SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 7, No. 3, pp. 838–876

Known Boundary Emulation of Complex Computer Models[*]

Ian Vernon[†], Samuel E. Jackson[†], and Jonathan A. Cumming[†]

Abstract. Computer models are now widely used across a range of scientific disciplines to describe various complex physical systems; however, to perform full uncertainty quantification we often need to employ emulators. An emulator is a fast statistical construct that mimics the complex computer model and greatly aids the vastly more computationally intensive uncertainty quantification calculations that a serious scientific analysis often requires. In some cases, the complex model can be solved far more efficiently for certain parameter settings, leading to boundaries or hyperplanes in the input parameter space where the model is essentially known. We show that for a large class of Gaussian process style emulators, multiple boundaries can be formally incorporated into the emulation process, by Bayesian updating of the emulators with respect to the boundaries, for trivial computational cost. The resulting updated emulator equations are given analytically. This leads to emulators that possess increased accuracy across large portions of the input parameter space. We also describe how a user can incorporate such boundaries within standard black box Gaussian process emulation packages that are currently available, without altering the core code. Appropriate designs of model runs in the presence of known boundaries are then analyzed, with two kinds of general purpose designs proposed. We then apply the improved emulation and design methodology to an important systems biology model of hormonal crosstalk in *Arabidopsis thaliana*.

Key words. Bayes linear emulation, boundary conditions, design of experiments, systems biology, Gaussian process emulation

AMS subject classifications. 62J, 62K, 62P

DOI. 10.1137/18M1164457

1. Introduction.

The use of mathematical models to describe complex physical systems is now commonplace in a wide variety of scientific disciplines. We refer to such models as *simulators*. Often they possess high numbers of input and/or output dimensions and are sufficiently complex that they may require substantial time for the completion of a single evaluation. The simulator may have been developed to aid understanding of the real-world system in question, or to be compared to observed data, necessitating a high-dimensional parameter search or model calibration, or to make predictions of future system behavior, possibly with the goal of aiding a future decision process. The responsible use of simulators in all of the above contexts usually requires a full (Bayesian) uncertainty analysis [6], which will aim to incorporate all the major relevant sources of uncertainty, for example, parametric uncertainty on the inputs to the model, observation uncertainties on the data, structural model discrepancy that represents the uncertain differences between the simulator and the

[*]Received by the editors January 9, 2018; accepted for publication (in revised form) February 25, 2019; published electronically July 11, 2019.
https://doi.org/10.1137/18M1164457

[†]Department of Mathematical Sciences, Durham University, Stockton Road, DH1 3LE Durham, UK (i.r.vernon@durham.ac.uk, samuel.jackson@durham.ac.uk, j.a.cumming@durham.ac.uk).

real system, both for past and future and the relation between them, and also all the many uncertainties related to the decision process [41].

However, such an uncertainty analysis, which represents a critically important part of any serious scientific study, usually requires a vast number of simulator evaluations. For complex simulators with even a modest runtime, this means that such an analysis is utterly infeasible. A solution to this problem is found through the use of emulators: an emulator is a statistical construct that seeks to mimic the behavior of the simulator over its input space but which is several orders of magnitude faster to evaluate. Early uses of Gaussian process emulators for computer models were given by [36, 13] with a more detailed account given in [37]. A vital feature of an emulator is that it gives both an expectation of the simulator's outputs at an unexplored input location as well as an uncertainty statement about the emulator's accuracy at this point, an attribute that elevates emulation above interpolation or other simple proxy modeling approaches. Therefore, emulators fit naturally within a Bayesian approach and help facilitate the full uncertainty analysis described above. For an early example of an uncertainty analysis using multilevel emulation combined with structural discrepancy modeling in a Bayesian history matching context see [9, 10], and for a fully Bayesian calibration of a complex nuclear radiation model, again incorporating structural model discrepancy, see [27].

Emulators have now been successfully employed across several scientific disciplines, including cosmology [41, 42, 8, 38, 20, 43, 32], climate modeling [46, 35, 25, 22] (the later employing emulation to increase the efficiency of an approximate Bayesian computation algorithm), epidemiology [2, 3, 1, 31, 30], systems biology [45, 24], oil reservoir modeling [11, 12], environmental science [16], traffic modeling [7], vulcanology [5], and even Bayesian analysis itself [44]. The development of improved emulation strategies therefore has the potential to benefit multiple scientific areas, allowing more accurate analyzes with lower computational cost. If additional prior insight into the physical structure of the model is available, it is of real importance that emulator structures capable of incorporating such insights have been developed to fully exploit this information.

Here we describe such an advance in emulation strategy that, when applicable, can lead to substantial improvements in emulator performance. In most cases, complex deterministic simulators have to be solved numerically for arbitrary input specifications, which leads to substantial runtimes. However, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved far more efficiently, either analytically in the ideal case or just significantly faster using a much more efficient and simpler solver. This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings, possibly due, for example, to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off). Note that this leads to Dirichlet boundary conditions, i.e., known simulator behavior on various hyperplanes, that impose constraints on the emulator itself and that these are distinct from Dirichlet boundary conditions imposed on the physical simulator model, which we do not require here (which are, for example, analyzed approximately using KL expansions by [39]). The goal, then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process, which should lead to significantly improved emulators. We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, and show that this is possible for a large class of emulators, for multiple boundaries of various forms (specifically collections of parallel or perpendicular hyperplanes), and most importantly, for trivial extra computational cost. We note that [40] have examined this problem; however, they used an approach which requires multiple extra emulator parameters that have to be estimated, as they essentially included substantial extra modeling to ensure both the mean and the variance of the emulator were consistent with the known boundary a priori. In contrast, our approach includes no extra modeling and zero additional parameters, instead updating the Gaussian process style emulator with the boundary information in a natural way. We also detail how users can include known boundaries when using standard black box Gaussian process software (without altering the core code), although this method is less powerful than implementing the fully updated emulators that we develop here. We then analyze the design problem of how to choose an efficient set of runs of the full simulator, given that we are aware of the existence of one or more known boundaries. Finally we apply this approach to a model of hormonal crosstalk in *Arabidopsis*, an important model in systems biology, which possesses these features.

The article is organized as follows. In section 2 we describe a standard emulation approach for deterministic simulators. In section 3 we develop the full known boundary emulation (KBE) methodology, explicitly constructing emulators that have been updated by one or two perpendicular (or parallel) known boundaries. In section 4 we discuss the design problem of how to choose efficient sets of simulator runs. In section 5 we apply both the KBE and the design techniques to the systems biology *Arabidopsis* model, before concluding in section 6.

2. Emulation of complex computer models.

We consider a complex computer model represented as a function $f(x)$, where $x \in \mathcal{X}$ denotes a d-dimensional vector containing the computer model's input parameters, and $\mathcal{X} \subset \mathbb{R}^d$ is a prespecified input parameter space of interest. We imagine that a single evaluation of the computer model takes a substantial amount of time to complete, and hence we will only be able evaluate it at a small number of locations. Here we assume $f(x)$ is univariate, but the results we present should directly generalize to the corresponding multivariate case.

We represent our beliefs about the unknown $f(x)$ at unevaluated input x via an emulator. For now, we assume that the form of the emulator is that of a pure Gaussian process (or in a less fully specified version, a weakly second order stationary stochastic process):

$$f(x) = u(x). \tag{2.1}$$

We make the judgment, consistent with most of the computer model literature, that the $u(x)$ have a product correlation structure:

$$\text{Cov}\left[u(x), u(x')\right] = \sigma^2 r(x - x') = \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i') \tag{2.2}$$

with $r_i(0) = 1$, corresponding to deterministic $f(x)$. Product correlation structures are very common, with the most popular being the Gaussian correlation structure given by

$$r(x - x') = \exp\{-|x - x'|^2/\theta^2\} = \prod_{i=1}^{d} \exp\{-|x_i - x'_i|^2/\theta^2\}, \tag{2.3}$$

which can be generalized to have different θ_i in each direction, while maintaining the product structure. As usual, we will also assume stationarity, but the following derivations do not require this assumption.

If we perform a set of runs at locations $x_D = (x^{(1)}, \dots, x^{(n)})$ over the input space of interest $\mathcal{X}$, giving computer model outputs as the column vector $D = (f(x^{(1)}), \dots, f(x^{(n)}))^T$, then we can update our beliefs about the computer model $f(x)$ in light of D. This can be done either using the Bayes theorem (if $u(x)$ is assumed to be a Gaussian process) or using the Bayes linear update formulas (which, following De Finetti [14], treat expectation as primitive and require only a second order specification [15, 17]):

$$\mathrm{E}_D[f(x)] = \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), D]\,\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]), \tag{2.4}$$

$$\mathrm{Var}_D[f(x)] = \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), D]\,\mathrm{Var}[D]^{-1}\,\mathrm{Cov}[D, f(x)], \tag{2.5}$$

$$\mathrm{Cov}_D\left[f(x), f(x')\right] = \mathrm{Cov}\left[f(x), f(x')\right] - \mathrm{Cov}[f(x), D]\,\mathrm{Var}[D]^{-1}\,\mathrm{Cov}\left[D, f(x')\right], \tag{2.6}$$

where $\mathrm{E}_D[f(x)]$, $\mathrm{Var}_D[f(x)]$, and $\mathrm{Cov}_D[f(x), f(x')]$ are the expectation, variance, and covariance of $f(x)$ adjusted by D [15, 17]. Equation (2.5) is obviously a special case of (2.6), but we include it explicitly to aid clarity in subsequent derivations. The fully Bayesian calculation, using the Bayes theorem, would yield similar update formulas for the analogous posterior quantities. Although we will work within the Bayes linear formalism, the derived results will apply directly to the fully Bayesian case, were one willing to make the additional assumption of full normality that use of a Gaussian process entails. In this case all Bayes linear adjusted quantities can be directly mapped to the corresponding posterior versions, e.g., $\mathrm{E}_D[f(x)] \to \mathrm{E}[f(x)|D]$ and $\mathrm{Var}_D[f(x)] \to \mathrm{Var}[f(x)|D]$. See [15, 17] for discussion of the benefits of using a Bayes linear approach and [41, 42] for its benefits within a computer model setting.

The results presented in this article rely on the product correlation structure of the emulator. As such, expansion of these methods to more general emulator forms requires further calculation. For example, a more advanced emulator is given by [10, 41]

$$f(x) = \sum_j \beta_j g_j(x_A) + u(x_A) + v(x), \tag{2.7}$$

where the active inputs x_A are a subset of x that are strongly influential for $f(x)$, the first term on the right-hand side is a regression term containing known functions $g_j(x_A)$ and possibly unknown β_j, $u(x_A)$ is a Gaussian process over the active inputs only, and $v(x)$ is an uncorrelated nugget term, representing the inactive variables. See also [11] and [41, 42] for discussions of the benefits of using an emulator structure of this kind, and see [27, 21] for discussions of alternative structures. We will discuss the generalization of our results to (2.7) in section 6, but currently we note that if the regression parameters β_j are assumed known, perhaps due to sufficiently large run number, and if all variables are assumed active, then (2.7) reduces to the required form, and all our results will apply.

