B., G. A., Gray, S.L. Digabel, H.K.H. Lee, P. Ranjan, G. Wells, and S.M. Wild (2015), Modeling an augmented Lagrangian for improved blackbox constrained optimization. *Technometrics*, 61, pp. 1-38

Guhaniyogi, R. and S. Banerjee (2018), Meta-Kriging: scalable Bayesian modeling and inference for massive spatial datasets. *Technometrics*, DOI: 10.1080/00401706.2018.1437474

Guinness, J. (2018), Permutation and grouping methods for sharpening Gaussian process approximations. *Technometrics*, DOI: 10.1080/00401706.2018.1437476

Gutiérrez de Ravé, E., F.J. Jiménez-Hornero, A.B. Ariza-Villaverde, and J.M. Gómez-López (2014), Using general-purpose computing on graphics processing units (GPGPU) to accelerate the ordinary kriging algorithm. *Computers & Geosciences*, 64, pp. 1–6

Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2), 150–194

Kajero, O.T. , T. Chen, Y. Yao, Y-C. Chuan, D.S.H. Wong (2017), Metamodelling in chemical process system engineering. *Journal of the Taiwan Institute of Chemical Engineers*, 73, pp. 135–145

Kamiński, B. (2015), A method for updating of stochastic Kriging metamodels. *European Journal of Operational Research*, 247, no. 3, pp. 859–866

Kleijnen, J.P.C. (2017), Design and analysis of simulation experiments: tutorial. Invited chapter for *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences*, edited by A. Tolk, J. Fowler, G. Shao, and E. Yücesan, Springer, pp. 135-158

Kleijnen, J.P.C. (2015), *Design and analysis of simulation experiments; second edition.* Springer

Kleijnen, J. P. C. and W. Shi (2017). Sequential probability ratio tests: conservative and robust. CentER Discussion Paper; Vol. 2017-001, Tilburg: CentER, Center for Economic Research

Law, A.M. (2015), *Simulation modeling and analysis, 5th edition.* McGraw-Hill, Boston

Le Guiban, K., A. Rimmel, M-A. Weisser, and J. Tomasik (2017), The first approximation algorithm for the maximin Latin hypercube design problem. *Operations Research*, in press

Lim, C.Y., C-H. Chen, W-Y. Wu (2017), Numerical instability of calculating inverse of spatial covariance matrices. *Statistics and Probability Letters*, 129, pp.182–188

Loeppky, J.L., J. Sacks, and W. Welch (2009) Choosing the sample size of a computer experiment: a practical guide. *Technometrics*, 51, no. 4, pp. 366–376

Lophaven, S.N., H.B. Nielsen, and J. Sondergaard (2002) DACE: a Matlab Kriging toolbox, version 2.0. IMM Technical University of Denmark, Kongens, Lyngby

McKay, M.D., R.J. Beckman, and W.J. Conover (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, no. 2, pp. 239–245 (reprinted in *Technometrics*, 42, no. 1, 2000, pp. 55–61)

Martinez-Cantin, R. (2016), Funneled Bayesian optimization for design, tuning and control of autonomous systems. arXiv:1610.00366v1 [cs.AI] 2 Oct 2016

Meng, Q. and S.H. Ng (2016), Combined Global and local method for stochastic simulation optimization with an AGLGP model. *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, pp. 827-838

Morgan, B.L., H.C. Schramm, J.R. Smith, T.W. Lucas, M.L. McDonald, P.J. Sánchez, S.M. Sanchez, and S.C. Upton (2017), Improving U.S. Navy campaign analyses with big data. *Interfaces*, in press

Mukhopadhyay, T., S Chakraborty, S Dey, S Adhikari, and R. Chowdhury (2016), A critical assessment of Kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells. *Archives of Computational Methods in Engineering*, pp. 1–24

