SIAM/ASA J. Uncertainty Quantification
Vol. 4, pp. 739–766

Sequential Design with Mutual Information for Computer Experiments (MICE): Emulation of a Tsunami Model[*]

Joakim Beck[†] and Serge Guillas[†]

Abstract. Computer simulators can be computationally intensive to run over a large number of input values, as required for optimization and various uncertainty quantification tasks. The standard paradigm for the design and analysis of computer experiments is to employ Gaussian random fields to model computer simulators. Gaussian process models are trained on input-output data obtained from simulation runs at various input values. Following this approach, we propose a sequential design algorithm MICE (mutual information for computer experiments) that adaptively selects the input values at which to run the computer simulator in order to maximize the expected information gain (mutual information) over the input space. The superior computational efficiency of the MICE algorithm compared to other algorithms is demonstrated by test functions and by a tsunami simulator with overall gains of up to 20% in that case.

Key words. active learning, best linear unbiased prediction, Gaussian process, shallow water equations

AMS subject classifications. 60G15, 62M20, 62K99, 65Y20, 91B74

DOI. 10.1137/140989613

1. Introduction.

Computer experiments are widely employed to study physical processes [31, 36] and involve running a computer simulator which mimics the physical process at various input values. When the computer simulator is computationally expensive to run, say, minutes, hours, or even days, often on a high performance cluster, only a limited number of simulation runs can be afforded, making the planning of such experiments even more important. Surrogate models, also known as emulators, are often used as means for designing and analyzing computer experiments [31]. Emulators are statistical models that have been used to approximate the input-output behavior of computer simulators for making probabilistic predictions. In this setting, we want to find a design of computer experiments that with minimal computational effort leads to a surrogate model with a good overall fit. We restrict our attention to deterministic computer simulators with a scalar output. In design of experiments it is customary to use space-filling designs [36] such as uniform designs, multilayer designs, maximin (Mm)- and minimax (mM)-distance designs, and Latin hypercube designs (LHD). Space-filling designs treat all regions of the design space as equally important, but are "one shot" designs that may waste computations over some unnecessary regions of the input space. A variety of adaptive designs have been proposed which can take advantage of information collected during the experimental design process [21, 31], typically in the form of input-output

[*]Received by the editors September 30, 2014; accepted for publication (in revised form) February 16, 2016; published electronically June 2, 2016. This work was supported by NERC grant NE/J017434/1 "Probability, Uncertainty and Risk in the Environment."
http://www.siam.org/journals/juq/4/98961.html

[†]Department of Statistical Science, University College London, London WC1E 7HB, UK (joakim.beck@ucl.ac.uk, s.guillas@ucl.ac.uk).

data from simulation runs. An algorithm is called adaptive if it updates its behavior to new data. Some classical adaptive design criteria are the maximum mean squared prediction error (MMSPE), the integrated MSPE (IMSPE), and the entropy criterion (see, e.g., [30]).

We adopt the design and analysis of computer experiments (DACE) framework proposed in the seminal paper of Sacks et al. [30], within which the computer simulator output is modeled as a realization of a random field, typically assumed Gaussian. When given a set of input-output data, the best linear unbiased predictor (BLUP) and the associated MSPE for the random field can be expressed in closed forms [30, 31]. Moreover, when the random field is Gaussian, the resulting BLUP is a so-called Gaussian process (GP) emulator. GP emulators are routinely applied to handle computationally intensive computer simulators in the fields of simulation [31], global optimization [17], and uncertainty quantification [3, 32], among others. Applications include CFD simulation of a rocket booster [13] and climate simulation [5]. By using the GP approach, a range of statistical design criteria can be estimated directly [5, 34]. Finding an optimal design is usually computationally very intensive, except for relatively small designs. A way to circumvent this issue is to consider sequential designs [5, 13, 21]. In a sequential design, points are systematically chosen, often one at a time. Sequential designs are generally not optimal, but often very effective in practice. Two popular sequential designs are active learning MacKay (ALM), and active learning Cohn (ALC). ALC tends to have better overall predictive performance but involves a higher computational cost [13].

In this work we propose a new sequential algorithm, called MICE (mutual information for computer experiments), which is based on the information theoretic mutual information measure given in [7], where the objective of maximizing the information that a design provides about the other input values, as suggested by Caselton and Zidek [4]. Mutual information is a measure of the information contained in one random variable about another. Krause, Singh, and Guestrin [19] proposed a sequential MI algorithm for sensor placement, which sequentially maximizes the mutual information between a GP over the chosen sensor locations and another GP over the locations which have not yet been selected. The MICE criterion is a modified version of the MI criterion in [19], where an extra parameter is introduced to improve robustness. This modification is critical when high-dimensional spaces are considered. We demonstrate by numerical examples that MICE balances well prediction accuracy and computational complexity. We are particularly interested in deterministic computer simulation experiments with more than just a few input variables.

This paper is organized as follows. Section 2 reviews GP modeling for prediction and presents some popular sequential design algorithms within the DACE framework. In section 2.2, an MI-based design criterion is proposed for computer experiments. The MI algorithm is described in section 3.1, and a practical limitation is shown in section 3.1.2. Section 3.2 presents the MICE algorithm and some theoretical results. Section 4 details the computational costs associated with the different sequential design algorithms. A numerical comparison of MICE with other methods is provided for a few standard test functions, in lieu of computer simulators, in section 5, and for a tsunami simulator that solves nonlinear shallow water equations in section 6. Critically, we examine accuracy versus computational cost, as some algorithms can be quite time consuming. Section 7 summarizes our conclusions. Proofs of the theorems are given in Appendix A.

2. Gaussian process modeling for prediction.

Here, we follow the approach proposed by Sacks et al. [30], where a deterministic computer simulator $y(\boldsymbol{x}) : \mathcal{X} \subseteq \mathbb{R}^p \to \mathbb{R}$ is treated as a random function, $Y(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, except at the points where the simulator output is known. More specifically, $Y(\boldsymbol{x})$ is modeled as a random field with $E\left(Y^2(\boldsymbol{x})\right) < \infty$ given a set of training data, which consists of n input-output pairs $(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X} = (\boldsymbol{x}_j)_{j=1}^n$, $\boldsymbol{y} = (y_j)_{j=1}^n$, and $y_j = y(\boldsymbol{x}_j)$.

The aim is to determine a random process that can describe the set of data sufficiently well. It is customary that the mean $\mathrm{E}[Y(\boldsymbol{x})]$ takes the form $\boldsymbol{h}^T(\boldsymbol{x})\boldsymbol{\beta}$, that is, a linear combination of q regressors $\boldsymbol{h}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^q$ with coefficients $\boldsymbol{\beta} \in \mathbb{R}^q$. In practice, a fixed constant, or a linear regression model, tends to perform well. The covariance $\mathrm{Cov}(Y(\boldsymbol{x}), Y(\boldsymbol{x}'))$, for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, is written in the form $\Sigma(\boldsymbol{x}, \boldsymbol{x}'; \sigma^2) = \sigma^2 K(\boldsymbol{x}, \boldsymbol{x}')$, where $\sigma^2 (> 0)$ is a scale parameter (often called the process variance) and $K(\boldsymbol{x}, \boldsymbol{x}')$ is the correlation function. The correlation function is often expressed as a product of stationary, one-dimensional correlation functions. One such choice is the squared-exponential (SE) correlation [27]:

$$K(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}) = \prod_{i=1}^{p} \exp\left(\frac{-(x_i - x_i')^2}{2\ell_i^2}\right), \tag{2.1}$$

where $\boldsymbol{\xi} = (\ell_1, \ell_2, \ldots, \ell_p)^T \in \mathbb{R}_+^p$. Here ℓ_i represents the correlation length for the ith input dimension.

In this approach, for predicting the output $y(\boldsymbol{x})$ at any desired $\boldsymbol{x} \in \mathcal{X}$, linear predictors are considered of the form $\hat{y}(\boldsymbol{x}) = \boldsymbol{\lambda}^T(\boldsymbol{x})\boldsymbol{y}$ for some vector $\boldsymbol{\lambda}(\boldsymbol{x}) \in \mathbb{R}^n$. BLUP, assuming $\boldsymbol{\xi}$ is known, is the one that minimizes the MSPE with respect to $\boldsymbol{\lambda}(\boldsymbol{x})$,

$$\mathrm{MSE}[\boldsymbol{\lambda}^T(\boldsymbol{x})\boldsymbol{y}] = E\left[(\boldsymbol{\lambda}^T(\boldsymbol{x})\boldsymbol{y} - Y(\boldsymbol{x}))^2\right], \tag{2.2}$$

subject to the unbiasedness constraint $\boldsymbol{H}^T\boldsymbol{\lambda}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})$, where $\boldsymbol{H} = (\boldsymbol{h}(\boldsymbol{x}_j))_{j=1}^n$. The MSPE of $\hat{y}(\boldsymbol{x})$ is minimized for

$$\hat{\lambda}(\boldsymbol{x}) = \boldsymbol{k}^T(\boldsymbol{x})\boldsymbol{K}^{-1} + \boldsymbol{K}^{-1}\boldsymbol{H}\frac{\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x})}{\boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{H}}, \tag{2.3}$$

which leads to the BLUP of $Y(\boldsymbol{x})$,

$$\begin{aligned}\hat{y}(\boldsymbol{x}) = \boldsymbol{\lambda}^T(\boldsymbol{x})\boldsymbol{y} &= \boldsymbol{k}^T(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{y} + \boldsymbol{K}^{-1}\boldsymbol{H}\frac{\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x})}{\boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{H}}\boldsymbol{y} \\ &= \boldsymbol{h}^T(\boldsymbol{x})\hat{\boldsymbol{\beta}} + \boldsymbol{k}^T(\boldsymbol{x})\boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}}),\end{aligned} \tag{2.4}$$

where $\hat{\boldsymbol{\beta}} = (\boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{y}$ is the generalized least squares estimate of $\boldsymbol{\beta}$, $\boldsymbol{K}$ is the $n \times n$ correlation matrix whose (i, j)th entry is given by $K(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\xi})$ for $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}$, and the $n \times 1$ vector $\boldsymbol{k}(\boldsymbol{x}; \boldsymbol{\xi})$ has entry j given by $K(\boldsymbol{x}, \boldsymbol{x}_j; \boldsymbol{\xi})$ for $\boldsymbol{x}_j \in \boldsymbol{X}$. The correlation matrix must be positive semidefinite. The MSPE is given by

$$\begin{aligned}\mathrm{MSE}[\hat{y}(\boldsymbol{x})] = \sigma^2 \Bigg(& 1 - \boldsymbol{k}^T(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}) \\ & + \frac{(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}))^T(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}))}{\boldsymbol{H}^T\boldsymbol{K}^{-1}\boldsymbol{H}}\Bigg).\end{aligned} \tag{2.5}$$

The predictor is unbiased and interpolates the training data, that is, $\hat{y}(\boldsymbol{x}_j) = y(\boldsymbol{x}_j)$ for $\boldsymbol{x}_j \in \boldsymbol{X}$. Note that the regularity of the correlation function $K(\boldsymbol{x}, \boldsymbol{x})$ determines the regularity of the predictor $\hat{y}(\boldsymbol{x})$ [39], which means that the regularity of $y(\boldsymbol{x})$ should ideally be reflected in the choice of correlation structure.

As in [30], we also make the assumption that $Y(\boldsymbol{x})$ is a GP, which is convenient from a computational perspective. This yields a GP emulator of $y(\boldsymbol{x})$ [28] with mean $\hat{y}(\boldsymbol{x})$ and variance

$$\hat{s}^2(\boldsymbol{x}) = \mathrm{MSE}[\hat{y}(\boldsymbol{x})], \tag{2.6}$$

which may be viewed as a measure of uncertainty in the prediction. A GP with the SE correlation function is infinitely mean square differentiable, and the realizations (or sample paths) of this process tend to be unrealistically smooth for modeling computer experiments [39]. To be more general, we consider the Matérn family of correlation functions [15],

$$K(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\xi}) = \prod_{i=1}^{p} \frac{1}{2^{\nu-1}\Gamma_\nu} \left(\frac{2\nu^{\frac{1}{2}}|x_i - x_i'|}{\ell_i} \right)^{\nu} J_\nu \left(\frac{2\nu^{\frac{1}{2}}|x_i - x_i'|}{\ell_i} \right), \tag{2.7}$$

where $\boldsymbol{\xi} = (\ell_1, \dots, \ell_p, \nu)^T$, Γ_ν is the Gamma function for ν, and J_ν is a modified Bessel function of order $\nu > 0$. The parameter ν regulates the smoothness of the process, which allows us to model data of different degrees of smoothness. The SE correlation is a special case of a Matérn correlation when ν goes to ∞. A GP with the Matérn correlation function is $\lfloor \nu - 1 \rfloor$ times mean square differentiable [39], where $\lfloor\ \rfloor$ denotes the floor function. The Matérn correlation function with fixed $\nu = 5/2$ can be written in an explicit form (see, e.g., [27]) and is the one used in our numerical tests, unless stated otherwise.

2.1. Maximum likelihood estimation of unknown parameters. The parameters involved in the covariance structure are usually unknown (σ^2 and $\boldsymbol{\xi}$, say) and need to be estimated. In this work, the parameters are estimated by maximum likelihood estimation (MLE) using available input-output data (see, e.g., [31] and references therein). The MLE of σ^2 is $\hat{\sigma}^2(\boldsymbol{\xi}) = (\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}}(\boldsymbol{\xi}))^T \boldsymbol{K}_{\boldsymbol{\xi}}^{-1} (\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}}(\boldsymbol{\xi}))/n$ for fixed $\boldsymbol{\xi}$ [30], and the MLE of $\boldsymbol{\xi}$, denoted by $\hat{\boldsymbol{\xi}}$, can be found by maximizing the profile log-likelihood,

$$\hat{\boldsymbol{\xi}} = \arg\max_{\boldsymbol{\xi} \in \Xi} \mathcal{L}_p(\boldsymbol{\xi}), \tag{2.8}$$

where $\mathcal{L}_p(\boldsymbol{\xi}) = \mathcal{L}(\hat{\sigma}^2(\boldsymbol{\xi}), \hat{\beta}(\boldsymbol{\xi}), \boldsymbol{\xi})$ is the profile log-likelihood for $\boldsymbol{\xi}$, $\mathcal{L}(\cdot)$ is the marginal log-likelihood function, and Ξ is a search domain. Assuming the data are normally distributed, the negative log-marginal likelihood is

$$-\mathcal{L} = \frac{1}{2} \ln|\boldsymbol{K}| + \frac{1}{2} \boldsymbol{H}^T \boldsymbol{K}^{-1} \boldsymbol{H} + \frac{n}{2} \log 2\pi\sigma^2, \tag{2.9}$$

which means optimization problem (2.8) can be solved by finding the values of $\boldsymbol{\xi}$ that maximize $n \ln \hat{\sigma}^2(\boldsymbol{\xi}) + \ln \det(\boldsymbol{K}_{\boldsymbol{\xi}})$; see [31]. By inserting the MLEs as if they were the *true* values, we have the so-called *estimated* BLUP (EBLUP) $\hat{y}(\boldsymbol{x}; \hat{\boldsymbol{\xi}})$ [40]. As shown in [40], the estimator $\mathrm{MSE}[\hat{y}(\boldsymbol{x}; \hat{\boldsymbol{\xi}})]$ tends to underestimate the MSPE.

