

Composite grid designs for adaptive computer experiments with fast inference

BY M. PLUMLEE, C. B. ERICKSON, B. E. ANKENMAN

*Department of Industrial Engineering and Management Sciences, Northwestern University,
2145 Sheridan Road, Evanston, Illinois 60208, U.S.A.*

mplumlee@northwestern.edu collinerickson@u.northwestern.edu ankenman@northwestern.edu

AND E. LAWRENCE

*Statistical Sciences, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos,
New Mexico 87544, U.S.A.*

earl@lanl.gov

SUMMARY

Experiments are often used to produce emulators of deterministic computer code. This article introduces composite grid experimental designs and a sequential method for building the designs for accurate emulation. Computational methods are developed that enable fast and exact Gaussian process inference even with large sample sizes. We demonstrate that the proposed approach can produce emulators that are orders of magnitude more accurate than current approximations at a comparable computational cost.

Some key words: Emulation; Gaussian process; Kriging; Sequential experiment; Sparse grid.

1. INTRODUCTION

A computer experiment is a set of evaluations of a piece of deterministic computer code with a d -dimensional input. Such experiments are often used to build an emulator, which is a statistical estimator of the code. The classical recommendation for the sample size is $10d$ (Loeppky et al., 2009). However, $100d$ to $10\,000d$ samples may be needed to produce an emulator for a highly complex model with demanding goals for prediction accuracy (Harari et al., 2018). Gaussian process inference is the standard approach to constructing statistical emulators because of its flexibility (Santner et al., 2019), but a major drawback of Gaussian process inference with large sample sizes is that the standard statistical computational costs scale cubically with the sample size; this limits fast and exact inference to cases where the sample size is less than a thousand. There are ways of overcoming this computational barrier, such as the method of Gramacy & Apley (2015), but these approaches generally provide only approximate inference, which can lead to suboptimal emulation performance. The question of how to conduct exact Gaussian process inference with large sample sizes remains a challenge for computer experimenters.

Plumlee (2014) paired Gaussian processes with a special type of experimental design based on classical sparse grid interpolation rules (Wasilkowski & Wozniakowski, 1995; Nobile et al., 2008b) for fast inference. The algorithm shortened the computing time for some examples by several orders of magnitude. However, such sparse grid designs treat all input dimensions as homogeneous. The classical $10d$ sample size recommendation is based on the idea that the computer code's response is quite sensitive to some input perturbations and less sensitive to others (Linkletter et al., 2006; Savitsky et al., 2011). From this perspective, the sparse grid designs of Plumlee (2014) are inefficient since they do not account for the differing sensitivity of input dimensions. Other popular computer experiment methods, such as

those proposed by Joseph et al. (2015), Sun et al. (2019) and He (2020), are devised to be robust to input dimension heterogeneity; however, these types of experiments will not provide fast inference.

This article describes a computer experiment design strategy that accounts for heterogeneous sensitivity among the input dimensions while maintaining fast inference. The designs are unions of grids termed composite grid designs. We also present an algorithm that adaptively builds the design so as to achieve better emulation performance. In a simulation study, this emulation approach outperforms modern competitors while taking a similar amount of computation to build the emulator and significantly less computation to predict using the emulator.

2. COMPOSITE GRID DESIGNS

We label our input $x = (x_1, \dots, x_d)$ and, without loss of generality, take our input space to be $[0, 1]^d$. If we let \mathcal{X} represent a discrete set of points in $[0, 1]$, a grid is $\mathcal{X} \times \mathcal{X} \times \dots \times \mathcal{X}$, where \times is the Cartesian product. One problem with using a single grid for a computer experiment is that grids have poor projection properties. When \mathcal{X} has cardinality n , the existence of k inert input dimensions would mean that we have used n^k times more samples than needed. This would be an extremely wasteful design even if k is small. For this reason, grids are rarely directly useful for computer experiments.