Nickson, T., T. Gunter, C. Lloyd, M.A. Osborne, and S. Roberts (2015), Blitzkriging: Kronecker-structured stochastic Gaussian processes. *arXiv:*1510.07965v2 [stat.ML] 31 Oct 2015

Peng, C-Y. and C.F.J. Wu (2014), On the choice of nugget in Kriging modeling for deterministic computer experiments. *Journal of Computational and Graphical Statistics,* 23, no. 1, pp. 151–168

Pronzato, L. and M-J. Rendas (2017), Bayesian local Kriging. *Technometrics*, in press

Rasmussen, C. and Williams, C. (2006). *Gaussian processes for machine learning.* MIT Press, Cambridge, Massachusetts

Santner, T.J., B.J. Williams, and W.I. Notz (2018), *The design and analysis of computer experiments; second revised edition.* Springer, New York

Sung, C-L., R. B. Gramacy, and B. Haaland (2016), Potentially predictive variance reducing sub-sample locations in local Gaussian process regression. *ArXiv e-prints*

Surjanovic, S. and D. Bingham (2016), Virtual library of simulation experiments: test functions and datasets. http://www.sfu.ca/~ssurjano

Tzeng, S. and H-C. Huang (2018), Resolution adaptive fixed rank kriging. *Technometrics*, 60, no. 2, pp. 198–208

Uhrmacher, A., S. Brailsford, J. Liu , M. Rabe, and A. Tolk (2016), Panel - Reproducible Research in Discrete Event Simulation - A Must or Rather a Maybe? *Proceedings of the 2016 Winter Simulation Conference*, T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds., pp. 1301–1315

Van Stein, B., H. Wang, W. Kowalczyk, M. Emmerich, and T. Bäck (2017), Cluster-based Kriging approximation algorithms for complexity reduction. *arXiv*:1702.01313v1 [cs.LG] 4 Feb 2017

Wang, H., L. Tang, and G.Y. Li (2011), Adaptive MLS-HDMR metamodeling techniques for high dimensional problems. *Expert Systems with Applications*, 38, no. 11, pp. 14117–14126

Wang, W. and X. Chen (2017), An adaptive two-stage dual metamodeling approach for stochastic simulation experiments. Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg

Woods, D.C. and S.M. Lewis (2016), Design of experiments for screening. *Handbook of uncertainty quantification*, Springer (preprint: arXiv:1510.05248)

Xiong, S., P.Z. Qian, and C.J. Wu (2013). Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55, no. 1, pp. 37–46

Xiong, Y., W. Chen, D. Apley, and X. Ding. 2007. A non-stationary covariance-based kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering*, 71, no. 6, pp. 733–756

Xu, J., E. Huang, L. Hsieh, L.H. Lee, Q-S. Jia, and C-H. Chen (2016), Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *Journal of Simulation*, 10, pp. 310–320

Yuan, J., V. Nian, B. Su, and Q. Meng (2017), A simultaneous calibration and parameter ranking method for building energy models. *Applied Energy*, 206, pp. 657–666

Acknowledgement 2 *Robert Gramacy (Virginia Tech) shared his 50 combinations of two inputs in the three designs for his example (which we reproduce in Appendix 4), and he provided us with 60 combinations of the eight inputs for each of the ten new combinations to be predicted for the borehole example. Ivo Couckuyt (University of Gent, Belgium) referred us to Damianou (2015) and Nickson et al. (2015). Tom Santner (Ohio State University) suggested to make three plots for the borehole example, as explained in the text. Dick den Hertog (Tilburg University) suggested the linear programming solution for our convex hull problem.*

Appendix 1: Abbreviations and major mathematical symbols

Abbreviations:

ASPE: average squared prediction error
MSPE: mean squared prediction error
PDF: probability density function
PE: prediction error
SPE: squared prediction error

Symbols:

We list Latin symbols before Greek symbols, and lower-case symbols before upper-case symbols

$F(x)$: cumulative density function of x
m: number of macroreplications
n:
x: random (simulation) observation (output, response)