3. Known boundary emulation.

We now consider the situation where the computer model is analytically solvable on some lower dimensional boundary $\mathcal{K}$. Hence we can evaluate $\{f(x) : x \in \mathcal{K}\}$ a vast number of times m on $\mathcal{K}$ and use these to supplement our standard emulator evaluations over $\mathcal{X}$ to produce an emulator that respects the functional behavior of $f(x)$ along $\mathcal{K}$. We first examine the case of finite (but large) m, which can be analyzed using the standard Bayes linear update, but structure our calculations so that they can be simply generalized to continuous model evaluations on $\mathcal{K}$, which will require a generalized version of the Bayes linear update, as described in section 3.6.

Call the corresponding length m vector of model evaluations K. Unfortunately simply plugging these m runs into the Bayes linear update equations (2.4), (2.5), and (2.6), replacing D with K, would be infeasible due to the size of the $m \times m$ matrix inversion $\text{Var}\,[K]^{-1}$. For example, if the dimension $d_{\mathcal{K}}$ of $\mathcal{K}$ is not small, we may need m to be extremely large (billions or trillions, say) to capture all the information contained in $\mathcal{K}$. Hence a direct update of the emulator in light of the information in K is nontrivial. Here we show from first principles that this update can be performed analytically for a wide class of emulators. We do this by exploiting a sufficiency argument briefly described in the supplementary material of [27], and in [34], but which, to our knowledge, has not been fully explored or utilized in the context of KBE. The emulation problem is further compounded in the general case where we have both evaluations K on the boundary, and in the main bulk D defined as above. For this case we will develop a sequential update that first updates analytically by K to obtain $\text{E}_K[f(x)]$, $\text{Var}_K[f(x)]$, and $\text{Cov}_K[f(x), f(x')]$, as developed in section 3.1, and then subsequently updates by D, to obtain $\text{E}_{D\cup K}[f(x)]$, $\text{Var}_{D\cup K}[f(x)]$, and $\text{Cov}_{D\cup K}[f(x), f(x')]$, as described in section 3.3.

3.1. A single known boundary.

We wish to update the emulator, and hence our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$ in light of a single known boundary $\mathcal{K}$, where $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction (but we note that our results naturally extend to lower dimensional boundaries). To capture the simulator behavior along $\mathcal{K}$, we evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \ldots, y^{(m)}$. We also evaluate the perpendicular projection of the point of interest, x, onto the boundary $\mathcal{K}$, which we denote as x^K. We therefore extend the collection of boundary evaluations, K, to be the $m+1$ column vector

$$K = (f(x^K), f(y^{(1)}), \ldots, f(y^{(m)}))^T,$$

which is illustrated in Figure 1 (left panel). We start by examining the expression for $\text{E}_K[f(x)]$

$$\text{E}_K\,[f(x)] = \text{E}\,[f(x)] + \text{Cov}\,[f(x), K]\,\text{Var}\,[K]^{-1}\,(K - \text{E}\,[K]). \tag{3.1}$$

As noted above, this calculation is seemingly infeasible due to the $\text{Var}\,[K]^{-1}$ term. However, if we evaluate it at the point x^K itself, which lies on $\mathcal{K}$, and as we have evaluated $f(x^K)$, we must find, for the emulator of a smooth deterministic function with suitably chosen correlation structure, that $\text{E}_K[f(x^K)] = f(x^K)$ (and that $\text{Var}_K[f(x^K)] = 0$). This is indeed the case as can be seen by examining the structure of the $\text{Var}\,[K]^{-1}$ term. As $f(x^K)$ is included as the

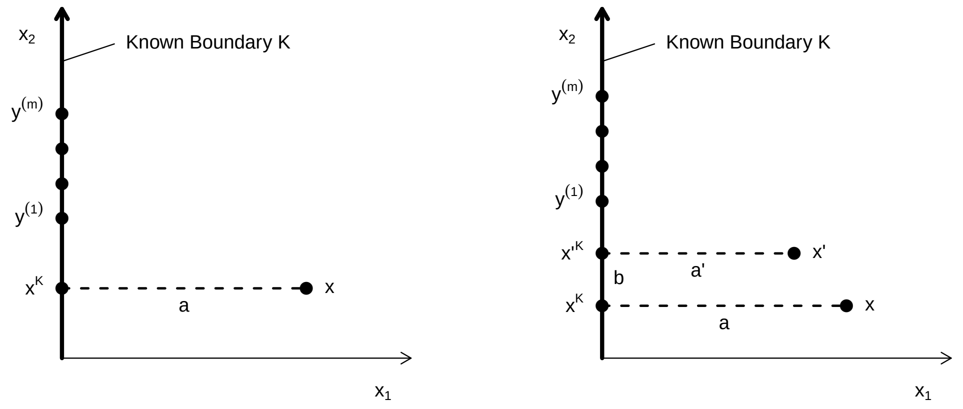


Figure 1. *The single known boundary case. Left panel: The points required for the* $\mathrm{E}_K[f(x)]$ *and* $\mathrm{Var}_K[f(x)]$ *calculation.* x *is the point we wish to emulate at,* x^K *its orthogonal projection onto the known boundary* $\mathcal{K}$ *at distance* a*. Right panel: The points required for the* $\mathrm{Cov}_K[f(x), f(x')]$ *calculation.* x *and* x' *are points we wish to update the covariance at, while* x^K *and* x'^K *are their orthogonal projection onto the known boundary* $\mathcal{K}$*, at distances* a *and* a'*, respectively. In both panels, the* $y^{(i)}$ *represent a large number of points for which we can evaluate* $f(y^{(i)})$ *analytically (or at least very quickly).*

first element of K, we note that

$$I_{(m+1)} = \mathrm{Var}\,[K]\,\mathrm{Var}\,[K]^{-1} \tag{3.2}$$

$$= \begin{pmatrix} \mathrm{Cov}[f(x^K), K] \\ \mathrm{Cov}[f(y^{(1)}), K] \\ \vdots \\ \mathrm{Cov}[f(y^{(m)}), K] \end{pmatrix} \mathrm{Var}\,[K]^{-1}, \tag{3.3}$$

where $I_{(m+1)}$ is the identity matrix of dimension $(m+1)$. Taking the first row of (3.3) gives

$$\mathrm{Cov}\left[f(x^K), K\right] \mathrm{Var}\,[K]^{-1} = (1, 0, \dots, 0). \tag{3.4}$$

Substituting (3.4) into the adjusted mean and variance naturally gives $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$ as it must. While unsurprising, this simple result is of particular value when considering the behavior at the point of interest, x. As we have defined x^K as the perpendicular projection of x onto $\mathcal{K}$, we can write $x = x^K + (a, 0, \dots, 0)$ for some constant a. Now we can exploit the symmetry of the product correlation structure (2.2) to obtain the covariance expressions

$$\mathrm{Cov}[f(x), f(x^K)] = \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i^K) = \sigma^2 r_1(x_1 - x_1^K) = \sigma^2 r_1(a)$$

$$= r_1(a)\,\mathrm{Cov}[f(x^K), f(x^K)] \tag{3.5}$$

since $x_i = x_i^K$ for $i = 2, \ldots, d$ and $r_i(0) = 1$. Furthermore,

$$\begin{aligned} \mathrm{Cov}[f(x), f(y^{(j)})] &= \sigma^2 \prod_{i=1}^{d} r_i(x_i - y_i^{(j)}) = \sigma^2 r_1(x_1 - x_1^K) \prod_{i=2}^{d} r_i(x_i - y_i^{(j)}) \\ &= \sigma^2 r_1(a) \prod_{i=2}^{d} r_i(x_i^K - y_i^{(j)}) \\ &= r_1(a)\, \mathrm{Cov}[f(x^K), f(y^{(j)})] \end{aligned} \tag{3.6}$$

since the first components of x^K and $y^{(j)}$ must be equal as they all lie on $\mathcal{K}$ (i.e., $x_1^K = y_1^{(j)}$). Combining (3.5) and (3.6), the covariance between point x and the set of boundary evaluations is given by

$$\begin{aligned} \mathrm{Cov}\left[f(x), K\right] &= \Big(\mathrm{Cov}[f(x), f(x^K)], \mathrm{Cov}[f(x), f(y^{(1)})], \ldots, \mathrm{Cov}[f(x), f(y^{(m)})]\Big) \\ &= r_1(a)\Big(\mathrm{Cov}[f(x^K), f(x^K)], \mathrm{Cov}[f(x^K), f(y^{(1)})], \ldots, \mathrm{Cov}[f(x^K), f(y^{(m)})]\Big) \\ &= r_1(a)\, \mathrm{Cov}\left[f(x^K), K\right]. \end{aligned} \tag{3.7}$$

Equations (3.4) and (3.7) are very useful results that greatly simplify the emulator calculations. We can use them to write the adjusted emulator expectation for $f(x)$ given in (3.1) as

$$\begin{aligned} \mathrm{E}_K\left[f(x)\right] &= \mathrm{E}\left[f(x)\right] + \mathrm{Cov}\left[f(x), K\right] \mathrm{Var}\left[K\right]^{-1}(K - \mathrm{E}\left[K\right]) \\ &= \mathrm{E}\left[f(x)\right] + r_1(a)\, \mathrm{Cov}\left[f(x^K), K\right] \mathrm{Var}\left[K\right]^{-1}(K - \mathrm{E}\left[K\right]) \\ &= \mathrm{E}\left[f(x)\right] + r_1(a)(1, 0, \ldots, 0)(K - \mathrm{E}\left[K\right]) \\ &= \mathrm{E}\left[f(x)\right] + r_1(a)(f(x^K) - \mathrm{E}\left[f(x^K)\right]). \end{aligned} \tag{3.8}$$

Thus we have eliminated the need to explicitly invert the large matrix $\mathrm{Var}\left[K\right]$ entirely by exploiting the symmetric product correlation structure and the identity (3.4). Similarly, we find the adjusted variance using (2.5), (3.4), and (3.7),

$$\begin{aligned} \mathrm{Var}_K\left[f(x)\right] &= \mathrm{Var}\left[f(x)\right] - r_1(a)(1, 0, \ldots, 0)\mathrm{Cov}\left[K, f(x)\right] \\ &= \mathrm{Var}\left[f(x)\right] - r_1(a)\mathrm{Cov}[f(x^K), f(x)] \\ &= \sigma^2(1 - r_1(a)^2). \end{aligned} \tag{3.9}$$

Equations (3.8) and (3.9) give the expectation and variance of the emulator at a point x, updated by a known boundary $\mathcal{K}$. As they require only evaluations of the analytic boundary function and the correlation function they can be implemented with trivial computational cost in comparison to a direct update by K. Note that they critically rely on the projected point $f(x^K)$ being in K.