Define an ordered sequence of discrete sets of points in $[0, 1]$, labelled $\mathcal{X}(1) \subset \mathcal{X}(2) \subset \mathcal{X}(3) \subset \dots$. While not needed for our results, it helps to imagine that $\mathcal{X}(1) = 0.5$, a single point in the middle of our space. In our simulations, $\mathcal{X}(2)$ is set to $(0.125, 0.5, 0.875)$. The results in this article hold regardless of this choice and require only that these sets be nested. Simulations indicate that the somewhat ad hoc choice described in the [Supplementary Material](#) yields good performance.

We propose using a composition of grids to create a better experimental design than a single grid. An index that describes a potential grid facilitates the explanation of these composite grid designs.

DEFINITION 1 (Index). *An index $i = (i_1, \dots, i_d)$ is a vector of positive integers. The grid corresponding to i is $\mathcal{X}(i_1) \times \mathcal{X}(i_2) \times \dots \times \mathcal{X}(i_d)$.*

Our composite grid design is formed as the union of all grids that correspond to a set of indexes. The structure of the index set is critical. If an index is composed of small elements, like $i = (1, 2, 1)$ when $d = 3$, the corresponding grid is reasonably small. If an index has large elements, like $i = (3, 8, 4)$ when $d = 3$, the corresponding grid is typically large. Thus, for both mathematical and practical reasons, one should ensure that smaller indexes appear before larger indexes in an index set.

DEFINITION 2 (Regular index set). *A regular index set \mathcal{I} is a set of indexes such that $\mathcal{A}(i) \subset \mathcal{I}$ for all $i \in \mathcal{I}$, where $\mathcal{A}(i)$ is the set of all indexes $i' \neq i$ such that $i'_k \leq i_k$ for $k = 1, \dots, d$.*

A regular index set can therefore be thought of as obeying a strong hereditary principle. Composite grid designs follow from the structure imposed on the regular index set.

DEFINITION 3 (Composite grid design). *A composite grid design built from a regular index set \mathcal{I} is defined as $\mathcal{C}(\mathcal{I}) = \bigcup_{i \in \mathcal{I}} \mathcal{X}(i_1) \times \mathcal{X}(i_2) \times \dots \times \mathcal{X}(i_d)$.*

Composite grid designs generalize the sparse grid designs from existing quadrature rules. Figure 1 gives four two-dimensional illustrations. The top panels show the index sets that generate the designs in the bottom panels using the definition of composite grid designs. The construction of the designs depends on $\mathcal{X}(1), \mathcal{X}(2), \dots, \mathcal{X}(5)$, where $\mathcal{X}(1)$ and $\mathcal{X}(2)$ are as described previously, and $\mathcal{X}(3), \mathcal{X}(4)$ and $\mathcal{X}(5)$ are described in the [Supplementary Material](#). The leftmost column in Fig. 1 represents the traditional sparse grid design used in Plumlee (2014), which employs the index set of all indexes such that $\sum_{k=1}^d i_k \leq \eta$ where $\eta = 5$. This type of index set clearly obeys our rules for a regular index set. A popular extension of sparse grid quadrature rules proposed by Nobile et al. (2008a) employs a weighted sum, illustrated in the second column of Fig. 1, where $i_1 + 2i_2 \leq 7$. This type of design accounts for some heterogeneous