A Bayesian approach to model the uncertain parameters could have been considered (see, for example, [15, 27]), but at a higher computational cost. Note that, although we restrict our attention to MLE, our results are still relevant within a Bayesian setting.

2.2. The design of computer experiments. This section presents some of the approaches to the design of computer experiments where the goal is to determine the input values at which data should be collected to best approximate the computer simulator output over the design space $\mathcal{X} \subseteq \mathbb{R}^p$. There are a variety of ways to design such experiments [21, 31]. Design criteria based on the MSPE are natural choices [30]. Examples are the maximum MSPE (MMSPE) criterion, $\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{MSE}[\hat{y}_N(\boldsymbol{x})]$, and the integrated MSPE (IMSPE) criterion, $\int_{\mathcal{X}} \mathrm{MSE}[\hat{y}_N(\boldsymbol{x})]\,\mathrm{d}\boldsymbol{x}$, both of which are to be minimized. Here the subscript N denotes the number of design points in the training data. When $\mathcal{X}$ is not discrete, the optimization search in $\mathcal{X}$ is a rather formidable task. In practice, when continuous, $\mathcal{X}$ is often discretized into a finite grid, $\boldsymbol{X}_G$, with N_G number of points. Consequently, we replace the search over $\mathcal{X}$ by a search over a set of candidate points $\boldsymbol{X}_{cand} \subseteq \boldsymbol{X}_G$.

There are also criteria based on information entropy as defined in [7]. For instance, Lindley [22] proposed that the expected change in entropy can serve as a criterion for design. This criterion has been employed by Currin et al. [8] for designing computer experiments. The *entropy* of a random vector $\bar{Y}_N = \bar{Y}[\boldsymbol{X}_N] = [Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2), \ldots, Y(\boldsymbol{x}_n)]$ with joint probability distribution $p_{\bar{Y}_n}(\boldsymbol{y})$ is defined (in bits) as

$$\mathcal{H}(\bar{Y}_n) = E[-\log_2(\bar{Y}_n)] = -\int\int\cdots\int \log_2(p_{\bar{Y}_n}(\boldsymbol{y}))p_{\bar{Y}_n}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}. \tag{2.10}$$

When $Y(\boldsymbol{x})$ is a GP with correlation matrix $\boldsymbol{K}$, we obtain the explicit entropy of $\bar{Y}_n$,

$$\mathcal{H}(\bar{Y}_n) = \frac{1}{2}\log_2(2\pi\,\mathrm{e})^n \det(\boldsymbol{K}). \tag{2.11}$$

Maximum entropy sampling [35] uses the entropy criterion to choose the subset of size N of highest entropy, that is,

$$\underset{\boldsymbol{X}_N\subset\boldsymbol{X}_{cand}}{\arg\max}\ \mathcal{H}(\bar{Y}_N), \tag{2.12}$$

wherein $\bar{Y}_N = \bar{Y}[\boldsymbol{X}_N]$. Finding the exact solution to optimization problem (2.12) is NP-hard [18].

We consider sequential designs as practical, computationally cheaper alternatives to "one shot" designs, albeit often suboptimal. The *sequential design* is defined as follows: Suppose that we have an initial design $D_n = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^n$, then for each $k = n, n+1, n+2, \ldots$ one collects an input-output pair $(\boldsymbol{x}_{k+1}, y(\boldsymbol{x}_{k+1}))$ by choosing the input values

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\boldsymbol{X}_{cand}\setminus\boldsymbol{X}_k}{\arg\max}\ f_k(\boldsymbol{x}), \tag{2.13}$$

wherein $f_k(\cdot)$ is a design criterion to be maximized. The algorithm iterates until a stopping criterion is met or the computational budget allocated is exhausted. The initial design D_n could be the empty set $\emptyset$. The sequential design allows sequential acquisition of a new design point and is called *adaptive* if $f_k(\cdot)$ exploits information provided by the known design $D_k = (\boldsymbol{X}_k, \boldsymbol{y}_k) = (\boldsymbol{x}_j, y(\boldsymbol{x}_j))_{j=1}^k$ via, for example, maximum likelihood. Their ability to adapt is why sequential designs often outperform one-stage designs such as LHDs. In our context, the covariance parameters need to be estimated. Sequential designs allow the estimates to be improved sequentially with the addition of new design points. This is especially advantageous when some input variables are considerably more influential on the output than others.

Two popular sequential designs for computer experiments are active learning MacKay (ALM) and active learning Cohn (ALC) [13]. Under the GP assumption, ALM and ALC can be viewed as sequential versions of MMSPE and IMSPE, respectively. Lam and Notz [21] developed the expected improvement for global fit (EIGF) criterion, inspired by a modified expected improvement criterion for global optimization [33]. EIGF selects the next point to be the one that maximizes the criterion $(\hat{y}_k(\boldsymbol{x}) - y(\boldsymbol{x}_*)) + \hat{s}_k^2(\boldsymbol{x})$, where $(\boldsymbol{x}_*, y(\boldsymbol{x}_*))$ is the nearest point in the current design. The EIGF criterion is used to select the points at which either the predictive variance or the difference between the prediction and the output at the nearest known design point is large. The computational complexity of EIGF is of the same order as ALM. ALM usually performs better than EIGF, although EIGF could be competitive when the output is highly nonstationary [20]. EIGF tends to do well only in special situations, in particular, when the output is constant except over a small portion of the design space [23]. As a result, we focus on ALM and ALC for comparison against MICE.

2.2.1. The ALM algorithm. At stage k in the sequential design, ALM chooses the design point $\boldsymbol{x}_{k+1}$ that maximizes the predictive variance, equation (2.6), of the GP:

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\boldsymbol{X}_{cand}}{\arg\max}\, \hat{s}_k^2(\boldsymbol{x}). \tag{2.14}$$

ALM places many points on the boundary of the design region, especially in the beginning of the selection process. Some argue that boundary points generally are less “informative” than nearby interior points; see [19]. The number of boundary points grows rapidly with the dimension size p. Suppose that we have a regular grid with N^p points; then the ratio of boundary points to the total number is $(1-(1-2/N)^p)$. For example, if $p = 4$ and $N = 10$, the ratio is about 0.59, and if $p = 6$ and $N = 10$, it is nearly 0.74.

2.2.2. The ALC algorithm. ALC chooses the design point $\boldsymbol{x}_{k+1}$ that yields the largest expected reduction in predictive variance over the design space and that is defined as

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\boldsymbol{X}_{cand}}{\arg\max} \int_{\mathcal{X}} \left(\hat{s}_k^2(\boldsymbol{x}') - \hat{s}_{k\cup\boldsymbol{x}}^2(\boldsymbol{x}')\right) \mathrm{d}\boldsymbol{x}'. \tag{2.15}$$

Standard practice is to approximate the integral over $\mathcal{X}$ with an average over a grid of N_{ref} reference points in the design space, that is,

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\boldsymbol{X}_{cand}}{\arg\max} \frac{1}{N_{ref}} \sum_{i=1}^{N_{ref}} \left(\hat{s}_k^2(\boldsymbol{x}_i) - \hat{s}_{k\cup\boldsymbol{x}}^2(\boldsymbol{x}_i)\right). \tag{2.16}$$

For each $\boldsymbol{x} \in \boldsymbol{X}_{cand}$, a Cholesky decomposition of $\boldsymbol{K}_{k\cup\boldsymbol{x}}$ is computed, resulting in a time complexity of $\mathcal{O}(N_{cand}N_{ref}k^3)$ for ALC. The computational complexity of step k in

ALC can be reduced further from $\mathcal{O}(N_{cand}N_{ref}k^3)$ to $\mathcal{O}(k^3 + N_{cand}N_{ref}k^2)$ by adopting the implementation used in [13] that is based on the following calculations. First, $\boldsymbol{K}_k^{-1}$ is obtained in $\mathcal{O}(k^3)$, and then $\boldsymbol{K}_{k\cup\boldsymbol{x}}^{-1}$ is computed in $\mathcal{O}(k^2)$ by exploiting that $\boldsymbol{K}_{k\cup\boldsymbol{x}}^{-1}$ can be expressed in terms of $\boldsymbol{K}_k^{-1}$ and $\boldsymbol{k}_k(\boldsymbol{x})$,

$$\boldsymbol{K}_{k\cup\boldsymbol{x}}^{-1} = \begin{pmatrix} \boldsymbol{K}_k^{-1} + \frac{1}{c}\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x})\boldsymbol{k}_k^T(\boldsymbol{x})\boldsymbol{K}_k^{-1}, & -\frac{1}{c}\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x}) \\ -\frac{1}{c}\boldsymbol{k}_k^T(\boldsymbol{x})\boldsymbol{K}_k^{-1}, & \frac{1}{c} \end{pmatrix}, \tag{2.17}$$

where $c = 1 - \boldsymbol{k}_k^T(\boldsymbol{x})\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x})$. Next, as shown in [13], the ALC solution can be obtained by solving the following problem in $\mathcal{O}(k^3 + N_{cand}N_{ref}k^2)$:

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\boldsymbol{X}_{cand}}{\arg\max}\ \frac{1}{N_{ref}}\sum_{i=1}^{N_{ref}}\frac{V_k^2(\boldsymbol{x},\boldsymbol{x}_i)}{\hat{s}_k^2(\boldsymbol{x})}, \tag{2.18}$$

where

$$V_k(\boldsymbol{x},\boldsymbol{x}_i) = \sigma^2\Bigg(1 - \boldsymbol{k}_k^T(\boldsymbol{x})\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x}_i) + \frac{(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{H}^T\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x}))^T(\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{H}^T\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x}_i))}{\boldsymbol{H}^T\boldsymbol{K}_k^{-1}\boldsymbol{H}}\Bigg). \tag{2.19}$$

ALC tends to provide a better global fit than ALM for a fixed design size [13, 34]. ALM, on the other hand, is easy to implement and relatively cheap computationally, and for this reason is often preferred over ALC; see, e.g., [3].

3. Mutual information for the design of computer experiments.

Mutual information is, like entropy, a classical information theoretic measure [7]. It has been used for sensor network design [4, 19], experimental design [16], and optimization [6]. This section begins with a brief account of mutual information–based design algorithms. Then in section 3.2 we propose a new sequential design algorithm based on mutual information.

Suppose that we have two random vectors $\bar{Y}$ and $\bar{Y}'$ with marginal probability density functions (pdfs) $p_{\bar{Y}}(\boldsymbol{y})$ and $p_{\bar{Y}'}(\boldsymbol{y}')$, and joint pdf $p_{\bar{Y},\bar{Y}'}(\boldsymbol{y},\boldsymbol{y}')$, Then the relationship between *mutual information* of the two vectors, denoted by $I(\bar{Y};\bar{Y}')$, and entropy can be written as follows [7]:

$$I(\bar{Y};\bar{Y}') = \mathcal{H}(\bar{Y}) - \mathcal{H}(\bar{Y}|\bar{Y}'). \tag{3.1}$$

The mutual information is equivalent to the Kullback–Leibler divergence between $p_{\bar{Y},\bar{Y}'}$ and $p_{\bar{Y}}p_{\bar{Y}'}$ [7].

$$I(\bar{Y};\bar{Y}') = \int\int\dots\int \log\left(\frac{p_{\bar{Y},\bar{Y}'}(\boldsymbol{y},\boldsymbol{y}')}{p_{\bar{Y}}(\boldsymbol{y})p_{\bar{Y}'}(\boldsymbol{y}')}\right)p_{\bar{Y},\bar{Y}'}(\boldsymbol{y},\boldsymbol{y}')\,\mathrm{d}\boldsymbol{y}\,\mathrm{d}\boldsymbol{y}', \tag{3.2}$$

with $\log(0)0 = 0$. Caselton and Zidek [4] showed that mutual information can be utilized to design sampling networks by choosing the design matrix $\boldsymbol{X}_N^* \subset \mathbb{R}^{N\times p}$ that maximizes the mutual information between $\bar{Y}[\boldsymbol{X}_N^*]$ and $\bar{Y}[\boldsymbol{X}_G\backslash\boldsymbol{X}_N^*]$, that is,

$$\boldsymbol{X}_N^* = \underset{\boldsymbol{X}_N\subset\boldsymbol{X}_{cand}}{\arg\max}\ I(\bar{Y}[\boldsymbol{X}_G\backslash\boldsymbol{X}_N];\bar{Y}[\boldsymbol{X}_N]), \tag{3.3}$$

where $\boldsymbol{X}_G$ is a discrete design space, and $\boldsymbol{X}_{cand} \subseteq \boldsymbol{X}_G$ is the set of candidate points available for selection. In other words, the objective is to select the set $\boldsymbol{X}_N^*$ that reduces the entropy over $\boldsymbol{X}_G \backslash \boldsymbol{X}_N^*$ the most. This optimization problem is NP-hard [19].

3.1. The MI algorithm. Krause, Singh, and Guestrin [19] presented an alternative that avoids the need to directly solve optimization problem (3.3), more specifically, a sequential algorithm that maximizes the difference $I(\bar{Y}[\boldsymbol{X}_k \cup \boldsymbol{x}]; \bar{Y}[\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x})]) - I(\bar{Y}[\boldsymbol{X}_k]; \bar{Y}[\boldsymbol{X}_G \backslash \boldsymbol{X}_k])$ with respect to $\boldsymbol{x} \in \boldsymbol{X}_{cand}$ at each stage k in the sequential design. By adopting the GP approach, as described in section 2, they further showed that this optimization problem can be written as

$$\underset{\boldsymbol{x} \in \boldsymbol{X}_{cand}}{\arg\max}\, \mathcal{H}(Y(\boldsymbol{x})|\bar{Y}_k) - \mathcal{H}(Y(\boldsymbol{x})|\bar{Y}_{G\backslash(k\cup\boldsymbol{x})}) = \underset{\boldsymbol{x} \in \boldsymbol{X}_{cand}}{\arg\max}\, \hat{s}_k^2(\boldsymbol{x})/\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x}), \tag{3.4}$$

since

$$\begin{aligned} \mathcal{H}(Y(\boldsymbol{x})|\bar{Y}_k) - \mathcal{H}(Y(\boldsymbol{x})|\bar{Y}_{G\backslash(k\cup\boldsymbol{x})}) &= \frac{1}{2}\log\left(2\pi\,\mathrm{e}\,\hat{s}_k^2(\boldsymbol{x})\right) - \frac{1}{2}\log\left(2\pi\,\mathrm{e}\,\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x})\right) \\ &\propto\ \hat{s}_k^2(\boldsymbol{x})/\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x}). \end{aligned} \tag{3.5}$$

Here $G\backslash(k \cup \boldsymbol{x})$ denotes $\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x})$. Note that the objective in optimization problem (3.4) has a closed-form expression. This is the greedy mutual information (MI) criterion or, in short, the MI criterion. This greedy formulation provides a constant-factor approximation of the original optimization problem (3.3) under some mild conditions [19]. More specifically, the approximation is within $1 - 1/e$ of the optimum, provided that certain regularity assumptions are satisfied and the spacing between the points in $\boldsymbol{X}_G$ is not too large (see Corollary 6 and Theorem 7 in [19]). Moreover, the proof exploits that mutual information is a submodular function [24], more specifically, that the set function $I(\bar{Y}[\boldsymbol{X}]; \bar{Y}[\boldsymbol{X}'])$ is submodular for any $\boldsymbol{X}, \boldsymbol{X}' \subseteq \boldsymbol{X}_G$, with $I(\emptyset; \bar{Y}[\boldsymbol{X}_G]) = 0$. Greedy algorithms are known to be quite efficient for submodular set functions. The *MI algorithm* proposed in [19] is given below.