Finally, we consider the Bayes linear update for the covariance between x and a second input point $x' \in \mathcal{X}$ given the boundary $\mathcal{K}$. We define the orthogonal projection of x' onto $\mathcal{K}$ as x'^K and denote its perpendicular distance from $\mathcal{K}$ as a', as shown in Figure 1 (right panel).

Equation (2.6) now gives

$$\begin{aligned}\mathrm{Cov}_K\left[f(x),f(x')\right] &= \mathrm{Cov}\left[f(x),f(x')\right] - \mathrm{Cov}\left[f(x),K\right]\mathrm{Var}\left[K\right]^{-1}\mathrm{Cov}\left[K,f(x')\right]\\ &= \mathrm{Cov}\left[f(x),f(x')\right] - r_1(a)(1,0,\dots,0)\mathrm{Cov}\left[K,f(x')\right]\\ &= \mathrm{Cov}\left[f(x),f(x')\right] - r_1(a)\mathrm{Cov}\left[f(x^K),f(x')\right]\\ &= \mathrm{Cov}\left[f(x),f(x')\right] - r_1(a)\mathrm{Cov}\left[f(x^K),f(x'^K)\right]r_1(a'), \end{aligned} \tag{3.10}$$

where in the final line we used the equivalent result to (3.6), rewritten for x'. Noting that we can also write $x' = x'^K + (a',0,\dots,0)$, and that $x_1^K = x_1'^K$, (3.10) becomes

$$\begin{aligned}\mathrm{Cov}_K\left[f(x),f(x')\right] &= \sigma^2\prod_{i=1}^d r_i(x_i-x_i') - r_1(a)r_1(a')\sigma^2\prod_{i=1}^d r_i(x_i^K - x_i'^K)\\ &= \sigma^2 r_1(a-a')\prod_{i=2}^d r_i(x_i-x_i') - \sigma^2 r_1(a)r_1(a')r_1(0)\prod_{i=2}^d r_i(x_i^K-x_i'^K)\\ &= \sigma^2 r_1(a-a')\prod_{i=2}^d r_i(x_i^K-x_i'^K) - \sigma^2 r_1(a)r_1(a')\prod_{i=2}^d r_i(x_i^K-x_i'^K)\\ &= \sigma^2\left(r_1(a-a')-r_1(a)r_1(a')\right)r_{-1}(x^K-x'^K)\\ &= \sigma^2 R_1(a,a')\,r_{-1}(x^K-x'^K), \end{aligned} \tag{3.11}$$

where we have defined the correlation function of the projection of x and x' onto $\mathcal{K}$ as

$$r_{-1}(x^K-x'^K) = \prod_{i=2}^d r_i(x_i^K-x_i'^K) = \mathrm{Cov}\left[f(x^K),f(x'^K)\right]$$

and defined the "updated correlation component" in the x_1 direction as

$$R_1(a,a') = r_1(a-a') - r_1(a)r_1(a'). \tag{3.12}$$

We see of course that (3.9) is a special case of (3.11), with $x = x'$.

These expressions for the expectation and (co)variance updated by the information at the simulator boundary provide several insights:

(a) *Sufficiency:* For the updating of our beliefs about the emulator at point x, we see that $f(x^K)$ is sufficient for K. Hence, only the evaluation $K = f(x^K)$ is required and the evaluations $y^{(i)}$ are redundant (note that under an assumption of an underlying Gaussian process, this result corresponds to a conditional independence statement discussed in the supplementary material to [27]). This has important ramifications for users of black box Gaussian process packages, as we discuss in section 3.3. It also implies that if we are interested in emulating at any point $x \in \mathcal{X}$, we only require the known boundary $\mathcal{K}$ to contain the projection of $\mathcal{X}$. So, for example, $\mathcal{K}$ could be only a bounded subset of a hyperplane, provided $\mathcal{X}$ is similarly bounded.

(b) *The correlation structure is now no longer stationary:* The contribution to the correlation function from dimensions 2 to d, denoted $r_{-1}(x^K - x'^K)$, is unchanged by the update (as we would expect from symmetry arguments); however, the contribution in the x_1 direction depends on the distance to the boundary $\mathcal{K}$, through $R_1(a, a')$, breaking stationarity.

(c) *The correlation structure is still in product form:* Critically, as the correlation structure has maintained its product from, this suggests that we can update by further known boundaries, either perpendicular to any of the remaining inputs x_i, with $i = 2, \ldots, d$, hence perpendicular to $\mathcal{K}$ or indeed by a second boundary parallel to $\mathcal{K}$. We perform these updates in sections 3.4 and 3.5.

(d) *Intuitive limiting behavior*: As we move x toward $\mathcal{K}$, the emulator tends toward the known boundary function, and as we move away from $\mathcal{K}$ the emulator reverts to its prior form, as expected:

$$\lim_{a\to 0} \mathrm{E}_K[f(x)] = f(x^K), \qquad \lim_{a\to 0} \mathrm{Var}_K[f(x)] = 0, \tag{3.13}$$

$$\lim_{a\to \infty} \mathrm{E}_K[f(x)] = \mathrm{E}[f(x)], \qquad \lim_{a\to \infty} \mathrm{Var}_K[f(x)] = \mathrm{Var}[f(x)], \tag{3.14}$$

as $\lim_{a\to\infty} r_1(a) = 0$. Similarly, the behavior of $\mathrm{Cov}_K[f(x), f(x')]$ is as expected, tending to its prior form far from the boundary (with $a - a'$ finite) and to zero as either a and a' tend to zero:

$$\lim_{a\to 0} \mathrm{Cov}_K\left[f(x), f(x')\right] = \lim_{a'\to 0} \mathrm{Cov}_K\left[f(x), f(x')\right] = 0, \tag{3.15}$$

$$\lim_{a,a'\to \infty} \mathrm{Cov}_K\left[f(x), f(x')\right] = \sigma^2 r(x - x') = \mathrm{Cov}\left[f(x), f(x')\right], \quad a - a' \text{ finite}. \tag{3.16}$$

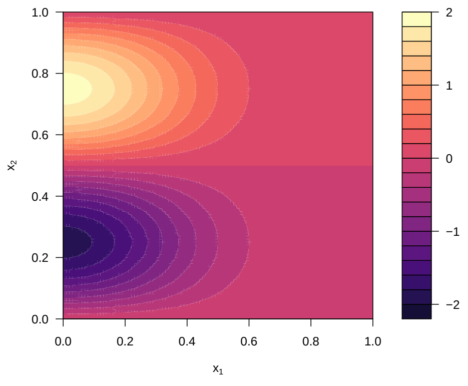
3.2. Application to a two-dimensional model. For illustration, we consider the problem of emulating the two-dimensional function

$$f(x) = -\sin(2\pi x_2) + 0.9 \sin(2\pi(1 - x_1)(1 - x_2)) \tag{3.17}$$

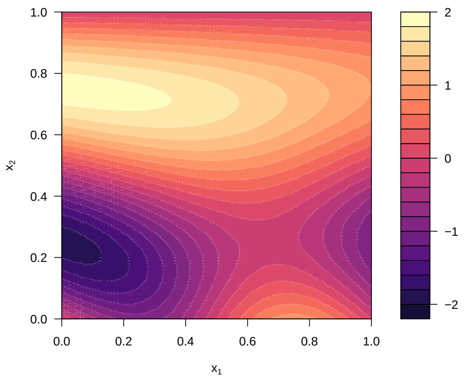
defined over the region $\mathcal{X}$ given by $0 < x_1 < 1$, $0 < x_2 < 1$, where we assume a known boundary $\mathcal{K}$ at $x_1 = 0$, and hence have that $f(x^K) = f(0, x_2) = -1.9 \sin(2\pi x_2)$. The true output of $f(x)$ over $\mathcal{X}$ is given in Figure 2(b) for reference. Using a prior expectation $\mathrm{E}[f(x)] = 0$ and a product Gaussian covariance structure with parameters $\theta = 0.4$ and $\sigma = 1$, we apply the expectation and variance updates (3.8) and (3.9) given the boundary $\mathcal{K}$ at $x_1 = 0$ and find that

$$\mathrm{E}_K[f(x)] = -1.9 \exp\{-x_1^2/\theta^2\} \sin(2\pi x_2),$$
$$\mathrm{Var}_K[f(x)] = 1 - \exp\{-2x_1^2/\theta^2\}.$$

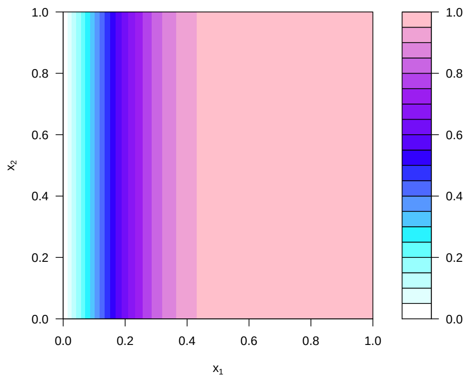
Figure 2(a) shows the adjusted expectation $\mathrm{E}_K[f(x)]$ over $\mathcal{X}$, clearly illustrating how the expectation surface has been changed in the vicinity of $\mathcal{K}$ to agree with the simulator behavior. Figure 2(c) shows the adjusted emulator standard deviation $\sqrt{\mathrm{Var}_K[f(x)]}$ and demonstrates the significant reduction in emulator uncertainty near $\mathcal{K}$. Finally, Figure 2(d) shows simple emulator diagnostics over $\mathcal{X}$ of the form of the standardized values $S_K(x) = (\mathrm{E}_K[f(x)] - f(x))/\sqrt{\mathrm{Var}_K[f(x)]}$. Thus any values of x for which $S_K(x)$ was far from 0 (a typical choice



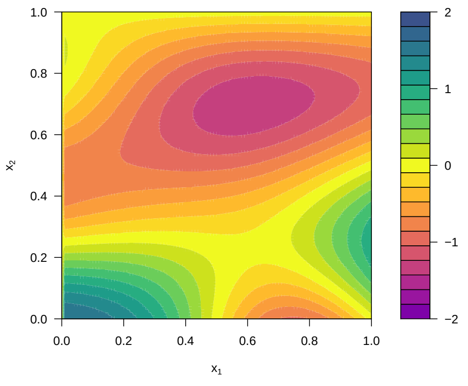
(a) The emulator expectation $\mathrm{E}_K[f(x)]$.