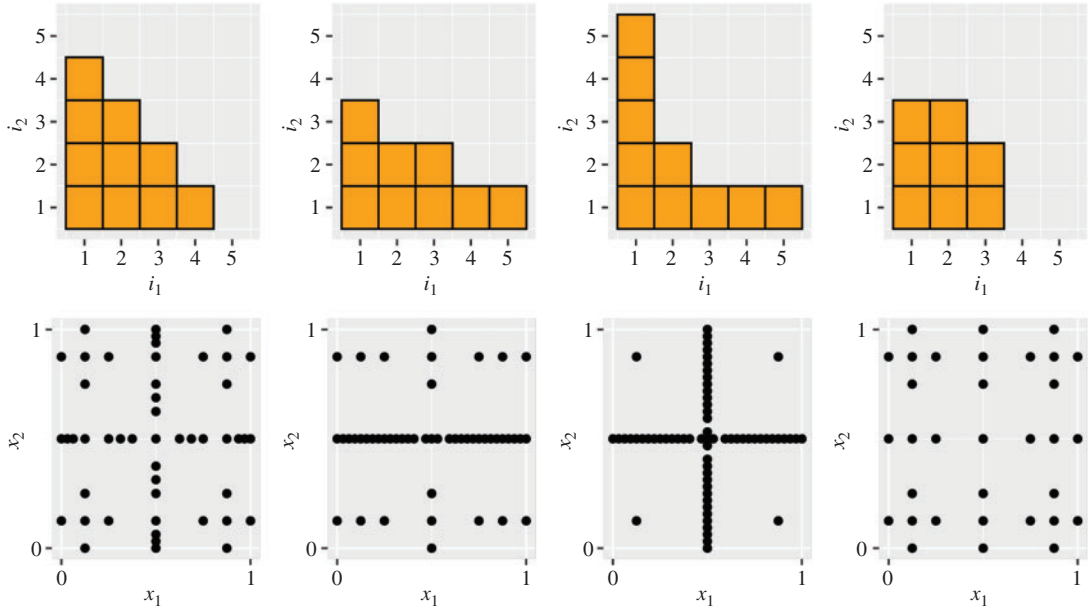


Fig. 1. Illustrations of composite grid designs in two dimensions: each of the top panels represents an index set, where a square appears if it is in the index set; the corresponding two-dimensional design is displayed below each index set. The specifics of the grids are given in the [Supplementary Material](#).

sensitivity among the input dimensions, as one dimension has significantly denser sampling than the other. The third and fourth columns of Fig. 1 represent alternative design approaches. The third column shows a design that frequently perturbs the input dimensions individually while infrequently perturbing them jointly. The design in the rightmost column of Fig. 1 takes the opposite approach.

There is no index set that is optimal across all Gaussian process models. An optimal index set for a specific Gaussian process model would perturb the sensitive input dimensions more than the insensitive input dimensions. Gaussian process models can have different input dimension sensitivities, which prevents optimality across all possible Gaussian process models. We will later discuss choosing the index set adaptively for good emulation. Before addressing this point, the next section gives the details of Gaussian process inference on composite grid designs.

3. COMPUTATIONALLY FAST INFERENCE

Given an input x , the scalar computer model response is labelled $y(x)$. In the statistical model, $y(\cdot)$ is a realization of a Gaussian process with mean zero and a separable covariance function such that the covariance between inputs x and x' is $\prod_{k=1}^d \phi_k(x_k, x'_k)$, where $\phi_1(\cdot, \cdot), \dots, \phi_d(\cdot, \cdot)$ are positive-definite functions on $[0, 1]^2$. If \mathcal{X} and \mathcal{X}' are sets of n and n' points in $[0, 1]$, $\phi_k(\mathcal{X}, \mathcal{X}')$ is the $n \times n'$ matrix of covariances.

Fast inference is possible when using grid designs because the covariance matrix is a Kronecker product of smaller matrices. This section describes how to extend this inference technique to composite grid designs. Our inference employs small covariance matrices corresponding to each dimension, $\phi_k\{\mathcal{X}(m), \mathcal{X}(m)\}$ for each k and integers $m \geq 1$. For each input dimension, the largest matrix we need to manipulate is the one corresponding to the maximum grid size in $\mathcal{C}(\mathcal{I})$ along that dimension. Even when the total design size of $\mathcal{C}(\mathcal{I})$ is large, these matrices remain small. Two manipulations of these smaller covariance matrices will be used:

$$\tilde{\Phi}_k(m) = \phi_k\{\mathcal{X}(m), \mathcal{X}(m)\}^{-1}, \quad l_k(m) = \log(\det[\phi_k\{\mathcal{X}(m), \mathcal{X}(m)\}]).$$

Both of these operations, matrix inversion and determinant calculation, have computational costs that practically scale with the size of the matrix cubed. Because these matrices are small, computations that use them will be significantly faster than those that use the full covariance matrix.