MI algorithm.

```
Require: GP emulator $(\boldsymbol{h}(\cdot), K(\cdot,\cdot;\boldsymbol{\xi}))$, nugget parameter $\tau^2$, grid $\boldsymbol{X}_G$, candidate set $\boldsymbol{X}_{cand} \subseteq \boldsymbol{X}_G$, a design $\boldsymbol{X}_k \subset \boldsymbol{X}_{cand}$ of size $k$, desired design size $N$
Step 1. Let $\boldsymbol{X}_{cand} \leftarrow \boldsymbol{X}_{cand} \backslash \boldsymbol{X}_k$
Step 2. Solve $\boldsymbol{x}_{k+1} \leftarrow \underset{\boldsymbol{x} \in \boldsymbol{X}_{cand}}{\arg\max}\, \hat{s}_k^2(\boldsymbol{x};\tau^2)/\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x};\tau^2)$
Step 3. Let $\boldsymbol{X}_{k+1} \leftarrow \boldsymbol{X}_k \cup \boldsymbol{x}_{k+1}$, and $\boldsymbol{X}_{cand} \leftarrow \boldsymbol{X}_{cand} \backslash \boldsymbol{x}_{k+1}$
Step 4. If $k+1 = N$, then stop; otherwise let $k = k+1$ and go to Step 1
Output: $\boldsymbol{X}_N$
```

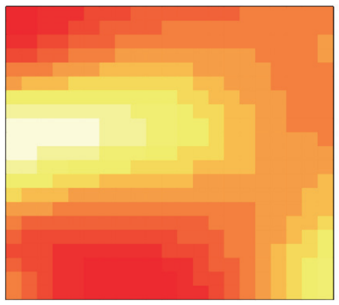
In Step 2, a GP emulator is assigned to the set of points $D_k = (\boldsymbol{X}_k, \boldsymbol{y}_k)$, and, for each $\boldsymbol{x} \in \boldsymbol{X}_{cand}$, a GP emulator is assigned to $\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x})$. The GP over $\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x})$ is required in order to estimate the difference between the total information and the information we have obtained by $\boldsymbol{X}_k \cup \boldsymbol{x}$.

Assuming the covariance is known, Krause, Singh, and Guestrin [19] demonstrated that the MI algorithm for a sensor placement problem on an equidistant mesh can achieve a good accuracy at a relatively low computational cost compared to ALC.

3.1.1. Example: A stationary Gaussian random field. As a first example, we consider a realization of a stationary Gaussian random field with zero mean, and SE covariance function with $\sigma^2 = 1$ and $\boldsymbol{\xi} = (0.8, 0.5)^T$, on a 21×21 regular grid over $[0,1]^2$. The candidate set is a regular subgrid of size 11×11. The remaining 320 design points are used to calculate the prediction accuracy. In all of our numerical examples, the prediction accuracy is measured by the normalized root MSPE (RMSPE),

$$\text{RMSPE} = \sqrt{\sum_{j=1}^{m} \frac{(y(\boldsymbol{x}_j) - \hat{y}(\boldsymbol{x}_j))^2}{m}}, \tag{3.6}$$

where the validation data set $\{\boldsymbol{x}_j\}_{j=1}^m$ consists of m input values at which the difference between the simulator output value $y(\boldsymbol{x}_j)$ and the predicted value $\hat{y}(\boldsymbol{x}_j)$ is evaluated. The *normalized* RMSPE is given by RMSPE $/(\max_j y(\boldsymbol{x}_j) - \min_j y(\boldsymbol{x}_j))$.



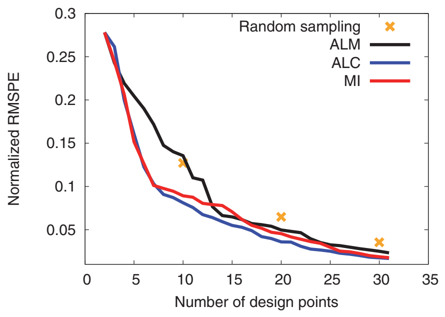


Figure 1. *Left: A realization of the stationary Gaussian random field. Right: Prediction errors.*

Figure 1 shows results obtained for ALM, ALC, and MI. The prediction errors are given as averages of 10 tries with different initial two-point designs. The average performance of random sampling over 100 tries has also been included for comparison, and, as expected, the sequential designs outperform random sampling. MI and ALC perform similarly. ALM performs the worst, partially because it systematically places most points on the boundary of the domain and, as a result, does not capture well the large variation in the interior.

3.1.2. A practical issue with the MI criterion. Krause, Singh, and Guestrin [19] showed theoretically and demonstrated empirically that the MI criterion is a promising criterion for sequential design of sensor networks on a discrete space. In computer experiments, however, the design space is generally not discrete but a compact subset of $\mathbb{R}^p$, where p can be quite large. For the MI criterion to be considered, we have to discretize $\mathcal{X}$ into a finite set $\boldsymbol{X}_G \subset \mathcal{X}$ of a grid G. This is because for each candidate point $\boldsymbol{x}^* \in \boldsymbol{X}_{cand}$, we want to assign a GP emulator over the points of a finite set $\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x}^*)$ that approximates well $\mathcal{X} \backslash (\boldsymbol{X}_k \cup \boldsymbol{x}^*)$. Recall that $\boldsymbol{X}_k$ is the set of design points at stage k of the sequential design.

We have observed that the MI criterion (3.5) is very sensitive to the distribution of points in $\boldsymbol{X}_G$. For example when the points of $\boldsymbol{X}_G$ are irregularly distributed, e.g., if some points are clustered, this criterion is unreliable. More specifically, if the criterion is evaluated at a point $\boldsymbol{x}^* \in \boldsymbol{X}_{cand}$ that is close to a point in $\boldsymbol{X}_G \backslash (\boldsymbol{X}_k \cup \boldsymbol{x}^*)$, then the denominator $\hat{s}^2_{G\backslash(k\cup\boldsymbol{x}^*)}(\boldsymbol{x}^*)$ can become very small and, as a result, produce a high MI score. In this situation, the issue is that the location of $\boldsymbol{x}^*$ in relation to the current design $\boldsymbol{X}_k$, which should be an important factor, has little influence. This issue did not present itself in [19], since the authors considered an equidistant grid.

See Figure 2 for an illustrative example where the MI criterion performs poorly. Two cases are considered: an equidistant grid $\boldsymbol{X}_G$, and $\boldsymbol{X}_G$ with an additional point $(2/3, 0.15)$, that is, $\boldsymbol{X}_G \cup \{(2/3, 0.15)\}$. A high MI score is marked in red, an intermediate score is yellow, and a low score is blue. The black dots are the points of design $\boldsymbol{X}_k$.

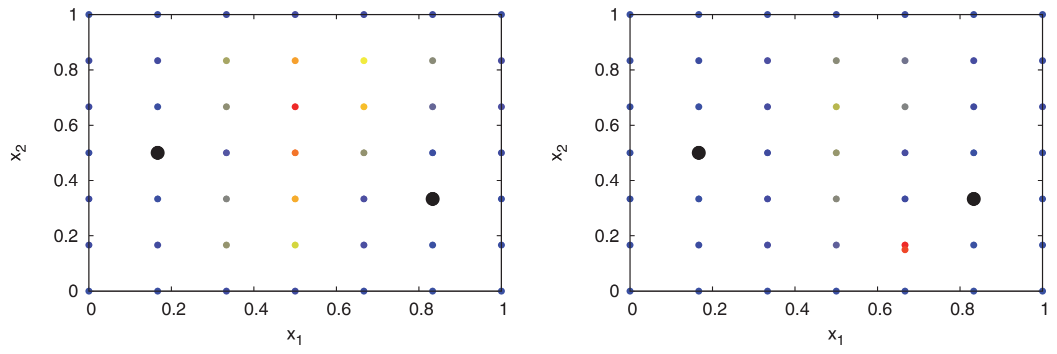


Figure 2. *Left: The score value of the MI criterion over a* 7×7 *equidistant grid. Right: The same grid with an additional point at* $(2/3, 0.15)$.

Two different choices of $\boldsymbol{X}_G$ can result in highly conflicting MI scores, as demonstrated in Figure 3 with two different maximin LHDs of size 100. Evidently, the MI criterion is not robust whenever the points are irregularly spaced. Moreover, for moderate- to high-dimensional spaces, typical of computer experiments, equidistant grids are too large to consider.

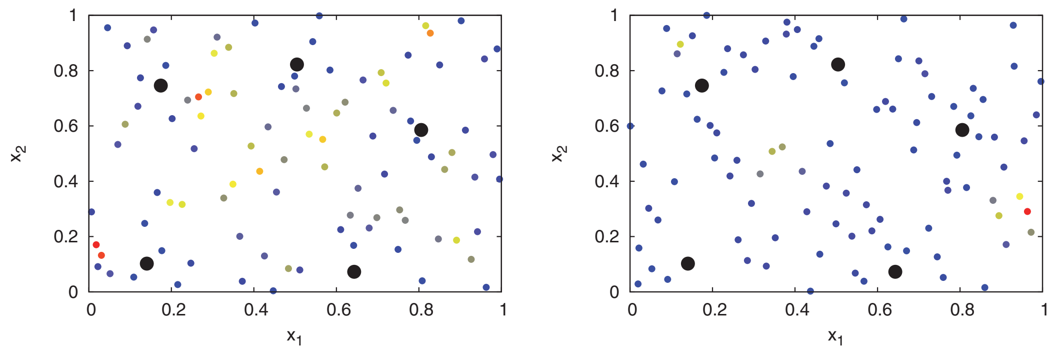


Figure 3. *Given a design* $\boldsymbol{X}_5$*, the score value of the MI criterion is displayed for two maximin LHD candidate sets of size* 100.

3.2. Mutual information for computer experiments. In this section we present a sequential design algorithm called MICE (mutual information for computer experiments) for prediction. The algorithm uses a modified MI criterion, and the correlation parameters are estimated adaptively using maximum likelihood. We also suggest that discretization $\boldsymbol{X}_G$ of $\mathcal{X}$ should not be held fixed, but instead a new $\boldsymbol{X}_G$ should be sampled at each iteration.

3.2.1. The MICE criterion: A modified MI criterion. We modify the MI criterion by introducing a parameter $\tau^2 > 0$ to the diagonal elements of the correlation matrix to smooth the prediction. Such a parameter is often called a *nugget parameter*. $\boldsymbol{K}_{\boldsymbol{\xi}}$ is replaced by $\boldsymbol{K}_{\boldsymbol{\xi},\tau^2} = \boldsymbol{K}_{\boldsymbol{\xi}} + \tau^2\mathcal{I}$, where $\mathcal{I}$ is the $n \times n$ identity matrix. A nugget parameter τ^2 is commonly used to stabilize the inversion, using the Cholesky decomposition, of a possibly ill-conditioned correlation matrix. When τ^2 is introduced to achieve numerical stability, it is usually chosen to be very small. Ranjan, Haynes, and Karsten [26] and Peng and Wu [25] suggest using a lower bound of the nugget approach to address ill-conditioning, whereas Dancik and Dorman [9] and Roustant, Ginsbourger, and Deville [29] use MLE of the nugget parameter for achieving numerical stability. Our objective is to introduce smoothing in the prediction which is critical for the proposed design criterion. Gramacy and Lee [14] argue in favor of using a nugget parameter to smooth the prediction. The BLUP model, equation (2.4), with a nonzero nugget is not a perfect interpolator of the data, and our Theorem 3.1 below clarifies the impact that a nugget parameter has on the GP emulator variance for any point in $\boldsymbol{X}$. Clearly, if $\tau^2 = 0$, $\hat{s}^2(\boldsymbol{x}_i) = 0$ for $\boldsymbol{x}_i \in \boldsymbol{X}$.

Theorem 3.1. *For a GP emulator with constant mean on $(\boldsymbol{X}, \boldsymbol{y})$, the predictive variance, equation (2.6), at any design point $\boldsymbol{x}_i \in \boldsymbol{X}$ can be written as*

$$\hat{s}^2_{\tau^2}(\boldsymbol{x}_i) = \sigma^2 \left(\tau^2 - \tau^4 \mathbf{e}_i^T (\boldsymbol{K} + \tau^2\mathcal{I})^{-1} \mathbf{e}_i + \tau^4 \frac{(\mathbf{e}_i^T (\boldsymbol{K} + \tau^2\mathcal{I})^{-1}\mathbf{1})^2}{\mathbf{1}^T (\boldsymbol{K} + \tau^2\mathcal{I})^{-1}\mathbf{1}} \right), \tag{3.7}$$

where $\tau^2 > 0$ is a nugget parameter and $\mathbf{e}_i$ is the ith unit vector.

According to Theorem 3.1, whenever $\tau^2 > 0$ is added to the correlation matrix diagonal, the variance of a GP emulator at a design point consists of terms in the order of $\sigma^2\tau^2$ and $\sigma^2\tau^4$. In practice, the nugget τ^2 is usually orders of magnitude smaller than 1. In (3.7), the magnitude of the last term in the round brackets tends to be much smaller than the other two; hence, the predictive variance is here typically smaller than $\sigma^2\tau^2$. Moreover, as τ^2 increases, the second and third terms approach τ^2 and τ^2/k, respectively, where k is the number of points in the design. This follows from the fact that as τ^2 increases, the inverse matrix reduces to $(\boldsymbol{K} + \tau^2\mathcal{I})^{-1} \approx \tau^{-2}\mathcal{I}$. Hence, if τ^2 is large enough, we can show by a simple calculation using Theorem 3.1 that $\hat{s}^2_{\tau^2}(\boldsymbol{x}_i) \approx \sigma^2\tau^2/k$ for $\boldsymbol{x}_i \in \boldsymbol{X}_k$.

In the sequential design, we define the MICE criterion as follows:

$$\boldsymbol{x}_{k+1} = \arg\max_{\boldsymbol{x} \in \boldsymbol{X}_{cand}} \hat{s}^2_k(\boldsymbol{x}) / \hat{s}^2_{G\setminus(k \cup \boldsymbol{x})}(\boldsymbol{x}; \tau_s^2), \tag{3.8}$$

where a nugget parameter $\tau_s^2 > 0$ (s for smoothing) is added to the correlation matrix $\boldsymbol{K}$ of the GP on $\boldsymbol{X}_G \setminus (\boldsymbol{X}_k \cup \boldsymbol{x})$ (in the denominator) with the specific purpose of flattening the GP's variance. The flattening of the variance is performed as a means of preventing the denominator term from being close to zero, which may happen whenever a candidate point $\boldsymbol{x}^*$

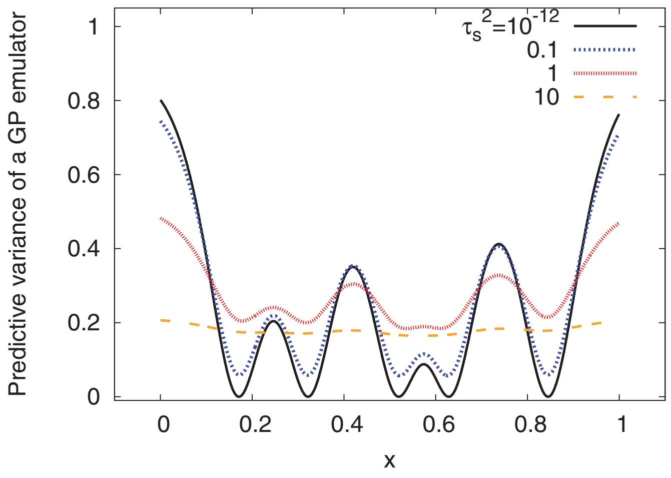


Figure 4. *The predictive variance of a GP emulator as a function of* τ_s^2 *for a one-dimensional problem in domain* $[0, 1]$.