(b) The true 2-dimensional function $f(x)$.



(c) The emulator stan. dev. $\sqrt{\mathrm{Var}_K\,[f(x)]}$.



(d) Emulator diagnostics $S_K(x)$.

Figure 2. *Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$.*

being $|S_K(x)| > 3$) would indicate a conflict between emulator and simulator (see [4] for details). For our boundary-adjusted emulator, the standardized diagnostics all maintain modest values lying well within ± 1.5 standard deviations giving no cause for concern. We specify θ and σ a priori here, according to the Bayesian paradigm and also mainly for simplicity, but note that the known boundary approach that we describe can be used in combination with various methods of assessing such covariance function parameters (and indeed covariance functions). For example, if one wished to use maximum likelihood, the likelihood calculated given D could also be reduced to tractable form using similar sufficiency arguments, by employing (3.11).

3.3. Updating by further model evaluations. Most importantly, as we have analytic expressions for $\mathrm{E}_K[f(x)]$, $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ we are now able to include additional simulator evaluations into the emulation process. To do this, we perform n (expensive) evaluations, D, of the full simulator across $\mathcal{X}$ and use these to supplement the evaluations, K, available on the boundary. We want to update the emulator by the union of the evaluations D and K, that is, to find $\mathrm{E}_{D\cup K}[f(x)]$, $\mathrm{Var}_{D\cup K}[f(x)]$, and $\mathrm{Cov}_{D\cup K}[f(x), f(x')]$. This can be achieved via a sequential Bayes linear update:

$$\mathrm{E}_{D\cup K}\left[f(x)\right] = \mathrm{E}_K\left[f(x)\right] + \mathrm{Cov}_K\left[f(x), D\right]\mathrm{Var}_K\left[D\right]^{-1}\left(D - \mathrm{E}_K\left[D\right]\right), \tag{3.18}$$

$$\mathrm{Var}_{D\cup K}\left[f(x)\right] = \mathrm{Var}_K\left[f(x)\right] - \mathrm{Cov}_K\left[f(x), D\right]\mathrm{Var}_K\left[D\right]^{-1}\mathrm{Cov}_K\left[D, f(x)\right], \tag{3.19}$$

$$\mathrm{Cov}_{D\cup K}\left[f(x), f(x')\right] = \mathrm{Cov}_K\left[f(x), f(x')\right] - \mathrm{Cov}_K\left[f(x), D\right]\mathrm{Var}_K\left[D\right]^{-1}\mathrm{Cov}_K\left[D, f(x')\right], \tag{3.20}$$

where we first update our emulator analytically by K, and subsequently update these quantities by the evaluations D [17]. As typically n is small due to the relative expense of evaluating the full simulator, these calculations will remain tractable, as $\mathrm{Var}_K[D]^{-1}$ will be feasible for modest n.

Not only will the known boundary $\mathcal{K}$ improve the accuracy of the emulator compared to just updating by D, for only trivial computational cost, it will also allow us to design a more informative set of runs that constitute D. We discuss appropriate designs for this scenario in section 4.

3.3.1. Incorporating known boundaries into black box emulation packages.

Consideration of the form of the sequential update given by (3.18)–(3.20), combined with the sufficiency argument presented in section 3.1, shows that for the full joint update by $D \cup K$, a sufficient set of points is composed of (a) the n points in D, (b) the n points formed from the projection of D onto the boundary $\mathcal{K}$, and (c) the projection x^K of the point of interest x, giving a total of $2n + 1$ points. This has ramifications for users of black box Gaussian process emulation packages (such as BACCO [19] or GPfit [29] in R, or GPy [18] for Python), which perhaps cannot be easily recoded to use the more sophisticated analytic emulation formula of (3.8) and (3.9) but for which the inclusion of extra simulator evaluations is trivial. Hence such a user simply has to add the extra $(n+1)$ projected points on $\mathcal{K}$ to their usual set of n runs, and their black box Gaussian process package will produce results that precisely match (3.18)–(3.20). This, however, will require inverting a matrix of size $(2n + 1)$ and hence will be slower than directly using the above analytic results, which only require inverting a matrix of size n.

3.4. Updating by two perpendicular known boundaries.

Given the above results, we now proceed to discuss the update of the emulator by a second known boundary, $\mathcal{L}$. In the first case, discussed here, $\mathcal{L}$ is assumed perpendicular to $\mathcal{K}$, and in the second case, discussed in section 3.5, $\mathcal{L}$ is parallel to $\mathcal{K}$. Detailed derivations of the results presented can be found in Appendices A and B, and the key results for single and dual boundaries are summarized in Table 1, for ease of comparison.

First, we assume the second known boundary $\mathcal{L}$ is a $d - 1$ dimensional hyperplane, perpendicular to the x_2 direction, as illustrated in Figure 3 (left panel). Our goal is to update the emulator for $f(x)$, $x \in \mathcal{X}$, by our knowledge of the function's behavior on both boundaries $\mathcal{K}$ and $\mathcal{L}$, and subsequently by a set of runs D within $\mathcal{X}$. Thus we must find $\mathrm{E}_{D\cup L\cup K}[f(x)]$ and $\mathrm{Var}_{D\cup L\cup K}[f(x)]$. We do this sequentially by analytically updating by K followed by L, then numerically by D.

As before, assume that the $f(x)$ is analytically solvable and hence inexpensive to evaluate along $\mathcal{L}$, permitting a large but finite number, m, of evaluations on $\mathcal{L}$, denoted $z^{(1)}, \ldots, z^{(m)}$.

Table 1

A summary for comparison of the updated emulator results found for the three main cases: a single boundary, two perpendicular boundaries, and two parallel boundaries. The variance results, which are of course special cases of the covariance results, are included for ease of interpretation.

When updating by one boundary $\mathcal{K}$, with $R_1(a,a') = r_1(a-a') - r_1(a)r_1(a')$ and $\Delta f(.) \equiv f(.) - \mathrm{E}\left[f(.)\right]$: $\begin{aligned} \mathrm{E}_K\left[f(x)\right] &= \mathrm{E}\left[f(x)\right] + r_1(a)\Delta f(x^K) \\ \mathrm{Cov}_K\left[f(x), f(x')\right] &= \sigma^2 R_1(a,a')\, r_{-1}(x^K - x'^K) \\ \mathrm{Var}_K\left[f(x)\right] &= \sigma^2(1 - r_1(a)^2) \end{aligned}$
When updating by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$: $\begin{aligned} \mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}\left[f(x)\right] + r_1(a)\Delta f(x^K) + r_2(b)\Delta f(x^L) - r_1(a)r_2(b)\Delta f(x^{LK}) \\ \mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= \sigma^2 R_1(a,a')\, R_2(b,b')\, r_{-1,-2}(x^{LK} - x'^{LK}) \\ \mathrm{Var}_{L\cup K}\left[f(x)\right] &= \sigma^2(1 - r_1^2(a))(1 - r_2^2(b)) \end{aligned}$
When updating by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$: $\begin{aligned} \mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}\left[f(x)\right] + \left[\frac{r_1(a) - r_1(b)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^K) + \left[\frac{r_1(b) - r_1(a)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^L) \\ \mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= \sigma^2\,\frac{r_{-1}(x^K - x'^K)}{1 - r_1^2(c)}\Big\{r_1(a-a')(1 - r_1^2(c)) - r_1(a)r_1(a') - r_1(b)r_1(b') \\ &\qquad + r_1(c)\left[r_1(a)r_1(b') + r_1(b)r_1(a')\right]\Big\} \\ \mathrm{Var}_{L\cup K}\left[f(x)\right] &= \sigma^2\,\frac{1}{1 - r_1^2(c)}\Big\{1 - r_1^2(c) - r_1^2(a) - r_1^2(b) + 2r_1(c)r_1(a)r_1(b)\Big\} \end{aligned}$

As in section 3.1, we define the corresponding length $m+1$ vector of boundary values L as

$$L = \Big(f(x^L), f(z^{(1)}), \dots, f(z^{(m)})\Big)^T, \tag{3.21}$$

which includes the projection x^L of x onto $\mathcal{L}$. An analogous proof to that of (3.4) gives

$$\mathrm{Cov}_K\left[f(x^L), L\right] \mathrm{Var}_K\left[L\right]^{-1} = (1, 0, \dots, 0), \tag{3.22}$$

while as the product correlation structure is not disturbed by the update by K, we also have

$$\mathrm{Cov}_K\left[f(x), L\right] = r_2(b)\, \mathrm{Cov}_K\left[f(x^L), L\right], \tag{3.23}$$

where b is the perpendicular distance from x to $\mathcal{L}$ and $r_2(\cdot)$ is the correlation function in the perpendicular direction to $\mathcal{L}$, as shown in Figure 3 (left panel). Using (3.22) and (3.23) the expectation of $f(x)$ adjusted by K then L can now be calculated using the sequential update

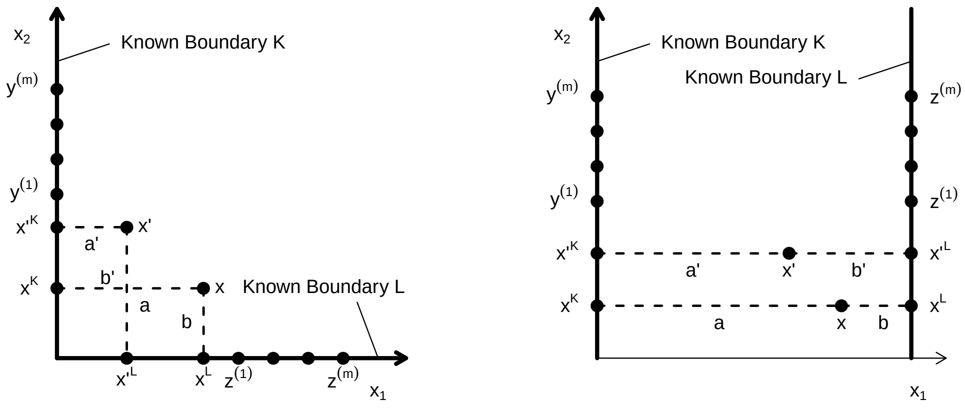


Figure 3. *Left panel: Two perpendicular known boundaries. Right panel: Two parallel known boundaries. In both cases x and x' are the points of interest for the emulation calculation, while x^K and x'^K are their orthogonal projection onto the known boundary $\mathcal{K}$, and x^L and x'^L their orthogonal projection onto the known boundary $\mathcal{L}$. The $y^{(i)}$ and $z^{(i)}$ represent a large number of points on the boundaries $\mathcal{K}$ and $\mathcal{L}$ respectively for which we can evaluate $f(y^{(i)})$ and $f(z^{(i)})$ analytically.*