α: complement of nominal coverage probability
μ: mean (expected value)
σ^2: variance
$\mathbf{\Sigma}$: covariance matrix

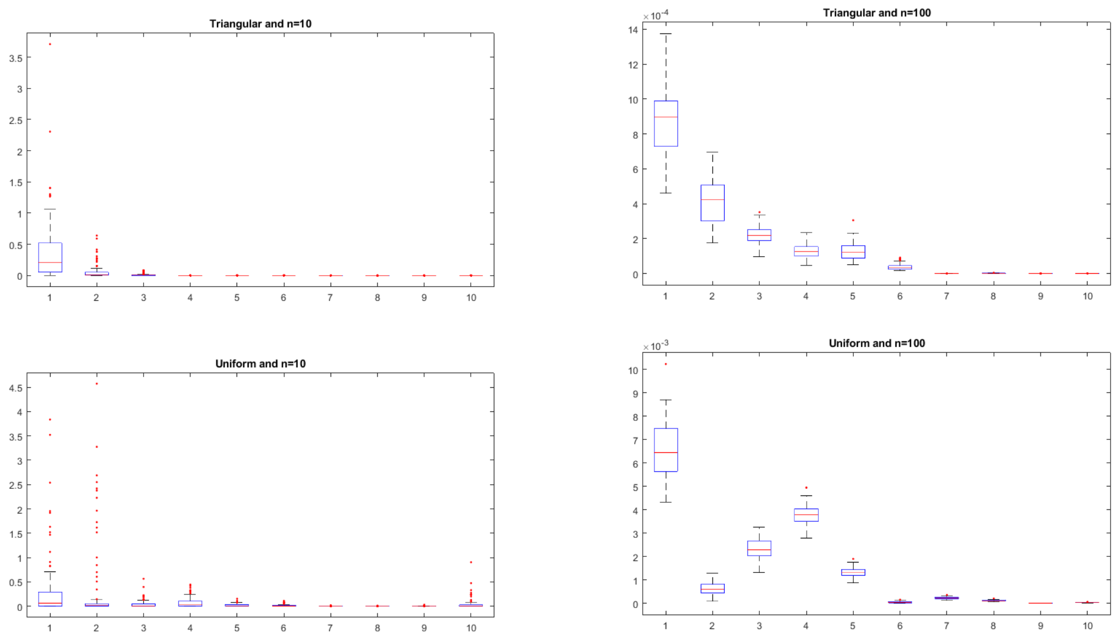

Figure 16: Box-and-whisker plots for SPEs in Xiong et al. (2007)'s example

Appendix 2: Xiong et al. (2007)'s example: box-and-whisker plots for estimated SPEs

Fig.16 displays box-and-whisker plots for the estimated SPEs in our experiment with Xiong et al. (2007)'s example; there are plots for n = 10 and n =100 old points, for n_0 = 10 new points $x_{0;t}$ with t = 1, ..., 10, and for T($\mathbf{x}_0$) with sampled points, and U$(0, 1)$ with sampled points (obviously, midpoints for T($\mathbf{x}_0$) or U$(0, 1)$ implies that all m macroreplications give the same result). To make these plots, we use the standard MATLAB function called "boxplot", which creates a plot for each $x_{0;t}$; we point out that $x_{0;t}$ is not equispaced (see column 1 of Table 1).