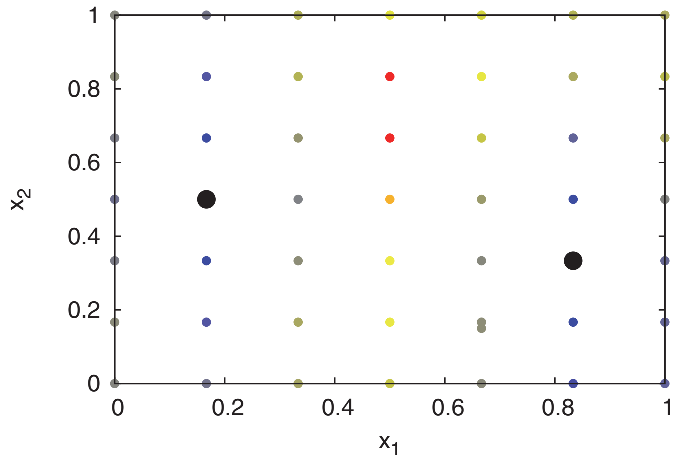


Figure 5. *The score values of the MICE criterion using* $\tau_s^2 = 1$ *for a* 7×7 *equidistant grid with an additional candidate point at* $(2/3, 0.15)$.

is too close to a point in $\boldsymbol{X}_G\backslash(\boldsymbol{X}_k \cup \boldsymbol{x}^*)$. Figure 4 shows the predictive variance for different choices of τ_s^2.

The sweet spot of τ_s^2 is around 1, where the variance is not close to 0 and the shape of the variance curve is well preserved. Hence, our default choice is $\tau_s^2 = 1$. Figures 5 and 6 show MICE scores with $\tau_s^2 = 1$, which can be compared with the corresponding figures for MI (see Figures 2 and 3, respectively). By examining the figures, we can conclude that MICE is more robust than MI. For a simple regular grid, MICE and MI perform the same.

3.2.2. Adaptivity. The original implementation of the MI algorithm assumed that the covariance is fully known, but that is rarely the case in modeling of computer experiments.

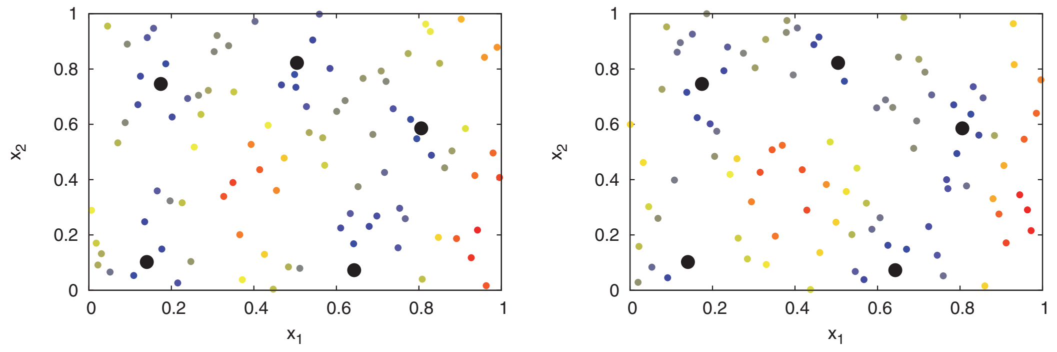


Figure 6. *Given a design* $\boldsymbol{X}_5$*, the score values of the MICE criterion using* $\tau_s^2 = 1$ *are shown for two maximin LHD candidate sets of size* 100.

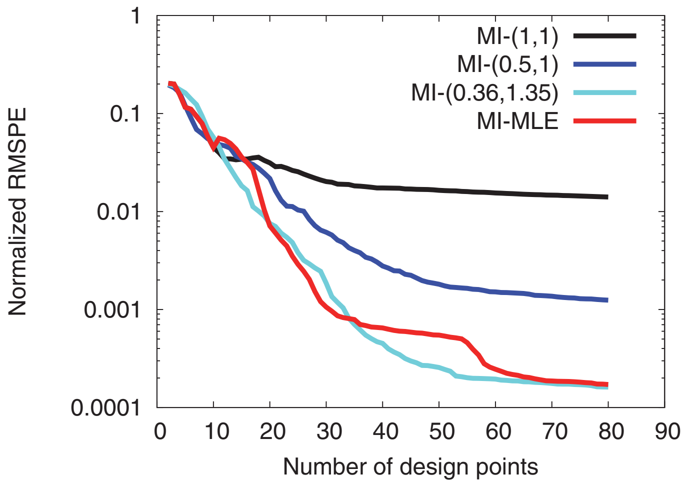


Figure 7. *Prediction errors for the MI algorithm when using estimates of* $\boldsymbol{\xi}$ *(three guesses, and one using MLE updates).*

Therefore, in our implementation, whenever the correlation parameters $\boldsymbol{\xi}$ are unknown, we provide point estimates that maximize the likelihood. This approach is described in section 2.1. The MLEs of $\boldsymbol{\xi}$ are sequentially updated at each stage k, denoted by $\hat{\boldsymbol{\xi}}_k$, by using all available input-output data. However, the updating may be skipped at some stages in order to make computational savings.

The prediction errors for different choices of estimates for the correlation parameters are shown in Figure 7, where the example is the so-called Branin function, $y(\boldsymbol{x}) = (x_2 - 5.1x_1^2/(4\pi^2) + (5/\pi)x_1 - 6)^2 + 10(1 - 1/(8\pi))\cos(x_1) + 10$, on a 21×21 regular grid over $[-5, 10] \times [0, 15] \subset \mathbb{R}^2$. A GP emulator is used with the Matérn correlation fixed at $\nu = 5/2$. Here, MI–MLE is the MI algorithm with the addition of an MLE step at each stage k of the sequential design. For $k < 10$, the tentative values $(1, 1)$ are assigned for $\boldsymbol{\xi}$. Three fixed guesses of $\boldsymbol{\xi}$ are considered: $(1, 1)$, $(0.5, 1)$, and $(0.36, 1.35)$. The latter guess is the final MLEs obtained by MI–MLE. The results show the importance of having good estimates of the correlation parameters and show that the MLE method can greatly improve upon simple guesses.

3.2.3. The MICE algorithm.

The *MICE algorithm* is outlined below with some details on some of the steps.

MICE algorithm.

```
**Require:** Function $y(\boldsymbol{x})$, GP emulator $(\boldsymbol{h}(\cdot),K(\cdot,\cdot;\boldsymbol{\xi}))$, nugget parameters $\tau^2$ and $\tau_s^2$, design space $\mathcal{X}$, initial data $(\boldsymbol{X}_k,\boldsymbol{y}_k)$, discrete set size $N_G$, candidate set size $N_{cand}$, desired design size $N$
    **Step 1.** MLE to obtain estimates $\hat{\boldsymbol{\xi}}_k$ of $\boldsymbol{\xi}$ in $K(\cdot,\cdot;\hat{\boldsymbol{\xi}}_k)$
    **Step 2.** Fit GP emulator to data $(\boldsymbol{X}_k,\boldsymbol{y}_k)$
    **Step 3.** Generate a discrete set $\boldsymbol{X}_G$ of size $N_G$, and choose a candidate set $\boldsymbol{X}_{cand}\subseteq\boldsymbol{X}_G$
    **Step 4.** Solve $\boldsymbol{x}_{k+1}=\arg\max_{\boldsymbol{x}\in\boldsymbol{X}_{cand}}\hat{s}_k^2(\boldsymbol{x};\hat{\boldsymbol{\xi}}_k,\tau^2)/\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x};\hat{\boldsymbol{\xi}}_k,\max\{\tau^2,\tau_s^2\})$
    **Step 5.** Evaluate $y_{k+1}=y(\boldsymbol{x}_{k+1})$, and let $\boldsymbol{X}_{k+1}=\boldsymbol{X}_k\cup\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}=\boldsymbol{y}_k\cup y_{k+1}$
    **Step 6.** If $k+1=N$, then stop; otherwise let $k=k+1$, and go to Step 1
    **Output:** $D_N=(\boldsymbol{X}_N,\boldsymbol{y}_N)$ of size $N$
```

In Step 3, we suggest that $\boldsymbol{X}_G$ is sampled in the design space $\mathcal{X}$, instead of keeping $\boldsymbol{X}_G$ fixed throughout. In our examples, the size of $\boldsymbol{X}_G$ is $k+N_G$, where k is the number of points of X_k. The additional N_G points are generated by picking an LHD from a set of LHDs by maximizing the minimum distance between the points in this LHD and the current design $\boldsymbol{X}_k$. In Step 4, the MICE criterion is evaluated for all $\boldsymbol{x}\in\boldsymbol{X}_{cand}$. The choice of τ_s^2 is critical; more details on this are given in section 5. Note that the parameter τ_s^2 is introduced to the GP for design $\boldsymbol{X}_G\backslash(\boldsymbol{X}_k\cup\boldsymbol{x})$, and not to the GP for design $\boldsymbol{X}_k$. However, a nugget parameter $\tau^2>0$ can still be introduced to any GP for other purposes such as achieving numerical stability (typically much smaller than $\tau_s^2=1$). We assume that the correlation parameters are the same for the GP on $\boldsymbol{X}_G\cup(\boldsymbol{X}_k\cup\boldsymbol{x})$ as those for the GP on $\boldsymbol{X}_k$.

3.2.4. Near optimality results.

Here, we provide an approximative bound of optimality for the MICE algorithm based on near optimality results in [19] for the MI algorithm under known $\boldsymbol{\xi}$. More generally, our results account for the possibility that the nugget parameter used for the GP over $\boldsymbol{X}_k$ is different than that over $\boldsymbol{X}_{\boldsymbol{G}}\backslash\boldsymbol{X}_k\cup\boldsymbol{x}$.

Theorem 3.2. *Let $Y(\boldsymbol{x})$ be a second-order stationary GP with constant mean on a compact set $\mathcal{X}\subset\mathbb{R}^p$ with a continuous correlation function $K(\boldsymbol{x},\boldsymbol{x}'):\mathcal{X}\times\mathcal{X}\to\mathbb{R}_0^+$. Assume that we have estimates $\hat{\boldsymbol{\xi}}_k$ for $\boldsymbol{\xi}$ at stage k that satisfy, for some constant $\alpha>0$, $|K(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\xi})-K(\boldsymbol{x},\boldsymbol{x}';\hat{\boldsymbol{\xi}}_k)|\le\alpha$. Then, for any $\varepsilon>0$ and any finite number N, there exists a discretization $\boldsymbol{X}_G$ of mesh width $\delta>0$ such that MICE is guaranteed to select a design $D_N=(\boldsymbol{X}_N,\boldsymbol{y}_N)$ with N design points, where $N\le 2|\boldsymbol{G}|$, for which*

$$MI(D_N)\ge(1-1/e)(OPT-N\varepsilon-2(\alpha\sigma^{-1}\tau^{-1})^2N^4(1+N^{3/2})^2-N^3\sqrt{N}|\tau_s^2-\tau^2|/\tau_s^2),$$

where e is the base of the natural logarithm, OPT is the value of the mutual information for the optimal design of size N, and τ^2 and τ_s^2 are nugget parameters in the correlation matrices for $\boldsymbol{X}_k$ and $\boldsymbol{X}_G\backslash\boldsymbol{X}_k$, respectively.

Under perfect conditions the upper bound in Theorem 3.2 guarantees a performance within 63% of the optimum. The term $N\varepsilon > 0$ is essentially zero as long as the discretization $\boldsymbol{X}_G$ is fine enough. The term $2(\alpha\sigma^{-1}\tau^{-1})^2 N^4(1+N^{3/2})^2$ is nonzero in the presence of parameter uncertainty, and the term $N^3\sqrt{N}|\tau_s^2 - \tau^2|/\tau_s^2$ appears when a nugget τ^2 is used for the GP emulator over $\boldsymbol{X}_k$. Our extension of the approximative bound of optimality to MICE reveals the effect of τ_s^2 on the performance. Our default choice $\tau_s^2 = 1$ is not causing the algorithm to diverge too much from MI, as long as τ^2 is not much larger than τ_s^2. In addition, whenever the correlation parameters are poorly estimated, the optimality bound is not sharp. To increase our understanding of the MICE behavior with respect to the choice of $\boldsymbol{X}_G$, we provide the following theorem.

Theorem 3.3. *Let $Y(\boldsymbol{x})$ be a second-order stationary GP with constant mean on a compact subset $\mathcal{X}$ of $\mathbb{R}^p$ with a Lipschitz-continuous correlation function. Then, for any $\varepsilon > 0$, there exists a regular grid $\boldsymbol{X}_G \subset \mathcal{X}$ with grid spacing $\delta = 2\varepsilon/(\sqrt{p}K_L)$ so that for any untried point $\boldsymbol{x}^* \in \mathcal{X}$ the predictive variance $\hat{s}^2_{\tau^2}(\boldsymbol{x})$ is bounded as*

$$-\tau^4 b_1(\tau^2) - \varepsilon < \sigma^{-2}\hat{s}^2_{\tau^2}(\boldsymbol{x}^*) - \tau^2 < \tau^4 b_2(\tau^2) + \varepsilon,$$

where

$$b_1(\tau^2) = \max\left\{\boldsymbol{e}_i^T(\boldsymbol{K} + \tau^2\mathcal{I})^{-1}\boldsymbol{e}_i : \boldsymbol{x}_i \in \boldsymbol{X}_G\right\}$$

and

$$b_2(\tau^2) = \max\left\{\frac{(\boldsymbol{e}_i^T(\boldsymbol{K} + \tau^2\mathcal{I})^{-1}\mathbf{1})^2}{\mathbf{1}^T(\boldsymbol{K} + \tau^2\mathcal{I})^{-1}\mathbf{1}} : \boldsymbol{x}_i \in \boldsymbol{X}_G\right\},$$

where $\mathcal{I}$ is the identity matrix, and $\boldsymbol{e}_n$ is the ith unit vector for member $\boldsymbol{x}_i$ of $\boldsymbol{X}_G$. Here K_L is the Lipschitz constant for $\hat{s}^2_{\tau^2}(\boldsymbol{x})$ over $\mathcal{X}$.

Theorem 3.3 tells us that when $\boldsymbol{X}_G$ is a regular grid dense enough in $\mathcal{X}$, while τ^2 and τ_s^2 are small enough, MICE is equivalent to ALM. In fact, MICE also behaves as ALM if $\boldsymbol{X}_G$ is more dense and τ_s^2 is large enough so that $(\boldsymbol{K} + \tau^2\mathcal{I})^{-1} = \tau^{-2}\mathcal{I}$ (approximately), since according to Theorem 3.3, as $\varepsilon > 0$ becomes arbitrarily small, $0 < \hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x}) < \varepsilon$. This can be seen in Figure 4. Nonetheless, with $\tau_s^2 = 1$, MICE is not expected to behave as ALM. Similarly, MI behaves as ALM whenever $\boldsymbol{X}_G$ is dense in $\mathcal{X}$ and τ^2 is very small. The prerequisites of Theorem 3.3 hold in our numerical tests, because both the SE correlation function and Matérn correlation with $\nu = 5/2$ are continuously differentiable (hence Lipschitz continuous) [15].