(3.18) giving

$$\begin{aligned}
\mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}_K\left[f(x)\right] + r_2(b)(1,0,\dots,0)(L-\mathrm{E}_K\left[L\right]) \\
&= \mathrm{E}_K\left[f(x)\right] + r_2(b)(f(x^L) - \mathrm{E}_K\left[f(x^L)\right]) \quad (3.24)\\
&= \mathrm{E}\left[f(x)\right] + r_1(a)(f(x^K) - \mathrm{E}\left[f(x^K)\right]) + r_2(b)f(x^L) \\
&\quad - r_2(b)(\mathrm{E}\left[f(x^L)\right] + r_1(a)(f(x^{LK}) - \mathrm{E}\left[f(x^{LK})\right])) \\
&= \mathrm{E}\left[f(x)\right] + r_1(a)\Delta f(x^K) + r_2(b)\Delta f(x^L) - r_1(a)r_2(b)\Delta f(x^{LK}), \quad (3.25)
\end{aligned}$$

where we have also used (3.8) for $\mathrm{E}_K[f(x)]$, defined $\Delta f(.) \equiv f(.) - \mathrm{E}\left[f(.)\right]$ and denoted the projection of x^L onto $\mathcal{K}$ as x^{LK}, which is just the perpendicular projection of x onto $\mathcal{L}\cap\mathcal{K}$. An expression for the covariance adjusted by K then L is obtained by a similar argument (see Appendix A),

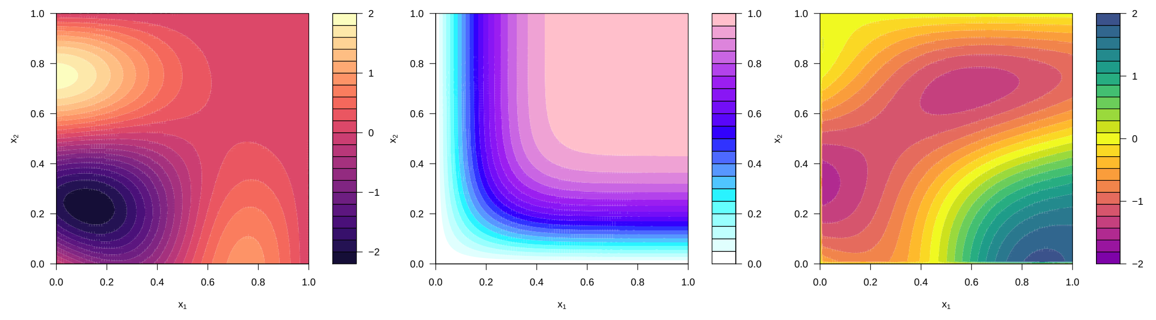
(3.26)
$$\begin{aligned}
\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= r_2(b-b')\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right] - r_2(b)\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right] r_2(b') \\
&= \sigma^2 R_1(a,a')\, R_2(b,b')\, r_{-1,-2}(x^{LK} - x'^{LK}), \quad (3.27)
\end{aligned}$$

where we have defined the correlation function of the projection of x and x' onto $\mathcal{L}\cap\mathcal{K}$ as

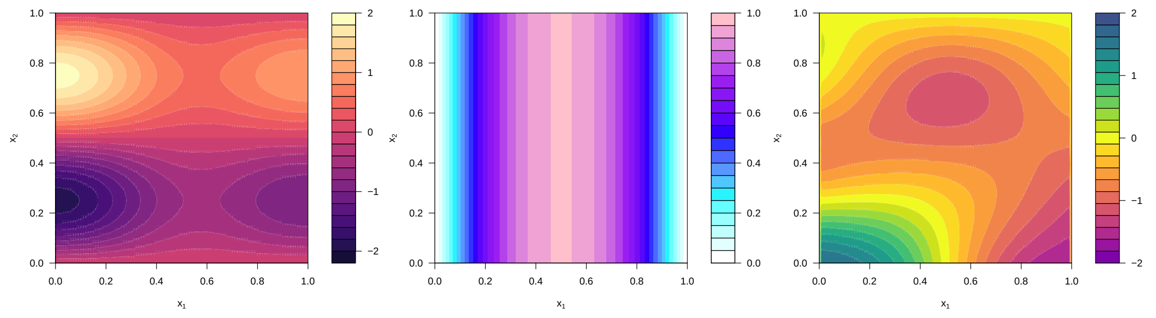
$$r_{-1,-2}(x^{LK} - x'^{LK}) = \prod_{i=3}^{d} r_i(x_i^{LK} - x_i'^{LK}) = \mathrm{Cov}\left[f(x^{LK}), f(x'^{LK})\right]. \quad (3.28)$$

The updated variance is trivially obtained by setting $x = x'$ to get

$$\begin{aligned}
\mathrm{Var}_{L\cup K}\left[f(x)\right] &= \sigma^2 R_1(a,a)\, R_2(b,b) \\
&= \sigma^2(1-r_1^2(a))(1-r_2^2(b)). \quad (3.29)
\end{aligned}$$



(a) $K \perp L$: $\mathrm{E}_{L\cup K}[f(x)]$ (b) $K \perp L$: $\sqrt{\mathrm{Var}_{L\cup K}\left[f(x)\right]}$ (c) $K \perp L$: Diagnostics $S_{L\cup K}(x)$



(d) $K \parallel L$: $\mathrm{E}_{L\cup K}[f(x)]$ (e) $K \parallel L$: $\sqrt{\mathrm{Var}_{L\cup K}\left[f(x)\right]}$ (f) $K \parallel L$: Diagnostics $S_{L\cup K}(x)$

Figure 4. *Emulators updated by two boundaries $\mathcal{K}$ and $\mathcal{L}$. Top row: Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$. Bottom row: Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$.*

As a consistency check, we see that all three expressions (3.25), (3.27), and (3.29) are invariant under interchange of the two boundaries, represented as the transformation $K \leftrightarrow L$ and $a \leftrightarrow b$, as they should be. They also exhibit intuitive limiting behaviors as the distance b from the boundary $\mathcal{L}$ tends to 0 or ∞ (see Appendix A). Again, we observe that were we to sequentially update by a further n evaluations, D, and calculate $\mathrm{E}_{D\cup L\cup K}[f(x)]$ and $\mathrm{Var}_{D\cup L\cup K}[f(x)]$, the only points we require for sufficiency are D and the projections of D and x onto $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{K}\cap\mathcal{L}$. This represents only $4n+3$ points, which is far fewer than the $2(m+1)+1+n$ points (with m extremely large) that we started with. Again, users of black box emulators can easily insert these points, at the cost of having to invert a matrix now of size $4n+3$, instead of a single inversion of size n were they to encode the above analytic results directly.

An example of an emulator updated by two perpendicular known boundaries is shown in Figures 4(a)–4(c), which give $\mathrm{E}_{L\cup K}[f(x)]$, $\sqrt{\mathrm{Var}_{L\cup K}[f(x)]}$, and $S_{L\cup K}(x)$, respectively, for the simple function $f(x)$ introduced in section 3.2. A second known boundary $\mathcal{L}$ is now located at $x_2 = 0$, where we know that $f(x^L) = f(x_1, 0) = -0.9\sin(2\pi x_1)$. As expected, we see the emulator expectation agrees exactly with the behavior of the simulator $f(x)$ on $\mathcal{K}$ and $\mathcal{L}$ (as given in Figure 2(b)). We note also the intuitive property that the variance of the emulator reduces to zero as we approach the boundary but remains at $\sigma^2 = 1$ when we are sufficiently distant. This sensibly represents the increase in knowledge about the simulator behavior the closer we are to $\mathcal{K}$ or $\mathcal{L}$, with the scale of the increase governed by the size of the correlation length parameter θ. Diagnostics $S_{L\cup K}(x)$ are again acceptable.

3.5. Updating by two parallel known boundaries. Consider now a second boundary $\mathcal{L}$ located at $x_1 = c$, that is therefore parallel to the original boundary $\mathcal{K}$ at $x_1 = 0$. As updating by $\mathcal{K}$ leaves the correlation structure $\mathrm{Cov}_K[f(x), f(x')]$ still in product form, critically with respect to the x_1 term, we can still perform a subsequent analytic update by $\mathcal{L}$. We define L as before by (3.21) and denote the distance from point x to its perpendicular projection, x^L, onto $\mathcal{L}$ as b (see Figure 3, right panel), where now $a+b=c$. See Appendix B for more details of the following derivations.

In this parallel case, results analogous to (3.23) and (3.22) can be derived, which combine to give

$$\mathrm{Cov}_K\left[f(x), L\right]\mathrm{Var}_K\left[L\right]^{-1} = R_1^{(2)}(a,c)(1,0,\dots,0), \tag{3.30}$$

where $R_1^{(2)}(a,c) = R_1(a,c)/R_1(c,c)$ and $R_1(\cdot,\cdot)$ is as in (3.12). Inserting into (3.18), the emulator expectation adjusted by both L and K can be shown to be (see Appendix B)

$$\begin{aligned}\mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)(1,0,\dots,0)(L - \mathrm{E}_K\left[L\right])\\ &= \mathrm{E}\left[f(x)\right] + \left[\frac{r_1(a) - r_1(b)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^K) + \left[\frac{r_1(b) - r_1(a)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^L),\end{aligned} \tag{3.31}$$

where we have exploited the fact that the projection of x^L onto $\mathcal{K}$ is just x^K. Similarly, the covariance adjusted by L and K can be shown to be (see Appendix B)

$$\begin{aligned}\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)(1,0,\dots,0)\mathrm{Cov}_K\left[L, f(x')\right]\\ &= \sigma^2\, r_{-1}(x^K - x'^K)\left\{R_1(a,a') - \frac{R_1(a,c)R_1(c,a')}{R_1(c,c)}\right\},\end{aligned}$$

which is just a generalized form of (3.11). We can expand the functions R_1 to obtain

$$\begin{aligned}\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] = \sigma^2\,\frac{r_{-1}(x^K - x'^K)}{1 - r_1^2(c)}\Big\{& r_1(a-a')(1 - r_1^2(c)) - r_1(a)r_1(a') - r_1(b)r_1(b')\\ &+ r_1(c)\big[r_1(a)r_1(b') + r_1(b)r_1(a')\big]\Big\},\end{aligned} \tag{3.32}$$

which we note is now explicitly invariant under interchange of $\mathcal{K}$ and $\mathcal{L}$ (with $a \leftrightarrow b$, etc.) as expected. Again, the adjusted variance is obtained by setting $x = x'$

$$\mathrm{Var}_{L\cup K}\left[f(x)\right] = \sigma^2\,\frac{1}{1 - r_1^2(c)}\Big\{1 - r_1^2(c) - r_1^2(a) - r_1^2(b) + 2r_1(c)r_1(a)r_1(b)\Big\}. \tag{3.33}$$

Again, by inspection of these results we see that the only relevant information for our updated emulator at a general point x is the projections of x onto $\mathcal{K}$ and $\mathcal{L}$. Thus to update the emulator sequentially by K, L, then D, we only need to include the additional $2(n+1)$ points of the projections of D and x onto $\mathcal{K}$ and $\mathcal{L}$, noting that unlike in the perpendicular case, the intersection $\mathcal{L}\cap\mathcal{K}$ is now empty.