Appendix 3: M/M/1-inspired example

The M/M/1 model itself is a Markov chain resulting from mutually independent exponential interarrival times (say) $a$that are independent of mutually independent exponential service times s, and has one server; implicit assumptions are that there is unlimited room for waiting customers, and that customers are served in first-in-first-out (FIFO) order. This model has k= 1 input; namely, the traffic rate ρdefined as the mean arrival rate λdivided by the mean service rate μ. The steady state can be reached only if ρ< 1. We let a_{t+1} denote the interarrival time between customers $t$and $t + 1$, and s_t the service time of customer t. We assume that the output of interest is w, the waiting time of a

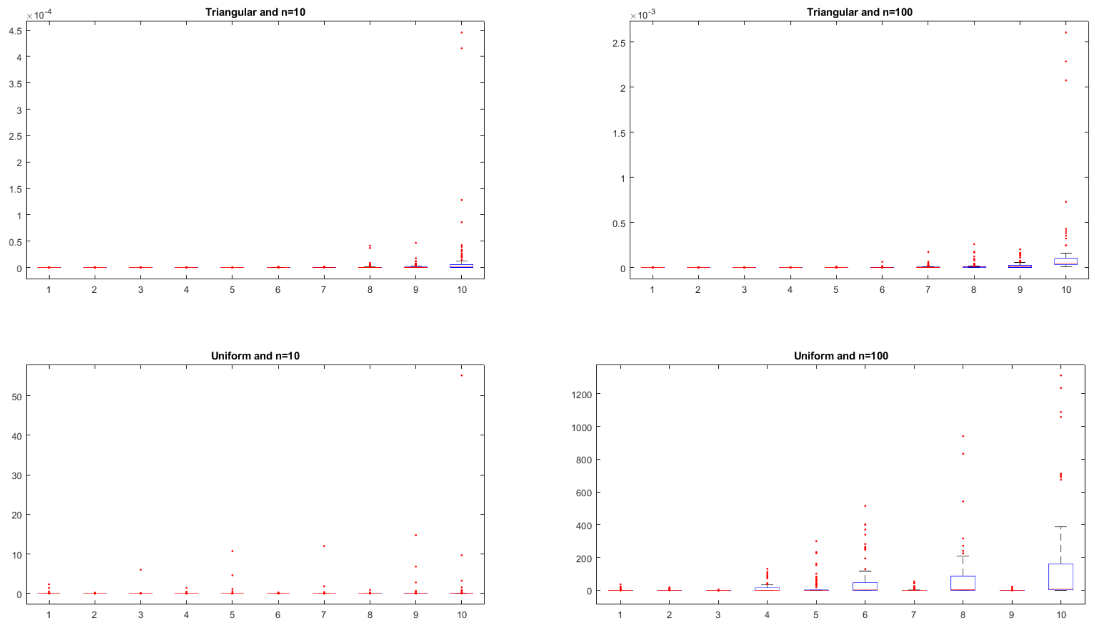

Figure 17: Box-and-whisker plots for SPEs in M/M/1-inspired example

customer. We estimate E(w)through the average $\overline{w}$= $\sum_{t=1}^{T} w_t/n$where Tdenotes the number of customers of the simulation run. Furthermore, we assume that the simulation starts in the "empty" state so w_1= 0.The dynamics of a single-server system are specified by the so-called *Lindley recurrence formula* w_{t+1}= $max(0, w_t + s_t - a_{t+1})$. Mathematical analysis gives the mean steady-state waiting-time:

$$E(w_t \mid t \uparrow \infty) = \frac{\lambda}{\mu(\mu - \lambda)} = \frac{1}{\mu}\frac{\rho}{(1-\rho)},$$

so if we select the time unit such that μ= 1, then $E(w_t|t \uparrow \infty)$= $\rho/(1-\rho)$. In our M/M/1-inspired example we replace $\rho$by x, and $E(w_t|t \uparrow \infty)$by w.

Fig.17 displays box-and-whisker plots for the M/M/1-inspired example; these plots closely resemble the plots for Xiong et al. (2007)'s example, which are displayed in the immediately preceding appendix.

Appendix 4: Gramacy (2016)'s example: three sequential designs

Table 4 gives the exact values of the $n = 50$ input combinations $(z_{i;1}, z_{i;2})$for the three designs derived in Gramacy (2016).