3.2.5. A computational improvement. In MICE, we compute $\hat{s}^2_{G\backslash(k\cup\boldsymbol{x})}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \boldsymbol{X}_{cand}$, which requires the Cholesky decomposition of an $(N_G-k-1)\times(N_G-k-1)$ correlation matrix $\boldsymbol{K}_{G\backslash(k\cup\boldsymbol{x})}$, where N_G is the number of points in $\boldsymbol{X}_G$. This is a computationally intensive task if N_G is large. To overcome this, we use the following implementation, which requires only a single Cholesky decomposition. First, invert the correlation matrix $\boldsymbol{K}_{G\backslash k}$. Then exploit the partitioned inverse formula for matrices in block form. That is, the inverse of

$$\boldsymbol{K}_{G\backslash k} = \begin{pmatrix} \boldsymbol{K}_* & \boldsymbol{k}^*(\boldsymbol{x}) \\ \boldsymbol{k}_*^T(\boldsymbol{x})^T & K(\boldsymbol{x},\boldsymbol{x}) \end{pmatrix} \tag{3.9}$$

can be written as

$$\boldsymbol{K}_{G\setminus k}^{-1} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{b}_{12} \\ \boldsymbol{b}_{21} & b \end{pmatrix}, \tag{3.10}$$

where $\boldsymbol{K}_* = \boldsymbol{K}_{G\setminus(k\cup\boldsymbol{x})}$ and $\boldsymbol{k}_*(\boldsymbol{x}) = \boldsymbol{k}_{G\setminus(k\cup\boldsymbol{x})}(\boldsymbol{x})$. Here $\boldsymbol{B} = \boldsymbol{K}_*^{-1} + \frac{1}{k}\boldsymbol{K}_*^{-1}\boldsymbol{k}_*(\boldsymbol{x})\boldsymbol{k}_*^T(\boldsymbol{x})\boldsymbol{K}_*^{-1}$, $\boldsymbol{b}_{12} = -\frac{1}{k}\boldsymbol{K}_*^{-1}\boldsymbol{k}_*(\boldsymbol{x})$, $\boldsymbol{b}_{21} = -\frac{1}{k}\boldsymbol{k}_*^T(\boldsymbol{x})\boldsymbol{K}_*^{-1}$, and $b = 1/(K(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}_*^T(\boldsymbol{x})^T\boldsymbol{K}_*^{-1}\boldsymbol{k}_*(\boldsymbol{x}))$. This relates $\boldsymbol{K}_{G\setminus(k\cup\boldsymbol{x})}^{-1}$ to $\boldsymbol{K}_{G\setminus k}^{-1}$ for any $\boldsymbol{x} \in \boldsymbol{X}_{G\setminus k}$ as follows: given $\boldsymbol{K}_{G\setminus k}^{-1}$, we can obtain $\boldsymbol{B}$, $\boldsymbol{b}_{12}$, $\boldsymbol{b}_{21}$, and b, directly from (3.10), and then we find that $\boldsymbol{K}_{G\setminus(k\cup\boldsymbol{x})}^{-1} = \boldsymbol{B} - \frac{1}{b}\boldsymbol{b}_{12}\boldsymbol{b}_{21}$. Therefore, $\boldsymbol{K}_{G\setminus(k\cup\boldsymbol{x})}^{-1}$ can be obtained from $\boldsymbol{K}_{G\setminus k}^{-1}$ in $\mathcal{O}((N_G - k)^2)$.

3.2.6. Example: A visualization of the design selection. Design selection with ALM, ALC, MI, and MICE on $[0,1]^2$ is shown in Figure 8. A GP emulator with a constant mean is used with a fixed Matérn covariance using $\sigma^2 = 1$, $\nu = 5/2$, and $\boldsymbol{\xi} = (0.4, 1)$. The black solid dots are design points, and the others are candidate points with the color representing the score value (red: high, blue: low) for the different design criteria. The initial design consists of the points $(0.3, 0.6)$ and $(0.7, 0.4)$. MICE with $\tau_s^2 = 1$ and ALC produce centered and well-spaced designs, whereas ALM focuses on the boundary. MI is the criterion most reluctant to select boundary points.

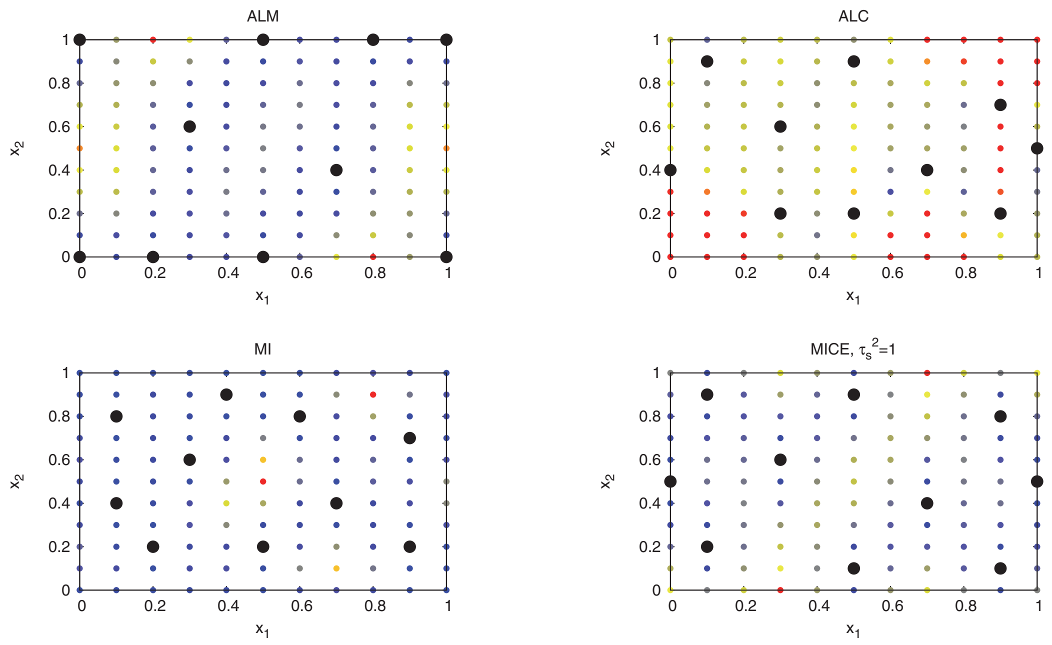


Figure 8. *Design selection with ALM, ALC, MI, and MICE on* $[0,1]^2$.

4. Computational complexity. For the sequential algorithms, the total computational cost to obtain a design $\boldsymbol{X}_N$ of size N may be divided into the cost to fit the GP emulator (in our case using MLE), the cost to generate the candidate set $\boldsymbol{X}_{cand}$, and the cost to evaluate a design criterion over the candidate points:

$$T_{total} = T_{mle} + T_{cand} + T_{select}. \tag{4.1}$$

The time complexity to compute the MLEs is in $\mathcal{O}(pN^2 + N_{mle}N^{\omega})$, where $\omega > 0$ is related to the efficiency of the algorithm for matrix inversion: for naïve Gaussian elimination $\omega = 3$, and for Strassen's algorithm $\omega = \log_2(7)$. The term pN^2 is the number of operations needed to determine the distances between distinct pairs of points in $\boldsymbol{X}$ (which is $N(N-1)/2$). The second term, $N_{mle}N^{\omega}$, is the time complexity of MLE, which is directly related to the cost of inverting the correlation matrix $\boldsymbol{K}_{\boldsymbol{\xi}}$, N_{mle} times. N_{mle} is the number of trial points visited during the optimization to find the MLEs of $\boldsymbol{\xi}$. To train the GP emulator, that is, determine the weights $\boldsymbol{\lambda}$ of the corresponding BLUP model (2.4), only matrix multiplications (each of order $\mathcal{O}\left(N^2\right)$) are required. The time to evaluate the mean of the GP emulator at an untried point is $\mathcal{O}(pN)$, and to evaluate the variance it is $\mathcal{O}\left(pN^2\right)$.

Table 1
Time complexity for ALM, ALC, and MICE.

Algorithm	Total time complexity for design size N
ALM	$\mathcal{O}\left(N_{mle}N^{1+\omega} + N_{cand}pN^2\right)$
ALC	$\mathcal{O}\left(N_{mle}N^{1+\omega} + N_{cand}N_{ref}pN^3\right)$
MICE	$\mathcal{O}\left(N_{mle}N^{1+\omega} + N(N_G - N)^{\omega} + N_{cand}pN^3 + N_{cand}Np(N_G - N)^2\right)$

The computational complexity for the different algorithms is presented in Table 1. For ALC, we have adopted formulation (2.18), which is the formulation with lowest computational cost. The time complexity for a single ALM step is $\mathcal{O}\left(pk^2 + N_{mle}k^{\omega} + N_{cand}pk\right)$, where k is the current design size. The total cost for ALM is $\mathcal{O}\left(pN^3 + N_{mle}N^{1+\omega} + N_{cand}pN^2\right)$, where N is the final design size. N_{ref} is specific to ALC and is the number of reference points used for averaging over the design space.

Usually, $N_{ref} \propto N_{cand} \propto N$, $N_G \propto N$, and $\omega = 3$. The expressions in Table 1 can thus be written as $\mathcal{O}\left(N_{mle}N^4 + pN^3\right)$ for ALM, $\mathcal{O}\left(N_{mle}N^4 + pN^5\right)$ for ALC, and $\mathcal{O}\left(N_{mle}N^4 + pN^4\right)$ for MICE. Observe that ALM has a much lower computational complexity than the others, and ALC is computationally prohibitive for large N. MICE is computationally cheaper than ALC as long as the ratio $(N_G - N)/N$ is not too large. In the computer experiment setting, N_G can be chosen to be not too large for computational convenience.

Because the maximum likelihood often is the most expensive step, a reduction in cost can be achieved by updating only the MLEs of $\boldsymbol{\xi}$ at every ith step for some number i. This tends to reduce T_{mle} substantially, giving MICE a significant advantage over ALC. A running time comparison between the algorithms is given in Figure 9.

The cost to generate candidate sets varies depending on the choice of sampling technique and the desired size. For instance, minimax designs are more computationally intensive than maximin designs [1].

5. A numerical comparison.

In this section, we present a numerical comparison between MICE, ALM, and ALC to better understand, as well as compare, the different sequential designs. We also consider MmLHD, which is a maximin-distance design within the class of LHDs, and mMLHD, a minimax-distance design within the class of LHDs. Note that Mm

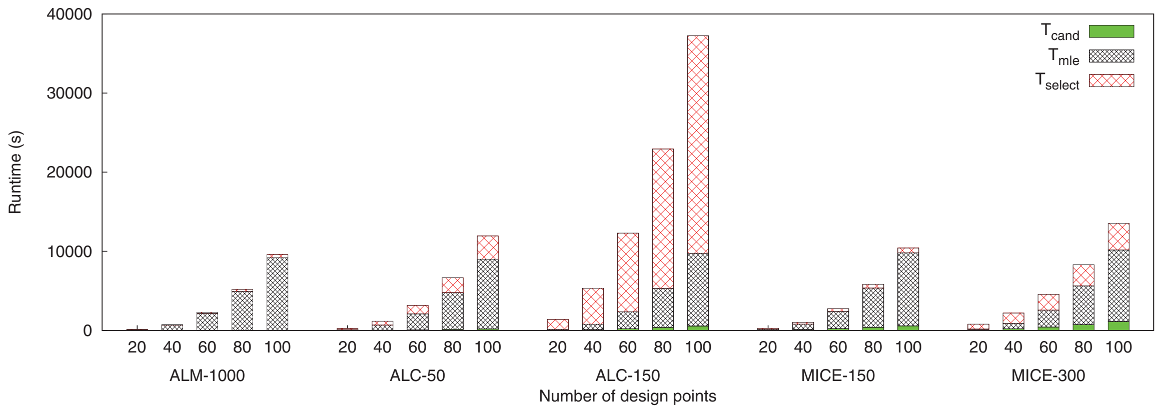


Figure 9. *Running time of the different sequential design algorithms for selecting designs of different fixed sizes. The study is on the oscillatory function over* $[0,1]^4$.

stands for maximin, and mM for minimax. MmLHDs tend to cover the parameter space better than Mm-distance designs, which are not restricted to the class of LHDs, but at the expense of lower Mm-distance scores. Hence, MmLHD can be seen as a compromise between Mm- and mM-distance designs [1]. MmLHD and mMLHD select an LHD from a pool of 1000 LHDs. The mM-distance is measured using 1000 reference points over $\mathcal{X}$ on an LHD.

When training the GP emulator, all of the input variables are scaled to lie in $[0,1]^p$, and all of the outputs are scaled to have zero mean and unit variance. The computational budget is limited to 150 design points. This budget is reasonable in realistic simulations where resolution is high. The metric of prediction accuracy is primarily the empirical RMSPE, equation (3.6), against design size. The RMSPE is calculated over a 1000-point LHD. The test functions have been selected to cover different input dimension sizes and difficulty levels. The results are presented as averages of 10 replicates. For each replication, a different initial design is used, consisting of two points sampled using mMLHD. All methods are compared using the same initial designs. The MmLHD and mMLHD results are averages of 10 tries and calculated for design sizes 50, 75, 100, and 120. The actual runtime is another factor that must be considered.

A stationary GP with a Matérn covariance with $\nu = 5/2$ is used in all examples. Because the size of the candidate set has such a significant effect on the results, the number of candidate points is included in the method names; for example, we denoted MICE with $N_{cand} = 150$ by MICE-150. With ALC, the computational cost, with respect to N_{cand}, is substantially higher than with ALM and MICE. Hence, for ALM, we consider $N_{cand} = 1000$ for MICE $N_{cand} = 150, 300$, and $N_{cand} = 150$ for ALC. ALM is kept at $N_{cand} = 1000$ because its algorithm cost is low. The candidate sets are LHDs, selected based on the maximin criterion with respect to the current design.

The remaining parameters are specified as $N_{ref} = N_{cand}$ for ALC, as used in [13, 34], and $\tau_s^2 = 1$ for MICE. We have also included results for a range of different choices of τ_s^2, in particular, $\tau_s^2 = 10^{-12}$ which behaves as the MI algorithm, since $\tau_s^2 \approx \tau^2$.

The optimizer employed for the MLE method is a real-coded genetic algorithm [10] with

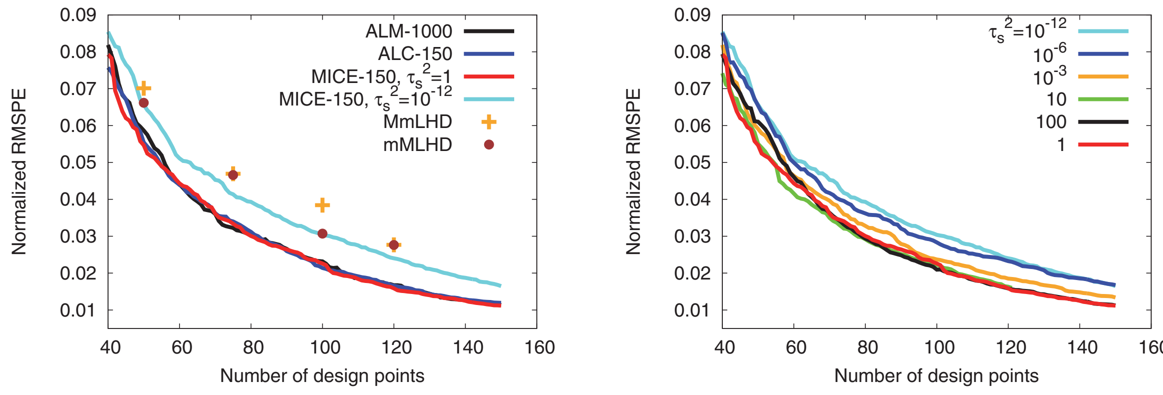


Figure 10. *Left: Comparison between algorithms for the oscillatory function over* $[0,1]^4$. *Right: The performance with MICE*-150 *for different choices of* τ_s^2.