3.6. Continuous known boundaries. We now show how to generalize the above calculations from using a slightly artificial discrete and finite set of m known points on each boundary, which only requires a standard Bayes linear update, to a more natural continuum of known points on a continuous boundary, which requires a generalized Bayes linear update. Let the points on the boundary $\mathcal{K}$ perpendicular to x_1 be denoted $K = \{f(y) : y \in \mathcal{K}\}$. The Bayes linear update can be generalized from the case of finite points to that of a continuum of points in the following way (we are unaware of this generalization having been performed previously, but note that it follows from the foundational position that views the Bayes linear update as a projection [17]). The adjusted expectation changes from the matrix equation,

$$\mathrm{E}_K[f(x)] = \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\,\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]),$$

to the integral equation,

$$\mathrm{E}_K[f(x)] = \mathrm{E}[f(x)] + \int_{y\in\mathcal{K}}\int_{y'\in\mathcal{K}} \mathrm{Cov}[f(x), f(y)]\ s(y,y')\ (f(y') - \mathrm{E}\left[f(y')\right])dydy', \tag{3.34}$$

and the covariance update becomes

$$\begin{aligned}\mathrm{Cov}_K\left[f(x), f(x')\right] &= \mathrm{Cov}\left[f(x), f(x')\right] \\ &\quad - \int_{y\in\mathcal{K}}\int_{y'\in\mathcal{K}} \mathrm{Cov}[f(x), f(y)]\ s(y,y')\ \mathrm{Cov}\left[f(y'), f(x')\right]\ dydy'.\end{aligned}$$

Here $s(x,x')$ represents the infinite dimensional generalization of $\mathrm{Var}[K]^{-1}$ and satisfies the equivalent inverse property to that of (3.4) giving

$$\int_{y'\in\mathcal{K}} \mathrm{Cov}\left[f(y), f(y')\right] s(y',y'')\ dy' = \delta(y - y'') \qquad \text{for } y, y'' \in \mathcal{K},$$

where $\delta(y-y'')$ is the Dirac delta function, the generalization of the identity matrix. Again, if we denote the projection of a general point $x \in \mathcal{X}$ onto $\mathcal{K}$ as x^K, we have for $y \in \mathcal{K}$ that

$$\mathrm{Cov}[f(x), f(y)] = r_1(a)\,\mathrm{Cov}\left[f(x^K), f(y)\right], \tag{3.35}$$

which on substitution into (3.34) yields

$$\begin{aligned}\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + \int_{y\in\mathcal{K}}\int_{y'\in\mathcal{K}} r_1(a)\,\mathrm{Cov}\left[f(x^K), f(y)\right]\ s(y,y')\ (f(y') - \mathrm{E}\left[f(y')\right])dydy' \\ &= \mathrm{E}[f(x)] + r_1(a)\int_{y'\in\mathcal{K}} \delta(x^K - y')\ (f(y') - \mathrm{E}\left[f(y')\right])dy' \\ &= \mathrm{E}[f(x)] + r_1(a)\ (f(x^K) - \mathrm{E}\left[f(x^K)\right])\end{aligned}$$

in agreement with (3.8). Similarly the updated covariance becomes

$$\begin{aligned}\mathrm{Cov}_K\left[f(x), f(x')\right] &= \mathrm{Cov}\left[f(x), f(x')\right] - r_1(a)\int_{y'\in\mathcal{K}} \delta(x^K - y')\ \mathrm{Cov}\left[f(y'), f(x'^K)\right] r_1(a')\ dy' \\ &= \mathrm{Cov}\left[f(x), f(x')\right] - r_1(a)\mathrm{Cov}\left[f(x^K), f(x'^K)\right] r_1(a')\end{aligned}$$

in agreement with (3.10), and the derivation of (3.11) then follows in exactly the same way as shown in section 3.1. The continuous versions of the two perpendicular and parallel boundary cases follow similarly, are given in Appendix C, and also agree with the discrete results.

4. Design of KBE experiments.

The existence of known boundaries allows us to design a more efficient set of runs over $\mathcal{X}$ to exploit this additional information. Standard computer model designs involving Latin hypercubes or low-discrepancy sequences [36] are of limited value here, as they seek uniform coverage over $\mathcal{X}$. As highlighted by Figure 4, after the boundary update the emulator variance will now exhibit clear (non-uniform) structure, which the design should now reflect. We therefore investigate some simple methods of optimal design and introduce a mechanism for transforming a uniformly space-filling design into a more appropriate configuration for KBE problems. The general design problem is as follows (see, for example, [26]):

- Given a simulator and corresponding emulator updated by known boundary $\mathcal{K}$, select input points $X_D \in \mathcal{X}$, that will give evaluations $D = f(X_D)$, chosen to optimize some criterion $c(X_D)$.

Typically, the criterion is such that we seek to maximize the information content of the chosen design X_D, which in computer models typically translates to minimizing a function of the emulator variance $\mathrm{Var}_{D\cup K}[f(x)]$ over $\mathcal{X}$ [37]. Due to the discrete nature of computer experiments, the criterion over $\mathcal{X}$ is typically approximated by some discrete grid X over $\mathcal{X}$. The optimization problem then becomes one of a search over a collection of candidate designs for the "best" candidate under the specified approximate criterion, yielding a locally (not globally) optimal design. This is usually sufficient, as the identification of the global optimum would only be warranted if all the assumptions used in the emulator construction process were thought to be highly accurate, which is rarely the case. A sensible choice for the design criterion, $c(X_D)$, suitable for our purposes is as follows:

- *V-optimality*: $c(X_D) = \mathrm{trace}(\mathrm{Var}_{D\cup K}[f(X)])$, the trace of the adjusted emulator variance matrix over the finite grid of points X, given the known boundary and the design X_D.

V-optimality just seeks to minimize the sum (and hence also the mean) of the point variances across X, calculated in the presence of known boundaries, and is a discrete approximation of the integrated mean squared prediction error criterion [37, 26]. To improve efficiency we exploit the fact that

$$\mathrm{trace}(\mathrm{Var}_{D\cup K}\,[f(X)]) = \mathrm{trace}(\mathrm{Var}_{K}\,[f(X)]) - \mathrm{trace}(\mathrm{RVar}_{D\cup K}(f(X))) \tag{4.1}$$

(see (3.19)) and so we simply need to seek designs that maximize $\mathrm{trace}(\mathrm{RVar}_{D\cup K}(f(X)))$.

Figure 5 shows 10-point V-optimal designs X_D (black points) and the corresponding emulator standard deviation $\sqrt{\mathrm{Var}_{D\cup K}[f(X)]}$ defined over $\mathcal{X}$ (colored contours) for the single known boundary case (top left), two perpendicular known boundaries (middle left), and two parallel known boundaries (bottom left). The V-optimality criteria $c(X_D)$ was assessed using a grid X of size 30×30 and calculated for any particular candidate design X_D using (4.1), (3.20), and, for the single known boundary case, (3.11). The designs in Figure 5 were generated using the standard optim() function in R, using 10-point space filling Latin hypercube designs as initial conditions. Note how the V-optimality criteria automatically moves the design points away from the known boundaries toward the less explored regions of $\mathcal{X}$, while still maintaining excellent space filling properties. The toy model was subsequently evaluated at X and the corresponding emulator diagnostics $S_{D\cup K}(x)$ given in the right column of Figure 5.

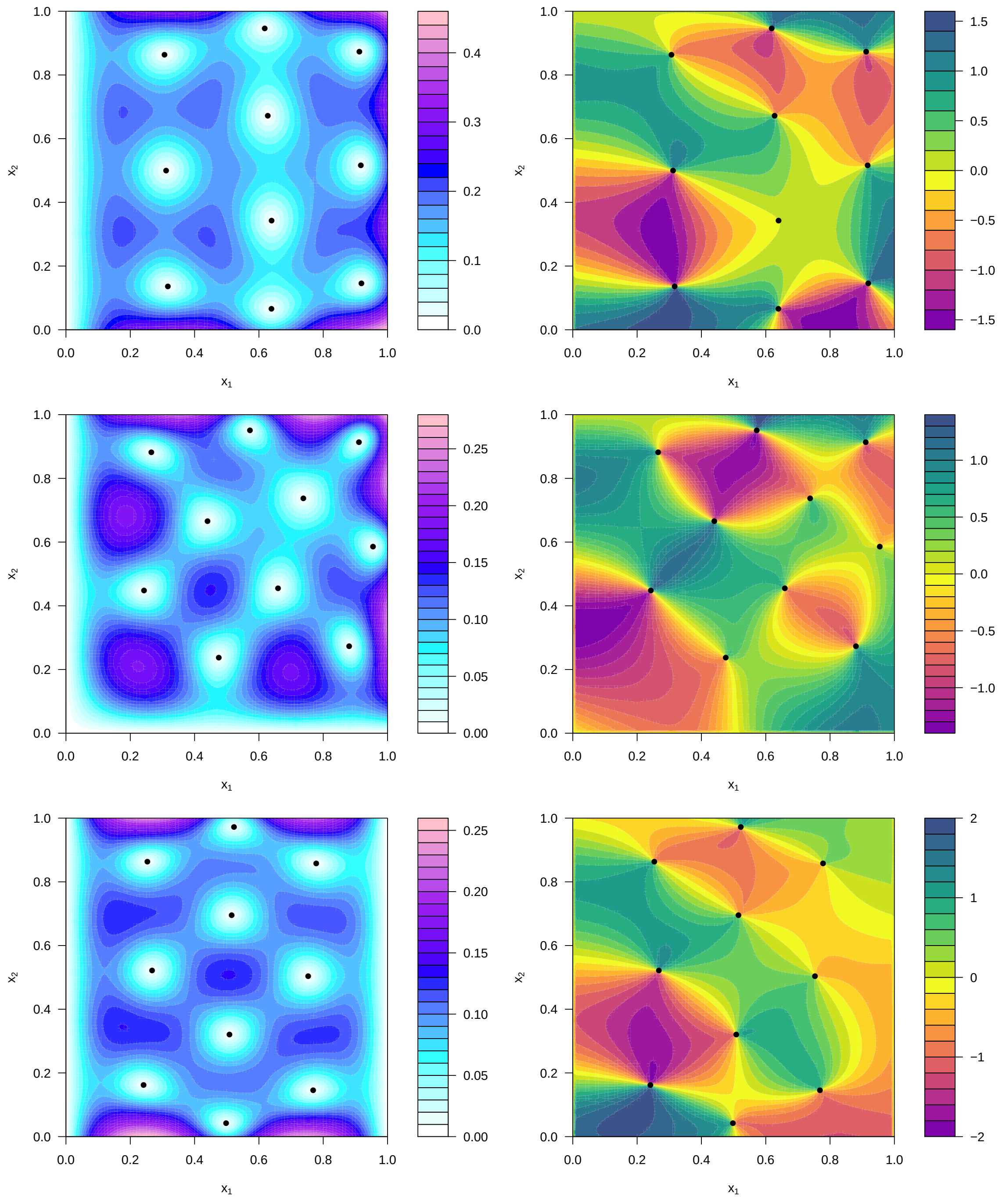


Figure 5. *Left column: 10-point V-optimal designs X_D (black points) and the corresponding emulator standard deviation $\sqrt{\mathrm{Var}_{D\cup K}[f(X)]}$ defined over $\mathcal{X}$ (colored contours) for the single known boundary case (top left), two perpendicular known boundaries (middle left), and two parallel known boundaries (bottom left), respectively. Emulator diagnostics of the form $S_{D\cup K}(x) = (\mathrm{E}_{D\cup K}[f(x)] - f(x))/\sqrt{\mathrm{Var}_{D\cup K}[f(x)]}$ are given over $\mathcal{X}$ in the right column for each of the three cases.*