The following three theorems will use $\tilde{\Phi}_k(m)$ and $l_k(m)$ to predict, estimate the prediction errors, and compute the likelihood of the observations. The first two tasks have obvious value, and the last one is commonly used in parameter estimation of the Gaussian process model using experimental observations (Santner et al., 2019). These results extend those obtained in Plumlee (2014) by using the general index set \mathcal{I} instead of the sparse grid index set. The proofs, which are given in the [Supplementary Material](#), differ from those in Plumlee (2014) and are, for the most part, considerably simpler.

THEOREM 1. *Let Y represent the vector of observations of $y(\cdot)$ at $\mathcal{C}(\mathcal{I})$. For all $i \in \mathcal{I}$, define $Y(i)$ as the vector of the observations at $\mathcal{X}(i_1) \times \mathcal{X}(i_2) \times \cdots \times \mathcal{X}(i_d)$, which will be a subset of all observations in Y . The conditional mean of $y(x)$ given data Y at $\mathcal{C}(\mathcal{I})$ is*

$$\hat{y}(x) = \sum_{i \in \mathcal{I}} a(i) \left[\bigotimes_{k=1}^d \phi_k\{x_k, \mathcal{X}(i_k)\} \tilde{\Phi}_k(i_k) \right] Y(i),$$

where \otimes is the Kronecker product, $a(i) = \sum_{j \in \mathcal{I}} \prod_{k=1}^d b(i_k, j_k)$, and

$$b(m, m') = \begin{cases} -1, & m + 1 = m', \\ 1, & m = m', \\ 0, & \text{otherwise.} \end{cases}$$

THEOREM 2. *For $k = 1, \dots, d$ and $t, t' \in [0, 1]$, define*

$$v_k(t, t', m) = \phi_k\{t, \mathcal{X}(m)\} \tilde{\Phi}_k(m) \phi_k\{\mathcal{X}(m), t'\}$$

for integers m greater than zero and let $v_k(t, t', 0) = 0$. Given $\hat{y}(\cdot)$ as described in Theorem 1, the covariance between $y(x) - \hat{y}(x)$ and $y(x') - \hat{y}(x')$ is

$$\prod_{k=1}^d \phi_k(x_k, x'_k) - \sum_{i \in \mathcal{I}} \prod_{k=1}^d \{v_k(x_k, x'_k, i_k) - v_k(x_k, x'_k, i_k - 1)\}.$$

THEOREM 3. *The log of the likelihood of Y is*

$$\begin{aligned} & -\frac{N}{2} \log(2\pi) - \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{k=1}^d \{l_k(i_k) - l_k(i_k - 1)\} \prod_{m \neq k} \{n(i_m) - n(i_m - 1)\} \\ & - \frac{1}{2} \sum_{i \in \mathcal{I}} a(i) \left[Y(i)^T \left\{ \bigotimes_{k=1}^d \tilde{\Phi}_k(i_k) \right\} Y(i) \right], \end{aligned}$$

where $l_k(0) = 0$ for all k , N is the cardinality of $\mathcal{C}(\mathcal{I})$, and $n(m)$ is the cardinality of $\mathcal{X}(m)$, with $n(0) = 0$.

4. SEQUENTIALLY BUILDING A COMPOSITE GRID

The preceding two sections have established the framework for composite grid designs and their fast inference, but the choice of index set has a substantial influence on the performance of the resulting emulator. Ideally, the indexes should have their largest terms in the input dimensions that are most sensitive. We propose a sequential approach whereby the design is built iteratively. The composite grid begins with

a single element $\mathcal{I} = \{(1, 1, \dots, 1)\}$, and we add indexes that ensure the index set will be regular. The potential set of indexes \mathcal{P} contains all i such that $i \notin \mathcal{I}$ and $\mathcal{A}(i) \subset \mathcal{I}$. After adding an index to \mathcal{I} , it is removed from \mathcal{P} , and \mathcal{P} is updated with new potential indexes to ensure the regularity. In each iteration, an index choice criterion is optimized to select which index is moved from \mathcal{P} to \mathcal{I} .