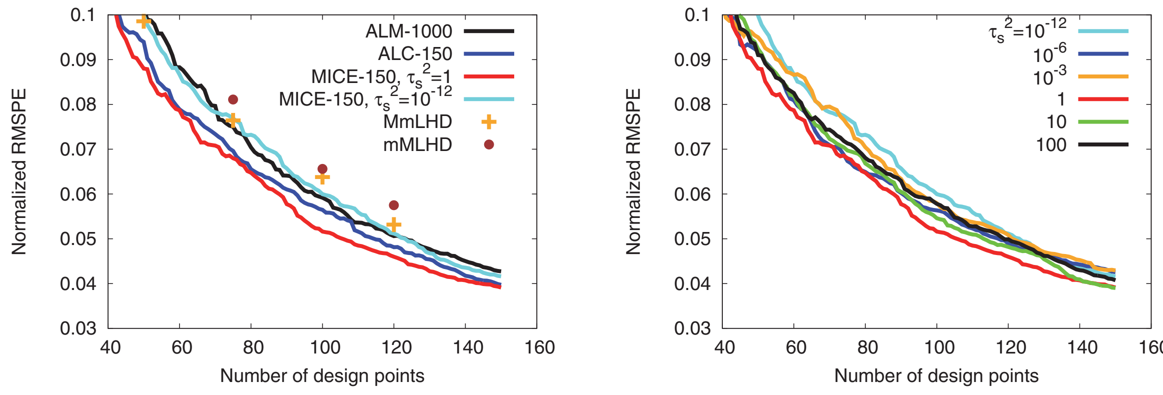


Figure 11. *Left: Comparison between algorithms for the oscillatory function over* $[0,1]^8$. *Right: The performance with MICE*-150 *for different choices of* τ_s^2.

settings that require 1024 calls to the log-likelihood. The values for the uncertain correlation parameters are fixed until the current design is of a specific size (20 if $p > 4$, else 10).

5.1. Alan Genz's oscillatory function. The "oscillatory" function belongs to a family of test functions [12] proposed by Alan Genz for the study of quadrature methods. The function is $y(\boldsymbol{x}) = \cos(\boldsymbol{c} \cdot \boldsymbol{x} + 2\pi w), \boldsymbol{x} \in [0,1]^p$. The vector $\boldsymbol{c} = (c_1, c_2, \ldots, c_p)$ determines the level of difficulty along the different directions of $\mathcal{X} \subset \mathbb{R}^p$, and w is the displacement. To study the impact of dimension size p on the difficulty of predicting untried points, $\boldsymbol{c}$ is constrained as $\sum_{i=1}^p c_i = h, c_i > 0$, where h can be held fixed in order to maintain the difficulty level of the problem for different choices of p. Two case examples are considered: $\boldsymbol{c} = (1.85, 2.51, 1.94, 2.70)^T$ and $w = 0.43$ over $[0,1]^4$, and $\boldsymbol{c} = (0.14, 1.69, 0.81, 1.73, 2.10, 0.42, 0.14, 1.97)$ and $w = 0.4$ over $[0,1]^8$, where $h = 9$.

As can be observed in Figures 10 and 11, the sequential designs outperform those based on LHDs. As expected, MmLHD and mMLHD, even if well spaced, do not take into account that $y(\boldsymbol{x})$ is anisotropic. The worst performing sequential design is MICE-150 with $\tau_s^2 = 10^{-12}$, which in fact uses the MI criterion. The poor performance is due to the issue discussed in section 3.1.2. In the four-dimensional case, ALM-1000, ALC-150, and MICE-150 produce similar results in terms of prediction error, but, as shown in Figure 9, the time to run ALC is significantly higher than for ALM and MICE, which in many cases makes it the least favorable, especially if $y(\boldsymbol{x})$ is cheaper to evaluate. Even if one assumes that the less costly ALC-50 would produce a similar performance as ALC-150, it would still not be competitive in this case. Observe that $\tau_s^2 = 1$ performs the best.

5.2. Piston simulation function. Here we consider a seven-dimensional example from [2], where the output describes the circular motion of a piston within a cylinder; it obeys the following equations:

$$y(\boldsymbol{x}) = 2\pi\sqrt{\frac{x_1}{x_2 + x_3^2\frac{x_4x_5}{x_6}\frac{x_7}{g_1(\boldsymbol{x})}}}, \qquad \text{where} \quad g_1(\boldsymbol{x}) = \frac{x_3}{2x_2}\left(\sqrt{g_2^2(\boldsymbol{x}) + 4x_2\frac{x_4x_5}{x_6}x_7} - g_2(\boldsymbol{x})\right),$$
$$g_2(\boldsymbol{x}) = x_3x_4 + 19.62x_1 - \frac{x_2x_5}{x_3}.$$

Here, $y(\boldsymbol{x})$ is the cycle time(s) which varies with seven input variables. The design space is given by $x_1 \in [30, 60]$ (piston weight, kg), $x_2 \in [1000, 5000]$ (spring coefficient, N/m), $x_3 \in [0.005, 0.020]$ (piston surface area, m^2), $x_4 \in [90000, 110000]$ (atmospheric pressure, N/m^2), $x_5 \in [0.002, 0.010]$ (initial gas volume, m^3), $x_6 \in [340, 360]$ (filling gas temperature, K), and $x_7 \in [290, 296]$ (ambient temperature, K). The nonlinearity makes this deterministic computer experiment problem challenging to emulate. MICE-300 yields a slight improvement over MICE-150; see Figure 12. MICE with 300 candidate points is not that much more expensive than with 150; in fact, it is significantly cheaper computationally than ALC with 150. Again, the proposed algorithm MICE performs the best. For high-dimensional problems, ALM tends to be the worst, probably due to the high percentage of points on the boundary.

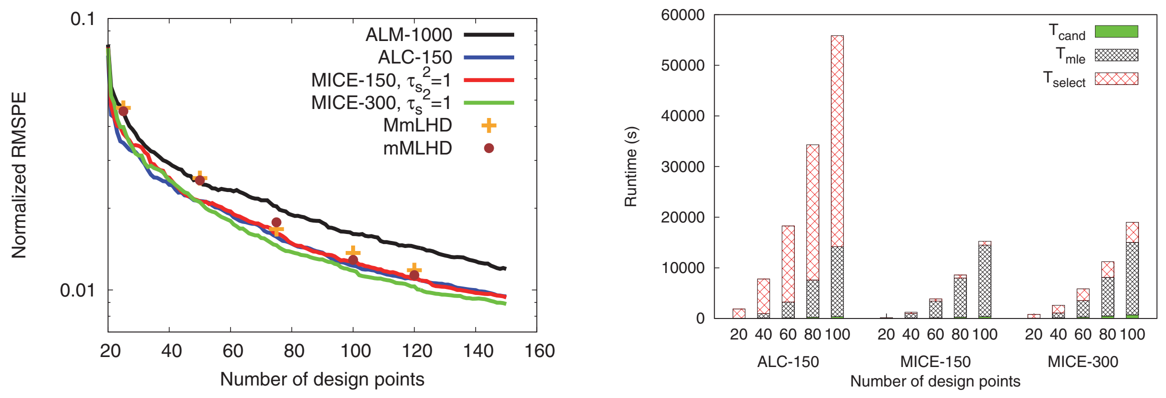


Figure 12. *Results for the seven-dimensional Piston simulation function.*

6. Application to a tsunami simulator. There is a pressing need in tsunami modeling for uncertainty quantification, with the specific purpose of providing accurate risk maps or issuing informative warnings. Sarri, Guillas, and Dias [32] were the first to demonstrate that statistical emulators can be used for these purposes. Recently, Sraj et al. [37] studied the propagation of uncertainty in Manning's friction parameterization to the prediction of sea surface elevations for the Tōhoku 2011 tsunami event. They used a polynomial chaos (PC) expansion as the surrogate model of a low resolution tsunami simulator. Note that Bilionis and Zabaras [3] showed that GP emulators can outperform PC expansions when small- to moderate-sized training data are considered. Stefanakis et al. [38] used an active experimental design approach for optimization to study whether small islands can protect nearby coasts from tsunamis.

We consider here the problem of predicting the maximum free-surface elevation of a tsunami wave at the shoreline, for a wide range of scenarios, following a subaerial landslide at an adjoining beach across a large body of shallow water. A tsunami wave simulator is used. A landslide of seafloor sediment, initially at the beach, has a Gaussian shaped mass distribution and generates tsunami waves that propagate toward the opposite shoreline across from the beach (see Figure 13). The seafloor bathymetry is changing over time and is used as input to the tsunami simulator. The floor motion is described by the change in bathymetry of the sloping beach over time, $h(x,t) = H(x) - h_0(x,t)$, where $H(x) = x \tan\beta$ is the static uniformly sloping beach, and $h_0(x,t) = \delta \exp\left(-(\tilde{x} - \tilde{t})^2\right)$ is the perturbation with respect to $H(x,t)$. Here $\tilde{x} = 2\frac{x\mu^2}{\delta \tan\phi_1}$, $\tilde{t} = \sqrt{\frac{g}{\delta}}\mu t$, δ is the maximum vertical slide thickness, μ is the ratio of the thickness and the slide length, and $\tan\phi_1$ is the beach slope. The free-surface elevation is defined as $z(x,t) = -h(x,t)$. It is assumed that the initial water surface is undisturbed, that is, $z(x,0) = 0$ for all x. The slope $\tan\phi_2$ of the beach at the opposite shoreline is chosen so that the distance between the shorelines is 2800 m. This is a shallow water problem, which means that $\tan\phi_1 \ll 1$ and that the translating mass movement is thin ($\mu = \delta/L \ll 1$).

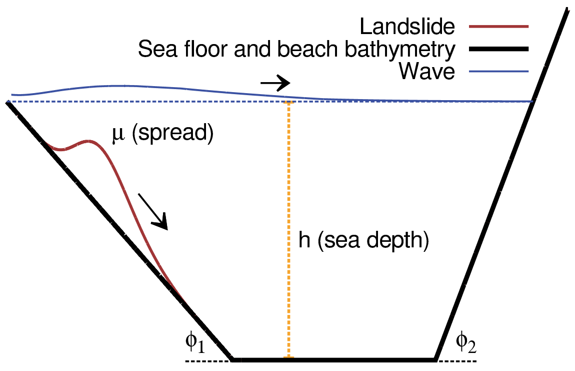


Figure 13. *Case example: landslide-generated tsunami event.*

We use the state-of-the-art numerical code VOLNA [11] to simulate all stages of this landslide-generated tsunami event, based on nonlinear shallow water equations. We run VOLNA on a single GPU on the cluster *Emerald*. The bathymetry defined above is given along only one spatial coordinate, but in the code implementation of VOLNA a second spatial dimension (in this case, along the shoreline) is added to cover 10 m of shoreline. The mesh is defined on $[-5, 5] \times [0, 3000]$ (m^2) and consists of 312,016 triangular elements.

We demonstrate the efficiency of the different sequential design methods for the design of

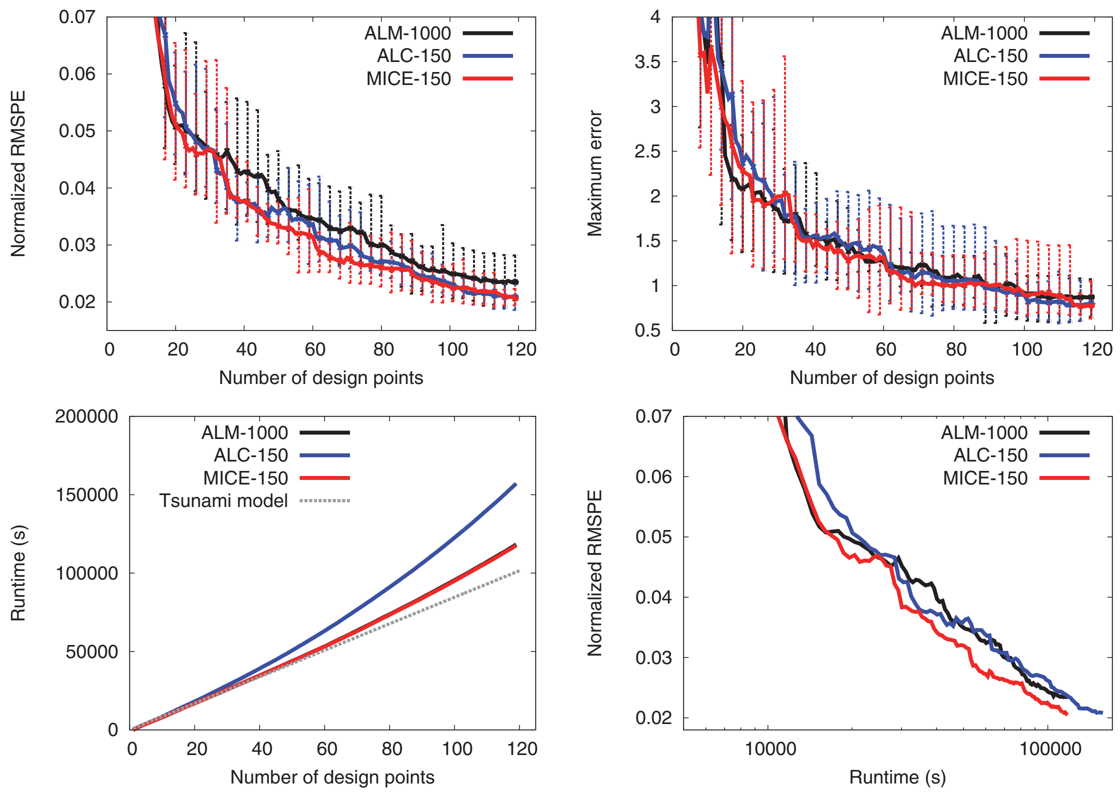


Figure 14. *Results for a simple tsunami model. Note the log-scale in the lower right figure.*

a realistic computer experiment. This problem is inspired by a benchmark problem given at the Catalina 2004 workshop[1] on long-wave runup models used in the validation of tsunami models. We consider four input parameters for emulation: $\phi_1 \in [35°, 70°]$, $\phi_2 \in [35°, 70°]$, $h \in [500.0, 1000.0]$, and $\mu = [0.01, 0.1]$.

Some of the method-specific parameters are $N_{ref} = 150$ (ALC) and $\tau_s^2 = 1$ (MICE). $N_{cand} = 150$ is used for ALC and MI, and since ALM is relatively cheap computationally with respect to N_{cand}, we let $N_{cand} = 1000$ for ALM. The results are averages of 10 runs. As before, the GPs have a constant mean and use the Matérn covariance $\nu = \frac{5}{2}$. A hold-off set of size 500 is used to calculate the normalized RMSPE and the maximum error.