Despite the desirable properties of such V-optimal designs, the projections onto lower dimensional subspaces of $\mathcal{X}$ can be less than satisfactory. For example, in the single and also two parallel known boundary cases as illustrated in the top-left and bottom-left panels of Figure 5, the projection of X_D onto x_1 only covers three distinct values of x_1. This could be very inefficient if it was found that x_1 was highly influential (and hence deemed an active input) while x_2 was found to be inactive. This is important as active variables are extremely useful in taming high-dimensional simulators [41].

Such projection concerns may promote the use of a more general purpose design. As mentioned above, in the computer model literature the maximin Latin hypercube is the standard choice [36]. Therefore, to account for the nonuniformity of the boundary updated emulator variance $\mathrm{Var}_{D\cup K}[f(X)]$, here we explore the use of simple "warped Latin hypercube designs" that share the useful properties of standard Latin hypercubes but are adapted to be more appropriate for a known boundary setting. These do not optimize any particular criteria but have good space filling and projection properties.

The warped designs are created by taking a maximin Latin hypercube design and warping it so that the marginal density of the design matches the marginal form of the new emulator variance adjusted by the known boundaries, which in the single or two perpendicular boundary cases is proportional to $(1 - r_i^2(a))$, as shown by (3.9) and (3.29). For example, in the two perpendicular known boundary case, each point $x^{(i)}$ in a maximin Latin hypercube design is warped via the transformation

$$x_1^{(i)} \to g_1(x_1^{(i)})/g_1(1), \qquad \text{where} \quad g_1^{-1}(a) = \int_0^a (1 - r_1^2(a'))da', \tag{4.2}$$

$$x_2^{(i)} \to g_2(x_2^{(i)})/g_2(1), \qquad \text{where} \quad g_2^{-1}(b) = \int_0^b (1 - r_2^2(b'))db', \tag{4.3}$$

$$x_j^{(i)} \to x_j^{(i)} \qquad j \neq 1, 2, \tag{4.4}$$

which ensures the marginal distributions $\pi(x_j^{(i)}) \propto (1 - r_j^2(x_j^{(i)}))$, for $j = 1, 2$, as required.

Figure 6 (left panel) shows a 20-point maximin Latin hypercube as the red points and the warped Latin hypercube as the black points. The black lines link the pre- and postwarped points to highlight the effect of the warping. The right panel shows the emulator standard deviation $\sqrt{\mathrm{Var}_{D\cup L\cup K}[f(X)]}$ updated by both boundaries L and K and the warped design D. Such designs are space filling, while they also maintain good projection properties. We illustrate these designs further in the next section to explore improvements to the KBE of a model of *Arabidopsis thaliana*.

5. Application to a systems biology model of *Arabidopsis*.

In the previous sections of this article we have presented methodology for utilizing knowledge of the behavior of a complex scientific model along particular boundaries of the input parameter space to aid emulation of the model across the whole input space, exploiting both emulation and design procedures. In this section we apply this methodology to a model of hormonal crosstalk in the root of an *Arabidopsis* plant.

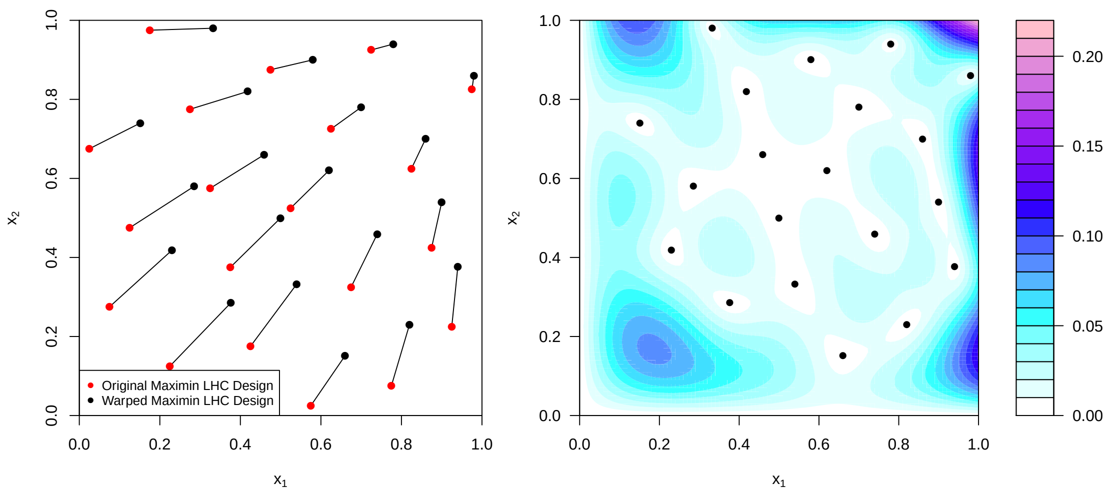


Figure 6. *Left panel: A 20-point maximin Latin hypercube (red points) and the corresponding warped Latin hypercube (black points). The black lines link the pre- and postwarped points to highlight the effect of the warping. Right panel: The emulator standard deviation* $\sqrt{\mathrm{Var}_{D \cup L \cup K}[f(X)]}$ *updated by both boundaries* L *and* K *and the warped design* D*.*

5.1. Model of hormonal crosstalk in *Arabidopsis thaliana*. *Arabidopsis thaliana* is a small flowering plant that is widely used as a model organism in plant biology. *Arabidopsis* offers important advantages for basic research in genetics and molecular biology for many reasons, including the facts that it has a short life cycle, changes in it are easy to observe, and it is genetically relatively simple. *Arabidopsis* was therefore the first plant to have its genome fully sequenced [23]. Understanding the genetic structure of *Arabidopsis* may lead to increased understanding of crop plants such as wheat and hence facilitate the future development of crop plants that are robust to adverse climate conditions.

We demonstrate our KBE techniques on a model of hormonal crosstalk in the root of an *Arabidopsis* plant that was constructed by Liu et al. [28]. This *Arabidopsis* model represents the hormonal crosstalk of auxin, ethylene, and cytokinin in *Arabidopsis* root development as a set of 18 differential equations, given in Table 2, which must be solved numerically. The full model takes an input vector of 45 rate parameters $(k_1, k_{1a}, k_2, \ldots)$, produces an output vector of 18 chemical concentrations $([Auxin], [X], [PLSp], \ldots)$, and has a typical runtime of approximately 0.1 to 1 seconds, depending on parameter settings and target model time [45]. This *Arabidopsis* model has been successfully emulated in the literature in the context of history matching [24, 45]. For the purposes of this article, we are interested in modeling the important output $[PLSp]$, which represents the concentration of POLARIS peptide [28] at early time, $t = 2$. We choose to explore six input rate parameters $\{k_4, k_6, k_{6a}, k_7, k_8, k_9\}$ of primary interest, although it is important to note that the benefits of using known boundaries would scale to larger numbers of inputs. The ranges over which we allowed these six inputs to vary is given in Table 6 in Appendix D. These ranges were square rooted and mapped to a $[-1, 1]$ scale. The remaining input rate parameters were fixed at values deemed reasonable by biological experts [28].

Table 2
Arabidopsis *model differential equations.*

$$\begin{aligned}
\frac{d[Auxin]}{dt} &= \frac{k_{1a}}{1+\frac{[X]}{k_1}} + k_2 + k_{2a}\frac{[ET]}{1+\frac{[CK]}{k_{2b}}}\frac{[PLSp]}{k_{2c}+[PLSp]} \\
&\quad + \frac{V_{IAA}[IAA]}{Km_{IAA}+[IAA]} \\
&\quad - \left(k_3 + \frac{k_{3a}[PIN1pm]}{k3auxin+[Auxin]}\right)[Auxin] \\
\frac{d[X]}{dt} &= k_{16} - k_{16a}[CTR1^*] - k_{17}[X] \\
\frac{d[PLSp]}{dt} &= k_8[PLSm] - k_9[PLSp] \\
\frac{d[Ra]}{dt} &= -k_4[Auxin][Ra] + k_5[Ra^*] \\
\frac{d[Ra^*]}{dt} &= k_4[Auxin][Ra] - k_5[Ra^*] \\
\frac{d[CK]}{dt} &= \frac{k_{18a}}{1+\frac{[Auxin]}{k_{18}}} - k_{19}[CK] \\
&\quad + \frac{V_{CK}[cytokinin]}{Km_{CK}+[cytokinin]} \\
\frac{d[ET]}{dt} &= k_{12} + k_{12a}[Auxin][CK] - k_{13}[ET] \\
&\quad + \frac{V_{ACC}[ACC]}{Km_{ACC}+[ACC]} \\
\frac{d[PLSm]}{dt} &= \frac{k_6[Ra^*]}{1+\frac{[ET]}{k_{6a}}} - k_7[PLSm]
\end{aligned}$$

$$\begin{aligned}
\frac{d[Re]}{dt} &= k_{11}[Re^*][ET] - (k_{10} + k_{10a}[PLSp])[Re] \\
\frac{d[Re^*]}{dt} &= -k_{11}[Re^*][ET] + (k_{10} + k_{10a}[PLSp])[Re] \\
\frac{d[CTR1]}{dt} &= -k_{14}[Re^*][CTR1] + k_{15}[CTR1^*] \\
\frac{d[CTR1^*]}{dt} &= k_{14}[Re^*][CTR1] - k_{15}[CTR1^*] \\
\frac{d[PIN1m]}{dt} &= \frac{k_{20a}}{k_{20b}+[CK]}[X]\frac{[Auxin]}{k_{20c}+[Auxin]} \\
&\quad - k_{1_v 21}[PIN1m] \\
\frac{d[PIN1pi]}{dt} &= k_{22a}[PIN1m] - k_{1_v 23}[PIN1pi] \\
&\quad - k_{1_v 24}[PIN1pi] + \frac{k_{25a}[PIN1pm]}{1+\frac{[Auxin]}{k_{25b}}} \\
\frac{d[PIN1pm]}{dt} &= k_{1_v 24}[PIN1pi] - \frac{k_{25a}[PIN1pm]}{1+\frac{[Auxin]}{k_{25b}}} \\
\frac{d[IAA]}{dt} &= 0 \\
\frac{d[cytokinin]}{dt} &= 0 \\
\frac{d[ACC]}{dt} &= 0
\end{aligned}$$

5.2. Known boundary emulation of the *Arabidopsis* model.

5.2.1. Establishing known boundaries. Establishing known boundaries requires some understanding of the scientific model. It is not uncommon for one or more known boundaries to occur in a model system for some outputs. Often, setting certain parameters to specific values will decouple smaller subsections of the system, which may allow subsets of the equations to be solved analytically, for particular outputs. This is the case for the *Arabidopsis* model.