	Design 1		Design 2		Design 3	
i	$z_{i;1}$	$z_{i;2}$	$z_{i;1}$	$z_{i;2}$	$z_{i;1}$	$z_{i;2}$
1	-1.72	1.72	-1.72	1.72	-1.72	1.72
2	-1.74	1.72	-1.74	1.72	-1.74	1.72
3	-1.72	1.74	-1.72	1.74	-1.72	1.74
4	-1.74	1.74	-1.74	1.74	-1.74	1.74
5	-1.72	1.7	-1.72	1.7	-1.72	1.7
6	-1.7	1.72	-1.7	1.72	-1.7	1.72
7	-1.64	1.64	-1.64	1.64	-1.8	1.6
8	-1.82	1.62	-1.82	1.62	-1.74	1.7
9	-1.74	1.7	-1.74	1.7	-1.7	1.74
10	-1.7	1.74	-1.7	1.74	-1.66	1.66
11	-1.62	1.82	-1.62	1.82	-1.7	1.7
12	-1.7	1.76	-1.7	1.76	-1.72	1.76
13	-1.76	1.7	-1.76	1.7	-1.8	1.88
14	-1.86	1.86	-1.86	1.84	-1.76	1.74
15	-1.74	1.76	-1.76	1.74	-1.84	1.7
16	-1.7	1.7	-1.7	1.7	-1.92	1.58
17	-1.76	1.72	-1.72	1.76	-1.66	1.82
18	-1.88	1.52	-1.76	1.72	-1.76	1.72
19	-1.72	1.76	-1.88	1.52	-1.76	1.7
20	-1.9	1.9	-1.92	1.9	-1.72	1.68
21	-1.68	1.7	-1.7	1.68	-1.7	1.76
22	-1.76	1.74	-1.68	1.74	-1.74	1.76
23	-1.72	1.68	-1.74	1.76	-1.74	1.68
24	-1.68	1.72	-1.72	1.68	-1.7	1.68
25	-1.74	1.68	-1.78	1.72	-1.68	1.7
26	-1.76	1.76	-1.68	1.72	-1.68	1.74
27	-1.5	1.92	-1.5	1.92	-1.82	1.82
28	-1.68	1.74	-1.74	1.68	-1.72	1.78
29	-1.5	1.48	-1.76	1.76	-1.88	1.52
30	-1.7	1.68	-1.68	1.76	-1.92	1.92
31	-1.78	1.72	-1.5	1.5	-1.76	1.76
32	-1.72	1.78	-1.6	1.6	-1.76	1.68
33	-1.62	1.62	-1.68	1.7	-1.74	1.78
34	-1.76	1.68	-1.46	1.46	-1.62	1.8
35	-1.68	1.76	-1.78	1.74	-1.6	2
36	-1.48	1.44	-1.72	1.78	-1.68	1.72
37	-1.68	1.68	-1.76	1.68	-1.68	1.76
38	-1.78	1.74	-1.68	1.68	-1.66	1.86
39	-1.74	1.78	-1.84	1.84	-1.7	1.78
40	-1.84	1.84	-1.74	1.78	-1.72	1.66
41	-1.6	1.6	-1.78	1.76	-1.78	1.72
42	-1.62	1.84	-1.82	1.6	-1.78	1.7
43	-1.42	1.96	-1.62	1.84	-1.78	1.76
44	-1.7	1.78	-1.72	1.66	-1.86	1.92
45	-1.84	1.58	-1.42	1.96	-1.74	1.66
46	-1.78	1.7	-1.7	1.78	-1.34	1.82
47	-1.72	1.66	-1.66	1.72	-1.6	1.68
48	-1.66	1.72	-1.62	1.62	-1.78	1.74
49	-1.76	1.78	-1.78	1.7	-1.66	1.72
50	-1.6	1.84	-1.7	1.66	-1.66	1.74

Table 4: Gramacy (2015)'s three designs, each with 50 combinations of 2 inputs