In Figure 14 we observe that MICE performs better than ALM and ALC when considering the actual runtime. ALM is more competitive when the objective is to minimize the maximum error, since it places most points on the boundary where the largest prediction errors often are located. The maximum prediction error, over the designs of size 120, is 1 m or less in sea surface elevation for waves up to almost 10 m. Note that in the bottom right figure the total runtime is given in logarithmic scale with base 10, and the computational savings are ∼10–20% using MICE or are greater if the MLE method is applied more sparsely, as it dominates the MICE cost. A single run of VOLNA takes on average 850 seconds. The time

[1]http://isec.nacse.org/workshop/2004_cornell/background.html.

consumed by the simulator is represented by a gray dashed line in Figure 14 (bottom left figure).

For a more realistic tsunami scenario, with more parameters, the convergence toward a good fit of the GP will be slower. Hence, ALC will become relatively much more costly than MICE, as ALC's cost increases steeply with the number of runs. Since each additional run will help gain a lot of precision, we expect that MICE will outperform ALC and ALM even more in such scenarios.

7. Conclusion.

In this paper we introduced a new mutual information-based design criterion, MICE, to find a design for good overall prediction in computer experiments. The MICE algorithm is particularly attractive in terms of time complexity of the entire design process. Our numerical studies show that for a good range of test functions, and a realistic tsunami simulator, MICE is able to outperform popular methods such as ALC, ALM, and LHD. In addition, MICE may outperform other designs even more (we conjecture around 50–70% more after examining our computational summaries above, depending on the other relative costs) with less frequent updates of the MLE (e.g., every 5–10 steps) of the correlation parameters; this is something to investigate in the future in practical implementations. Our theoretical results also improve our understanding of the nugget parameter on the variance estimation, which is a key ingredient in MICE.

In this paper we investigate the computational costs of the algorithms considered. The computational costs of the sequential design algorithms matter when the simulator is neither very cheap (no need for sequential design) nor extremely expensive (the cost of any algorithm is then negligible). This is generally the case in uncertainty quantification studies, as models are run at a high fidelity level, but not at their highest level, in order to allow for exploration of the input's influences on the outputs. If the cost of the sequential design algorithm is of the same order of magnitude as the simulator (or say 10–100 times less), then gains can be readily made by running the simulator more times, and the gain is even more when the cost of the algorithms increases steeply with design size. Furthermore, it is typical for a research project to be awarded a certain number of hours on a cluster, and thus computational complexity will increase accuracy under the same budget conditions. Another recurrent issue is that clusters are shared among many research projects, often at the local or national level. The queuing time becomes an issue as sometimes there is no cluster configuration that can accommodate the run at the time of job submission. Note that for well parallelized simulators (e.g., climate, fluid dynamics, and tsunami models), the queuing time on a busy cluster can be in the order of hours, or even days in some instances. By having a performance sequential design strategy, the queuing time can be reduced—sometimes dramatically in case of sudden bottlenecks—by running the simulator fewer times for the same accuracy.

Finally, further extensions of MICE would be welcome. One such extension would be a MICE algorithm in a nonstationary setting, for example, in the treed GP form [13] in which subdomains of the input space, where the input-output relationships are different, are identified and the sampling is carried out to account for this behavior. Another possible extension would be to account for multiple outputs in terms of spatial location or behavior. Also, the desire to screen active variables along the sequential design would constitute another extension for models whose large number of variables needs to be reduced before, for instance, carrying out uncertainty quantification tasks.

Appendix A. Proofs of the theorems.

Proof of Theorem 3.1. Given a GP emulator on $D_k = (\boldsymbol{X}_k, \boldsymbol{y}_k)$ with constant mean and a fixed correlation matrix with a nugget parameter τ^2, the predictive variance for any point $\boldsymbol{x}_i \in \boldsymbol{X}_k$ can be written as

$$\begin{aligned}\hat{s}^2_{\tau^2}(\boldsymbol{x}_i) = \sigma^2((1 - \boldsymbol{k}^T(\boldsymbol{x}_i))\left(\boldsymbol{K} + \tau^2\mathcal{I}\right)^{-1}\boldsymbol{k}(\boldsymbol{x}_i)\\ + (\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\boldsymbol{k}(\boldsymbol{x}_i)) - 1)^2/(\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1})),\end{aligned}$$

where $\mathcal{I}$ is the $k \times k$ identity matrix; then

$$\begin{aligned}\boldsymbol{k}^T(\boldsymbol{x}_i)\left(\boldsymbol{K}+\tau^2\mathcal{I}\right)^{-1}\boldsymbol{k}(\boldsymbol{x}_i) &= \boldsymbol{k}^T(\boldsymbol{x}_i)\left(\boldsymbol{K}+\tau^2\mathcal{I}\right)^{-1}(\boldsymbol{k}(\boldsymbol{x}_i)+\tau^2\mathbf{e}_i) - \tau^2\boldsymbol{k}^T(\boldsymbol{x}_i)\left(\boldsymbol{K}+\tau^2\mathcal{I}\right)^{-1}\mathbf{e}_i\\ &= \boldsymbol{k}^T(\boldsymbol{x}_i)\mathbf{e}_i - \tau^2\boldsymbol{k}^T(\boldsymbol{x}_i)(\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1})^T\\ &= 1 - \tau^2\boldsymbol{k}^T(\boldsymbol{x}_i)(\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1})^T\\ &= 1 - \tau^2\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\boldsymbol{k}(\boldsymbol{x}_i)\\ &= 1 - \tau^2\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\left(\boldsymbol{k}(\boldsymbol{x}_i)+\tau^2\mathbf{e}_i-\tau^2\mathbf{e}_i\right)\\ &= 1 - \tau^2 + \tau^4\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{e}_i,\end{aligned}$$

where $\mathbf{e}_i$ is the ith unit vector. Similarly, $\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\boldsymbol{k}(\boldsymbol{x}_i) = 1 - \tau^2\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1}$. Insert these results into $\hat{s}^2_{\tau^2}(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i \in \boldsymbol{X}_k$, and obtain

$$\hat{s}^2_{\tau^2}(\boldsymbol{x}_i) = \sigma^2\left(\tau^2 - \tau^4\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{e}_i + \tau^4\frac{(\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1})^2}{\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1}}\right). \qquad \blacksquare$$

Proof of Theorem 3.2. This proof follows closely the proofs of Lemma 5 and Theorem 7 in [19]. Let us suppose that $\boldsymbol{X}_{G_1} \subset \mathcal{X}$ and $\boldsymbol{X}_{G_2} \subset \mathcal{X}$ are equidistant grids with spacing 2δ, for some $\delta > 0$, and that $\boldsymbol{X}_{G_2}$ is obtained by translating $\boldsymbol{X}_{G_1}$ by distance δ in Euclidean norm. $\boldsymbol{X}_{G_1}$,$\boldsymbol{X}_{G_2}$ are assumed to cover $\mathcal{X}$ in terms of compactness. In the context of experimental design, let us consider $\boldsymbol{X}_{G_1}$ to be the set of points available for selection. For a design point $\boldsymbol{x}$ in $\boldsymbol{X}_{G_1}$, we denote by $\tilde{\boldsymbol{x}}$ the corresponding point in $\boldsymbol{X}_{G_2}$, that is, $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \geq \delta$ for all $\boldsymbol{x} \in \boldsymbol{X}_{G_1}$. Let us denote by $\bar{Y}_1, \bar{Y}_2$ the restriction of the GPs to $\boldsymbol{X}_{G_1}, \boldsymbol{X}_{G_2}$, respectively, and, for a random variable $\cdot$ in $\bar{Y}_1$, we denote by $\tilde{\cdot}$ the corresponding translated random variable in $\bar{Y}_2$. Also, $\mathcal{X}$ is compact and $K(\cdot,\cdot)$ is continuous; hence, $|K(\boldsymbol{x},\boldsymbol{x}') - K(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{x}}')| \leq \varepsilon_1$ for all $\boldsymbol{x},\boldsymbol{x}' \in \boldsymbol{X}_{G_1}$ ($K(\cdot,\cdot)$ uniformly continuous over $\mathcal{X}$). Let $\boldsymbol{X}_k$ be a subset of $\boldsymbol{X}_{G_1}$. For any $\boldsymbol{x} \in \boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k$, we assume that $\mathcal{H}(\boldsymbol{x}|\boldsymbol{X}_k) \geq \mathcal{H}(\boldsymbol{x}|\tilde{\boldsymbol{X}}_k)$ for $|\boldsymbol{X}_k| \leq 2N$, which is empirically justified in [19].

Let $\boldsymbol{X}_k$ be a subset of $\boldsymbol{X}_{G_1}$, and consider a GP on $D_k = (\boldsymbol{X}_k, \boldsymbol{y}_k)$ with a nugget parameter $\tau_1^2 > 0$, and a GP emulator on $\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k \subseteq \boldsymbol{X}_{G_2}$ with a nugget $\tau_2^2 > 0$. First, let us determine an upper bound for $|\hat{s}^2_k(\boldsymbol{x}) - \hat{s}^2_{G_1\backslash k}(\boldsymbol{x})|$:

$$\begin{aligned}|\hat{s}^2_k(\boldsymbol{x}) - \hat{s}^2_{G_1\backslash k}(\boldsymbol{x})| &= \sigma^2|\boldsymbol{k}_k^T(\boldsymbol{x})\boldsymbol{K}_k^{-1}\boldsymbol{k}_k(\boldsymbol{x}) - \boldsymbol{k}^T_{G_1\backslash k}(\boldsymbol{x})\boldsymbol{K}^{-1}_{G_1\backslash k}\boldsymbol{k}_{G_1\backslash}(\boldsymbol{x})|\\ &\leq \sigma^2(\|\boldsymbol{k}_k^T(\boldsymbol{x}) - \boldsymbol{k}^T_{G_1\backslash k}(\boldsymbol{x})\|_2\|\boldsymbol{K}_k^{-1}\|_2(\|\boldsymbol{k}_k(\boldsymbol{x})\|_2 + \|\boldsymbol{k}_{G_1\backslash k}(\boldsymbol{x})\|_2)\\ &\quad + \|\boldsymbol{k}_k^T(\boldsymbol{x})\|_2\|\boldsymbol{K}_k^{-1} - \boldsymbol{K}^{-1}_{G_1\backslash k}\|_2\|\boldsymbol{k}^T_{G_1\backslash k}(\boldsymbol{x})\|_2).\end{aligned}$$

Since $K(\cdot,\cdot)$ is uniformly continuous over $\mathcal{X}$, we know that for all $\varepsilon_1 > 0$ there exists a spacing $\delta > 0$ such that, for $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \le \delta$, $|K(\boldsymbol{x},\boldsymbol{x}') - K(\boldsymbol{x},\boldsymbol{x}')| \le \varepsilon_1$ for $\boldsymbol{x} \neq \boldsymbol{x}'$, and $\|\boldsymbol{K}_{k,\tau_1^2} - \boldsymbol{K}_{G_1\backslash k,\tau_2^2}\|_2 \le \sqrt{N}N\varepsilon_1 + \sqrt{N}|\tau_1^2 - \tau_2^2|$. We also derive $\|\boldsymbol{k}_k^T(\boldsymbol{x}) - \boldsymbol{k}_{G_1\backslash k}^T(\boldsymbol{x})\|_2 \le \varepsilon_1\sqrt{N}$, and similarly, $\|\boldsymbol{k}^T(\boldsymbol{x})\|_2 \le C\sqrt{N}$, where $\|\cdot\|_2$ is the Euclidean norm, and $C = \max_{\boldsymbol{x}\in\mathcal{X}} K(\boldsymbol{x},\boldsymbol{x})$. We assume, without loss of generality, that $C = 1$. Furthermore,

$$\begin{aligned}\|\boldsymbol{K}_{k,\tau_1^2}^{-1} - \boldsymbol{K}_{G_1\backslash k,\tau_2^2}^{-1}\|_2 &= \|\boldsymbol{K}_{k,\tau_1^2}^{-1}(\boldsymbol{K}_{k,\tau_1^2} - \boldsymbol{K}_{G_1\backslash k,\tau_2^2})\boldsymbol{K}_{G_1\backslash k,\tau_2^2}^{-1}\|_2 \\ &\le \|\boldsymbol{K}_{k,\tau_1^2}^{-1}\|_2\|\boldsymbol{K}_{k,\tau_1^2} - \boldsymbol{K}_{G_1\backslash k,\tau_2^2}\|_2\|\boldsymbol{K}_{G_1\backslash k,\tau_2^2}^{-1}\|_2 \\ &\le (1+\tau_1^2)^{-1}(1+\tau_2^2)^{-1}\sqrt{N}\left(N\varepsilon_1 + |\tau_1^2 - \tau_2^2|\right) \le \sqrt{N}N\varepsilon_1 + \sqrt{N}|\tau_1^2 - \tau_2^2|,\end{aligned}$$

where we used that $\boldsymbol{K}$ is positive semidefinite, which means that $\|\boldsymbol{K}^{-1}\|_2 = \lambda_{min}(\boldsymbol{K})^{-1} \le (1+\tau^2)^{-1}$, where $\lambda_{min}(\boldsymbol{K})$ is the smallest eigenvalue. We thus obtain the following bound:

$$\begin{aligned}|\hat{s}_{k,\tau_1^2}^2(\boldsymbol{x}) - \hat{s}_{G_1\backslash k,\tau_2^2}^2(\boldsymbol{x})| &\le \sigma^2(2\varepsilon_1 N(1+\tau_1^2)^{-1} + N(1+\tau_1^2)^{-1}(1+\tau_2^2)^{-1}\sqrt{N}(N\varepsilon_1 + |\tau_1^2 - \tau_2^2|)) \\ &\le \sigma^2(2\varepsilon_1 N + N\sqrt{N}(N\varepsilon_1 + |\tau_1^2 - \tau_2^2|)).\end{aligned}$$

Then, for any $\varepsilon > 0$ we can choose the grid spacing $\delta > 0$ such that $\varepsilon \ge \varepsilon_1\tau_2^2\sigma^2 N(2N + N^{3/2})$. Hence, $|\hat{s}_{k,\tau_1^2}^2(\boldsymbol{x}) - \hat{s}_{G_1\backslash k,\tau_2^2}^2(\boldsymbol{x})| \le \varepsilon\tau_2^2 + \sigma^2 N^{3/2}|\tau_1^2 - \tau_2^2|$, and, in turn,

$$\begin{aligned}\mathcal{H}_{\tau_1^2}(\boldsymbol{x}|\boldsymbol{X}_k) - \mathcal{H}_{\tau_2^2}(\boldsymbol{x}|\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k) &= \frac{1}{2}\log\left(\frac{\hat{s}_{k,\tau_1^2}^2(\boldsymbol{x})}{\hat{s}_{k,\tau_2^2}^2(\boldsymbol{x})}\right) \\ &= \frac{1}{2}\log\left(1 + (\hat{s}_{k,\tau_1^2}^2(\boldsymbol{x}) - \hat{s}_{G\backslash k,\tau_2^2}^2(\boldsymbol{x}))/\hat{s}_{G_1\backslash k,\tau_2^2}^2(\boldsymbol{x})\right) \\ &\le \frac{1}{2}\log\left(1 + \varepsilon + N^{5/2}|\tau_1^2 - \tau_2^2|/\tau_2^2\right) \le \varepsilon + N^{5/2}|\tau_2^2 - \tau_1^2|/\tau_2^2.\end{aligned}$$