Our index choice criterion is based on the variance of the emulator integrated over $[0, 1]^d$. By Theorem 2, if we use design $\mathcal{C}(\mathcal{I})$, the integrated variance, IV, (Sacks et al., 1989) is

$$\text{IV}(\mathcal{I}) = \int_{[0,1]^d} \left[\prod_{k=1}^d \phi_k(x_k, x_k) - \sum_{i \in \mathcal{I}} \prod_{k=1}^d \{v_k(x_k, x_k, i_k) - v_k(x_k, x_k, i_k - 1)\} \right] dx.$$

The most straightforward strategy is to choose whichever i maximizes the reduction in the integrated variance. The reduction for a new index in $i \in \mathcal{P}$ is

$$\Delta(i) = \prod_{k=1}^d \left[\int_0^1 \{v_k(t, t, i_k) - v_k(t, t, i_k - 1)\} dt \right],$$

which can be quickly computed by approximating the one-dimensional integrals $\int_0^1 \{v_k(t, t, i_k) - v_k(t, t, i_k - 1)\} dt$. The reduction must be considered alongside the sample size needed to achieve that reduction. The efficiency of an index set, denoted by $\text{EFF}(i)$, is $\Delta(i)$ divided by the number of new design points, $\prod_{k=1}^d \{n(i_k) - n(i_k - 1)\}$ where $n(\cdot)$ is as defined in Theorem 3. The efficiency of an index set acts as our index choice criterion. Algorithm 1 summarizes the proposed approach.

Algorithm 1. Building a composite grid design for emulation of size less than or equal to N^* .

Set \mathcal{P}' as all i in \mathcal{P} with $\prod_{k=1}^d \{n(i_k) - n(i_k - 1)\}$ less than N^* minus the cardinality of $\mathcal{C}(\mathcal{I})$
 While \mathcal{P}' is not empty
 Add $i^* = \arg \max_{i \in \mathcal{P}'} \text{EFF}(i)$ to \mathcal{I}
 Update \mathcal{P} using the new \mathcal{I}

The [Supplementary Material](#) explains why this greedy algorithm achieves nearly optimal performance under a reasonable assumption on $\text{EFF}(\cdot)$. In practice, $\Delta(i)$ depends on the covariance functions, which are usually unknown before experimentation. Given some experimental data, the standard approach is to parameterize a covariance function and maximize the likelihood to estimate parameters (Santner et al., 2019). These estimates allow us to leverage the collected data for better index selection in future iterations. Thus, our recommendation is to use sequential batches of experimentation: perform a computer experiment using $\mathcal{C}(\mathcal{I})$, find the maximum likelihood estimates of all covariance parameters, and append to \mathcal{I} via more iterations of Algorithm 1. The [Supplementary Material](#) also includes further details on the software implementation. More sophisticated methods for incorporating uncertainty in the estimation of the covariance parameters are briefly discussed, but in our experience these methods yield insignificant differences in performance.

5. COMPARISON TO MODERN ALTERNATIVES

In this section we compare the emulation performance of several methods on a set of four deterministic functions, termed Borehole, Circuit, Piston and Wing, with respective input dimensions 8, 6, 7 and 10. All these test functions have been used in the computer experiment literature, and their details are provided in the [Supplementary Material](#). Each function exhibits heterogeneous input dimension sensitivity, which we did not intentionally look for when selecting test functions, but was not unexpected as most functions do have heterogeneous sensitivity. Additionally, the 2019 Northwestern University PhD thesis by C. B. Erickson describes a successful implementation of this method on a computer model of cosmology, but the model's computational expense precluded the use of additional runs for comparison with other methods.