We used that $\hat{s}_{G_1\backslash k}^2(\boldsymbol{x}) \ge \sigma^2\tau_2^2/N$ (see Theorem 3.1). Suppose that estimates are available for the correlation parameters $\boldsymbol{\xi}$ by replacing $K(\boldsymbol{x},\boldsymbol{x})$ with $K(\boldsymbol{x},\boldsymbol{x}';\hat{\boldsymbol{\xi}})$ throughout the calculations above. Then, an extra term is added to $\hat{s}^2(\boldsymbol{x})$ to account for the parameter uncertainty [40]: $\hat{s}^2(\boldsymbol{x};\hat{\boldsymbol{\xi}}) = \sigma^2(1 - \boldsymbol{k}^T(\boldsymbol{x};\hat{\boldsymbol{\xi}}_i)\boldsymbol{K}_{\hat{\boldsymbol{\xi}}_i}^{-1}\boldsymbol{k}(\boldsymbol{x};\hat{\boldsymbol{\xi}}_i)) + E((\hat{y}(\boldsymbol{x};\boldsymbol{\xi}) - \hat{y}(\boldsymbol{x};\hat{\boldsymbol{\xi}}_i))^2)$. The estimates are updated at each greedy step, denoted by $\hat{\boldsymbol{\xi}}_i$, for greedy step i. Using (2.4), with zero mean, $\hat{y}(\boldsymbol{x};\boldsymbol{\xi}) - \hat{y}(\boldsymbol{x};\hat{\boldsymbol{\xi}}_i) = \boldsymbol{k}_{\boldsymbol{\xi}}^T(\boldsymbol{x})\boldsymbol{K}_{\boldsymbol{\xi}}^{-1}\boldsymbol{y}_k - \boldsymbol{k}_{\hat{\boldsymbol{\xi}}_i}^T(\boldsymbol{x})\boldsymbol{K}_{\hat{\boldsymbol{\xi}}_i}^{-1}\boldsymbol{y}_k$. Let us assume that $\|\boldsymbol{y}_k\|_2 \le \sqrt{N}$ (normalized). We know that there exists a constant $\alpha \ge 0$ such that, for all $\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^k$ and for all $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}, |K(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\xi}) - K(\boldsymbol{x},\boldsymbol{x}';\hat{\boldsymbol{\xi}}_i)| \le \alpha$. Then, $E(\hat{y}(\boldsymbol{x};\hat{\boldsymbol{\xi}}_i) - (\hat{y}(\boldsymbol{x};\boldsymbol{\xi}))^2) = E((\boldsymbol{k}_{\hat{\boldsymbol{\xi}}_i}^T(\boldsymbol{x})\boldsymbol{K}_{\hat{\boldsymbol{\xi}}_i}^{-1}\boldsymbol{y}_k - \boldsymbol{k}_{\boldsymbol{\xi}}^T(\boldsymbol{x})\boldsymbol{K}_{\boldsymbol{\xi}}^{-1}\boldsymbol{y}_k)^2) \le E((\|\boldsymbol{k}_{\boldsymbol{\xi}}^T(\boldsymbol{x}) - \boldsymbol{k}_{\hat{\boldsymbol{\xi}}_i}^T(\boldsymbol{x})\|_2\|\boldsymbol{K}_{\hat{\boldsymbol{\xi}}_i}^{-1}\|_2\|\boldsymbol{y}_k\|_2 + \|\boldsymbol{k}_{\hat{\boldsymbol{\xi}}_i}^T(\boldsymbol{x})\|_2\|\boldsymbol{K}_{\boldsymbol{\xi}}^{-1} - \boldsymbol{K}_{\hat{\boldsymbol{\xi}}_i}^{-1}\|_2)\|\boldsymbol{y}_k\|_2)^2) \le \alpha^2 N^2(1+N^{3/2})^2$. As a result, using similar calculations, $\mathcal{H}(\boldsymbol{x}|k) - \mathcal{H}(\boldsymbol{x}|k,\hat{\boldsymbol{\xi}}) \le \frac{1}{2}\log((\hat{s}_k^2(\boldsymbol{x}) + \alpha^2 N^2(1+N^{3/2})^2)/\hat{s}_D^2(\boldsymbol{x})) \le \frac{1}{2}\log(1 + (\alpha\sigma^{-1}\tau^{-1})^2 N^3(1+N^{3/2})^2)$. Hence,

$$\begin{aligned}&\mathcal{H}(\boldsymbol{x}|k,\hat{\xi},\tau_1^2) - \mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\hat{\xi},\tau_2^2) = (\mathcal{H}(\boldsymbol{x}|k,\tau_1^2) - \mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\tau_1^2)) \\ &+ (\mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\tau_1^2) - \mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\hat{\xi},\tau_1^2)) + (\mathcal{H}(\boldsymbol{x}|k,\hat{\xi},\tau_1^2) - \mathcal{H}(\boldsymbol{x}|k,\tau_1^2)) \\ &+ (\mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\hat{\xi},\tau_1^2) - \mathcal{H}(\boldsymbol{x}|(\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k),\hat{\xi},\tau_2^2)) \\ &\le \varepsilon + 2(\alpha\sigma^{-1}\tau^{-1})^2 N^3(1+N^{3/2})^2 + N^{5/2}|\tau_2^2 - \tau_1^2|/\tau_2^2.\end{aligned}$$

The two GPs on $\boldsymbol{X}_k$ and $\boldsymbol{X}_{G_1}\backslash\boldsymbol{X}_k$, respectively, use the same estimates $\hat{\boldsymbol{\xi}}$. Finally, by following the proof of Theorem 7 in [19], we can easily get the result of this theorem. ■

Proof of Theorem 3.3. Suppose that the design space $\mathcal{X}$ is a compact subset of $\mathbb{R}^p$ and discretized into a regular grid $\boldsymbol{X}_G \subset \mathcal{X}$ with spacing $\delta > 0$. Assume the correlation function $K(\cdot,\cdot)$ is Lipschitz continuous; then there exists a constant $K_L > 0$ such that $|\hat{s}^2(\boldsymbol{x}_1) - \hat{s}^2(\boldsymbol{x}_2)| \leq K_L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$ for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{X}_G$, where $\|\cdot\|_2$ is the Euclidean norm. Suppose we have a Gaussian emulator with constant mean, and a nonnegative nugget parameter τ^2. Then, for any $\varepsilon > 0$, assuming $\boldsymbol{X}_G$ has grid spacing $\delta \leq 2\varepsilon/(\sqrt{p}K_L)$, $\hat{s}^2(\boldsymbol{x}^*)$ is ε-close to $\hat{s}^2(\boldsymbol{x}_n)$ for any untried point $\boldsymbol{x}^* \in \mathcal{X}$, where $\boldsymbol{x}_n$ is the member of $\boldsymbol{X}_G$ closest to $\boldsymbol{x}^*$. According to Theorem 3.1, for any point $\boldsymbol{x}_i \in \boldsymbol{X}_G$ the predictive variance can be written as

$$\hat{s}^2_{\tau^2}(\boldsymbol{x}_i) = \sigma^2\left(\tau^2 - \tau^4\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{e}_i + \tau^4\frac{(\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1})^2}{\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1}}\right),$$

where $\mathcal{I}$ is the identity matrix and $\mathbf{e}_i$ is the ith unit vector. Hence, for any $\varepsilon > 0$ there exists a grid spacing $\delta > 0$ so that $-\sigma^2\tau^4 b_1(\tau^2) - \varepsilon < \hat{s}^2_{\tau^2}(\boldsymbol{x}^*) - \sigma^2\tau^2 < \sigma^2\tau^4 b_2(\tau^2) + \varepsilon$, where $b_1(\tau^2) = \max\left\{\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{e}_i : \boldsymbol{x}_i \in \boldsymbol{X}_G\right\}$ and

$$b_2(\tau^2) = \max\left\{\frac{(\mathbf{e}_i^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1})^2}{\mathbf{1}^T(\boldsymbol{K}+\tau^2\mathcal{I})^{-1}\mathbf{1}} : \boldsymbol{x}_i \in \boldsymbol{X}_G\right\}. \quad ■$$

Acknowledgments. The authors would like to acknowledge that the work presented here made use of the Emerald supercomputer, which was made available to them by the Center for Innovation in High Performance Computing. The Center is formed by the universities of Oxford, Southampton, and Bristol and University College London in partnership with the STFC Rutherford Appleton Laboratory. The authors would like to thank the anonymous reviewers for their constructive comments that greatly improved the manuscript.

REFERENCES

[1] S. Ba and V.R. Joseph, *Multi-layer designs for computer experiments*, J. Amer. Statist. Assoc., 106 (2011), pp. 1139–1149.

[2] E.N. Ben-Ari and D.M. Steinberg, *Modeling data from computer experiments: An empirical comparison of kriging with MARS and projection pursuit regression*, Qual. Eng., 19 (2007), pp. 327–338.

[3] I. Bilionis and N. Zabaras, *Multi-output local Gaussian process regression: Applications to uncertainty quantification*, J. Comput. Phys., 231 (2012), pp. 5718–5746.

[4] W.F. Caselton and J.V. Zidek, *Optimal monitoring network designs*, Statist. Probab. Lett., 2 (1984), pp. 223–227.

[5] J.A. Christen and B. Sansó, *Advances in the sequential design of computer experiments based on active learning*, Comm. Statist. Theory Methods, 40 (2011), pp. 4467–4483.

[6] E. Contal, V. Perchet, and N. Vayatis, *Gaussian process optimization with mutual information*, in Proceedings of the 31st International Conference on Machine Learning (ICML), Beijing, 2014, pp. 253–261.

[7] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, 2nd ed., Wiley Ser. Telecom., Wiley-Interscience, [John Wiley & Sons], Hoboken, NJ, 2006.

[8] C. Currin, T.J. Mitchell, M.D. Morris, and D. Ylvisaker, *A Bayesian Approach to the Design and Analysis of Computer Experiments*, Tech. report ORNL-6498, Oak Ridge National Laboratory, Oak Ridge, TN, 1988. Available online at http://web.ornl.gov/~webworks/cpr/rpt/6863.pdf.

[9] G.M. Dancik and K.S. Dorman, *mlegp: Statistical analysis for computer models of biological systems using R*, Bioinformatics, 24 (2008), pp. 1966–1967.
[10] K. Deb and R.B. Agrawal, *Simulated binary crossover for continuous search space*, Complex Syst., 9 (1995), pp. 115–148.
[11] D. Dutykh, R. Poncet, and F. Dias, *The VOLNA code for the numerical modeling of tsunami waves: Generation, propagation and inundation*, Eur. J. Mech. B Fluids, 30 (2011), pp. 598–615.
[12] A. Genz, *An adaptive numerical integration algorithm for simplices*, in Computing in the 90s, Lecture Notes in Comput. Sci. 507, Springer-Verlag, New York, 1991, pp. 279–285.
[13] R.B. Gramacy and H.K.H. Lee, *Adaptive design and analysis of supercomputer experiments*, Technometrics, 51 (2009), pp. 130–145.
[14] R.B. Gramacy and H.K.H. Lee, *Cases for the nugget in modeling computer experiments*, Stat. Comput., 22 (2012), pp. 713–722.
[15] M.S. Handcock and M.L. Stein, *A Bayesian analysis of kriging*, Technometrics, 35 (1993), pp. 403–410.
[16] X. Huan and Y.M. Marzouk, *Simulation-based optimal Bayesian experimental design for nonlinear systems*, J. Comput. Phys., 232 (2013), pp. 288–317.
[17] D.R. Jones, *A taxonomy of global optimization methods based on response surfaces*, J. Global Optim., 21 (2001), pp. 345–383.
[18] C.W. Ko, J. Lee, and M. Queyranne, *An exact algorithm for maximum entropy sampling*, Oper. Res., 43 (1995), pp. 684–691.
[19] A. Krause, A. Singh, and C. Guestrin, *Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies*, J. Mach. Learn. Res., 9 (2008), pp. 235–284.
[20] A.M. Kupresanin and G. Johannesson, *Comparison of Sequential Designs of Computer Experiments in High Dimensions*, Tech. report LLNL-TR-491692, Lawrence Livermore National Laboratory, Livermore, CA, 2011. Available online at https://e-reports-ext.llnl.gov/pdf/502919.pdf.
[21] C.Q. Lam and W.I. Notz, *Sequential adaptive designs in computer experiments for response surface model fit*, Stat. Appl., 6 (2008), pp. 207–233.
[22] D.V. Lindley, *On a measure of the information provided by an experiment*, Ann. Math. Statist., 27 (1956), pp. 986–1005.
[23] D. Maljovec, B. Wang, A. Kupresanin, G. Johannesson, V. Pascucci, and P.-T. Bremer, *Adaptive sampling with topological scores*, Int. J. Uncertain. Quantif., 3 (2013), pp. 119–141.
[24] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher, *An analysis of approximations for maximizing submodular set functions*, Math. Program., 14 (1978), pp. 265–294.
[25] C.Y. Peng and J. Wu, *On the choice of nugget in kriging modeling for deterministic computer experiments*, J. Comput. Graph. Statist., 23 (2014), pp. 151–168.
[26] P. Ranjan, R. Haynes, and R. Karsten, *A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data*, Technometrics, 53 (2011), pp. 366–378.
[27] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.
[28] G.K. Robinson, *That BLUP is a good thing: The estimation of random effects*, Statist. Sci., 6 (1991), pp. 15–32.
[29] O. Roustant, D. Ginsbourger, and Y. Deville, *DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, J. Stat. Softw., 51 (2012), pp. 1–55.
[30] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–423.
[31] T.J. Santner, B.J. Williams, and W.I. Notz, *The Design and Analysis of Computer Experiments*, Springer-Verlag, New York, 2003.
[32] A. Sarri, S. Guillas, and F. Dias, *Statistical emulation of a tsunami model for sensitivity analysis and uncertainty quantification*, Nat. Hazards Earth Syst. Sci., 12 (2012), pp. 2003–2018.
[33] M. Schonlau, *Computer Experiments and Global Optimization*, Ph.D. thesis, University of Waterloo, Waterloo, Ontario, 1998.

[34] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, *Gaussian process regression: Active data selection and test point rejection*, in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Vol. 3, Como, Italy, + 2000, pp. 241–246.

[35] M.C. Shewry and H.P. Wynn, *Maximum entropy sampling*, J. Appl. Statist., 14 (1987), pp. 165–170.

[36] T.W. Simpson, D.K.J. Lin, and W. Chen, *Sampling strategies for computer experiments: Design and analysis*, Int. J. Reliab. Appl., 2 (2001), pp. 209–240.

[37] I. Sraj, K.T. Mandli, O.M. Knio, C.N. Dawson, and I. Hoteit, *Uncertainty quantification and inference of Manning's friction coefficients using DART buoy data during the Tōhoku tsunami*, Ocean Model., 83 (2014), pp. 82–97.

[38] T.S. Stefanakis, E. Contal, N. Vayatis, F. Dias, and C.E. Synolakis, *Can small islands protect nearby coasts from tsunamis? An active experimental design approach*, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 470 (2014), 20140575.

[39] M.L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*, Springer-Verlag, New York, 1999.

[40] D.L. Zimmerman and N. Cressie, *Mean squared prediction error in the spatial linear model with estimated covariance parameters*, Ann. Inst. Statist. Math., 44 (1992), pp. 27–43.