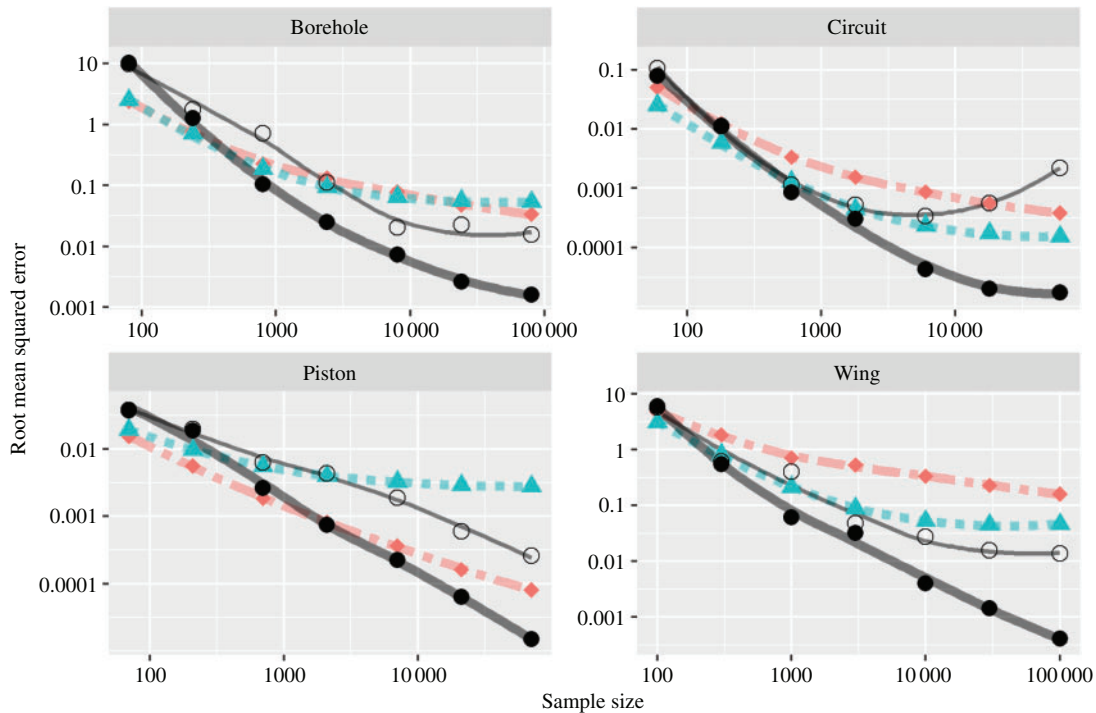


Fig. 2. Average root mean squared prediction error versus sample size for four test functions with four methods: 1aGP (diamonds and dash-dotted line), BASS (triangles and short-dashed line), the proposed approach (filled circles and thick solid line), and the proposed approach if a random index is selected among the ones with smallest sample size (open circles and thin solid line).

We compare our proposed method with existing emulation techniques that scale to large experiments. The R package 1aGP (R Development Core Team, 2020), described in Gramacy & Apley (2015), uses Gaussian process inference with only a carefully selected subset of the design points. The R package BASS, described in Francom et al. (2019), conducts Bayesian nonparametric regression with adaptive splines. Our method is implemented in the R package CGGP. For the competitors, the experimental design points are from a Latin hypercube that is maximin (Morris & Mitchell, 1995) when the sample size is less than 1000, and are from a random Latin hypercube for larger sample sizes because of the extremely high computational cost of finding a maximin design with large sample sizes. We assess the performance of these approaches by calculating the average prediction accuracy at 1000 test points using sample sizes of $10d$ to $10\,000d$. Ten replications of each simulation are used to account for sources of randomness in both the Latin hypercube designs and our designs, and we average over these replications. The differences between methods were much greater than the replication variability.

Figure 2 shows the root mean squared prediction error. The proposed approach performs the best by a wide margin for large sample sizes, several orders of magnitude better in some cases. The competing methods sometimes level off after 1000 to 10 000 observations. These results imply that the approximations employed in the other packages significantly affect the predictive performance. We also ran a comparison based on two predictive scoring rules from Gneiting & Raftery (2007) instead of the root mean squared prediction error, and found similar competitiveness in small samples and superiority of the proposed approach in large samples. These comparisons are reported in the Supplementary Material. Moreover, Fig. 2 also compares the results of our method without the adaptive design approach described in § 4. This comparison allows us to assess the relative contribution of the dimensional adaptivity, thus effectively comparing the proposed designs to those in Plumlee (2014). The results show that adaptivity is clearly advantageous for all the test functions. We also tried feeding our adaptive experiment data to the other

two approaches as input, but obtained very poor results. This implies that both the adaptive design and the exact inference contribute to the observed superior performance.

In the [Supplementary Material](#) we demonstrate that our proposed method is competitive in terms of the time required to build the emulator. The build time for a sample of size close to 10^5 was about 20 minutes, approximately four times faster than BASS and four times slower than LAGP. We also show that the resulting prediction obtained by our method for close to 10^5 samples was three times faster than BASS and 10 times faster than LAGP. Because LAGP finds optimal subdesigns during each prediction, it was expected to be much slower than our approach. We have no explanation for the slower BASS predictions.

ACKNOWLEDGEMENT

We thank the Isaac Newton Institute and the U.S. National Science Foundation's CDS & E programme for their support, as well as J. P. Gosling, W. J. Welch and H. P. Wynn for their comments and motivation.

SUPPLEMENTARY MATERIAL

[Supplementary Material](#) available at *Biometrika* online contains proofs of the theorems, implementation details for Algorithm 1, and further details of the comparisons.

REFERENCES

- FRANCOM, D., SANSÓ, B., BULAEVSKAYA, V., LUCAS, D. & SIMPSON, M. (2019). Inferring atmospheric release characteristics in a large computer experiment using Bayesian adaptive splines. *J. Am. Statist. Assoc.* **114**, 1450–65.
- GNEITING, T. & RAFTERY, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *J. Am. Statist. Assoc.* **102**, 359–78.
- GRAMACY, R. B. & APLEY, D. W. (2015). Local Gaussian process approximation for large computer experiments. *J. Comp. Graph. Statist.* **24**, 561–78.
- HARARI, O., BINGHAM, D., DEAN, A. & HIGDON, D. (2018). Computer experiments: Prediction accuracy, sample size and model complexity revisited. *Statist. Sinica* **28**, 899–910.
- HE, X. (2020). Lattice-based designs with quasi-optimal separation distance on all projections. *Biometrika* to appear, DOI: 10.1093/biomet/asaa057.
- JOSEPH, V. R., GUL, E. & BA, S. (2015). Maximum projection designs for computer experiments. *Biometrika* **102**, 371–80.
- LINKLETTER, C., BINGHAM, D., HENGARTNER, N., HIGDON, D. & YE, K. Q. (2006). Variable selection for Gaussian process models in computer experiments. *Technometrics* **48**, 478–90.
- LOEPPKY, J. L., SACKS, J. & WELCH, W. J. (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics* **51**, 366–76.
- MORRIS, M. D. & MITCHELL, T. J. (1995). Exploratory designs for computational experiments. *J. Statist. Plan. Infer.* **43**, 381–402.
- NOBILE, F., TEMPONE, R. & WEBSTER, C. G. (2008a). An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.* **46**, 2411–42.
- NOBILE, F., TEMPONE, R. & WEBSTER, C. G. (2008b). A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.* **46**, 2309–45.
- PLUMLEE, M. (2014). Fast prediction of deterministic functions using sparse grid experimental designs. *J. Am. Statist. Assoc.* **109**, 1581–91.
- R DEVELOPMENT CORE TEAM (2020). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0, <http://www.R-project.org>.
- SACKS, J., WELCH, W. J., MITCHELL, T. J. & WYNN, H. P. (1989). Design and analysis of computer experiments. *Statist. Sci.* **4**, 409–23.
- SANTNER, T. J., NOTZ, W. & WILLIAMS, B. J. (2019). *The Design and Analysis of Computer Experiments*. New York: Springer, 2nd ed.
- SAVITSKY, T., VANNUCCI, M. & SHA, N. (2011). Variable selection for nonparametric Gaussian process priors: Models and computational strategies. *Statist. Sci.* **26**, 130–49.
- SUN, F., WANG, Y. & XU, H. (2019). Uniform projection designs. *Ann. Statist.* **47**, 641–61.
- WASILKOWSKI, G. W. & WOZNAKOWSKI, H. (1995). Explicit cost bounds of algorithms for multivariate tensor product problems. *J. Complexity* **11**, 1–56.

[Received on 22 August 2019. Editorial decision on 2 September 2020]