We establish the known boundaries for output $[PLSp]$ by considering its rate equation:

$$\frac{d[PLSp]}{dt} = k_8[PLSm] - k_9[PLSp]. \tag{5.1}$$

A known boundary exists when rate parameter $k_8 = 0$, since in this case

$$\frac{d[PLSp]}{dt} = -k_9[PLSp] \tag{5.2}$$

$$\Rightarrow \quad [PLSp] = [PLSp^0]e^{-k_9 t}, \tag{5.3}$$

where $[PLSp^0]$ is the initial condition of the $[PLSp]$ output, and we see that $[PLSp]$ has been entirely decoupled from the rest of the system. The output $[PLSp]$ can now be obtained along the boundary $k_8 = 0$ with negligible computational cost.

The second (perpendicular) known boundary for output $[PLSp]$ occurs when $k_6 = 0$. This decouples the combined system of $[PLSm]$ and $[PLSp]$. We can solve for $[PLSm]$ first using

$$\frac{d[PLSm]}{dt} = -k_7[PLSm] \tag{5.4}$$

$$\Rightarrow \quad [PLSm] = [PLSm^0]e^{-k_7 t} \tag{5.5}$$

Inserting this solution for $[PLSm]$ into the rate equation for $[PLSp]$ then yields

$$[PLSp] = [PLSp^0]e^{-k_9 t} + \frac{k_8[PLSm^0]}{k_9 - k_7}(e^{-k_7 t} - e^{-k_9 t}), \tag{5.6}$$

which again requires negligible computational cost to evaluate for any given input combination. We now use these known boundaries to aid emulation of $[PLSp]$ in the *Arabidopsis* model.

5.2.2. Emulator structure and parameter specification. The emulation strategy used is as follows. As discussed in section 2, we restrict the form of our emulator to a pure Gaussian process, as given by (2.1). We used a product Gaussian correlation function of the form given in (2.3), as we assumed the solution to the *Arabidopsis* model would most likely be smooth and that many orders of derivatives would exist. The prior emulator expectation and variance were taken to be constant, that is, $\mathrm{E}[f(x)] = \beta$ and $\mathrm{Var}[f(x)] = \sigma^2$, where β and σ^2 were estimated to be the sample mean and variance of a set of previously evaluated scoping runs. The correlation length parameter θ was set to $\theta = 0.7$ for each input, a choice consistent with the argument for approximately assessing correlation lengths presented in [41]. This value for θ was also checked for adequacy using standard emulator diagnostics [4]. We have made this relatively simple emulator specification for illustrative purposes, so that we can focus on the effect of the inclusion of known boundaries.

5.2.3. Results. We now compare emulators of the above form constructed both with and without use of the known boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$, and with and without the addition of training points. In this section we fix the design for the training points as a maximin Latin hypercube design of size 60 across the six dimensional input space and explore the effects of more tailored designs in section 5.3. Bayes linear updates by one and two known boundaries were carried out using the single and two perpendicular boundary updates given by (3.8), (3.9), (3.11) and (3.25), (3.27), (3.29), respectively. Additional updating using the set of training points D was then performed using the sequential update formula given by (3.18)–(3.20).

We use visual representations of the emulators and various diagnostics in order to compare emulators built under the six scenarios of interest. These will be referred to using numerical labeling as follows, with the data used to update the emulators given in parentheses:

1. prior emulator beliefs only, no training points and no known boundaries: $(\emptyset)$;
2. single known boundary $k_6 = 0$, no training points: (K);

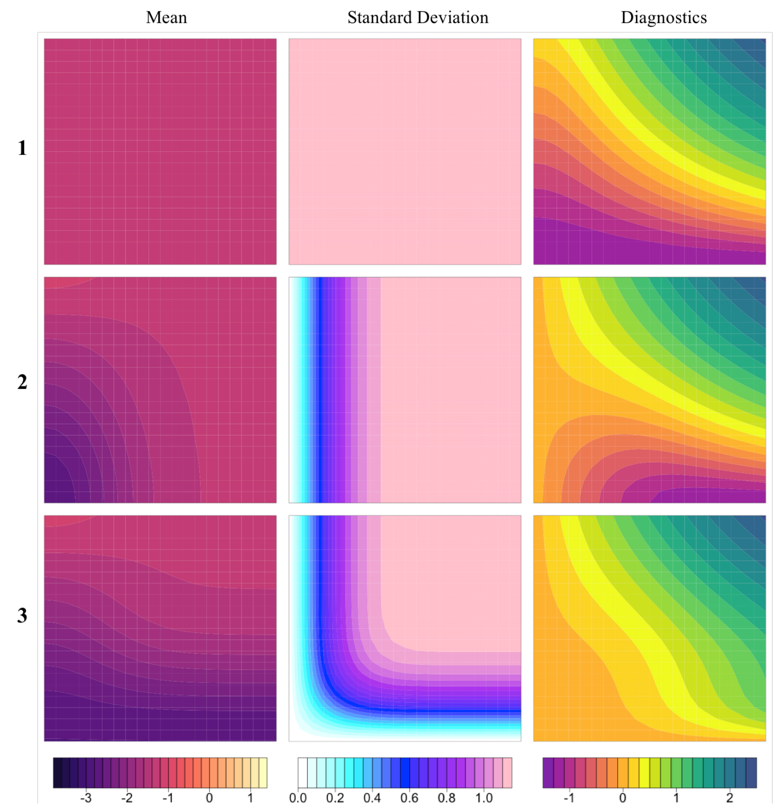


Figure 7. *Results of emulating, without training points, a two-dimensional k_6 (x-axes) by k_8 (y-axes) slice of the six-dimensional input space, with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. The first row shows the results when using prior emulator beliefs only, the second row shows the results when updating by the boundary $\mathcal{K} : k_6 = 0$ only, and the third row shows the results when updating using both boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$. Each column from left to right shows emulator mean, standard deviation, and diagnostics, respectively.*

3. two perpendicular known boundaries $k_6 = 0$ and $k_8 = 0$, no training points: $(L \cup K)$;
4. training points only: (D);
5. single known boundary and training points: $(D \cup K)$;
6. two perpendicular known boundaries $k_6 = 0$ and $k_8 = 0$, and training points: $(D \cup L \cup K)$.

Plots equivalent to those shown in Figures 2 and 4 are substantially more difficult to visualize across all dimensions of a high-dimensional space. Instead, to show intuitively the effect of the various known boundaries, we first examine a slice of the full six-dimensional space. Figures 7 and 8 show the results of emulating, with and without training points using no, one, and two boundaries, respectively, a two-dimensional k_6 (x-axes) by k_8 (y-axes) slice of the six-dimensional input space, with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. The rows are labeled in terms of the above six scenarios, and the columns give the emulator mean, standard deviation, and diagnostics, defined as in section 3.2. These figures can be compared to the true function, shown in Figure 9.

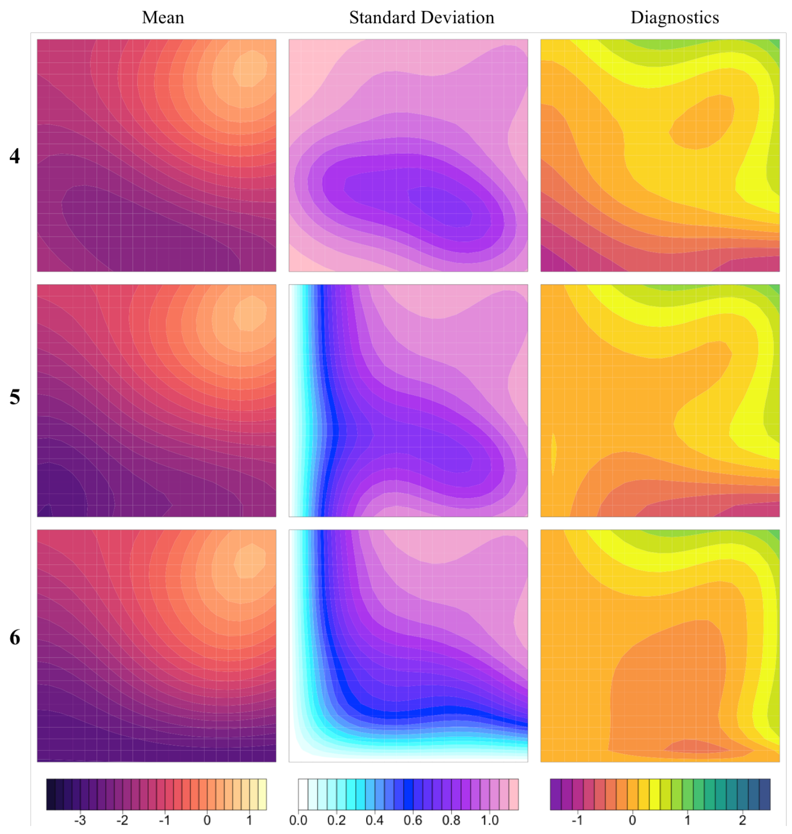


Figure 8. *Results of emulating, with training points, a two-dimensional k_6 (x-axes) by k_8 (y-axes) slice of the six-dimensional input space, with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. The first row shows the results when updating by the training points only, the second row shows the results when updating by the training points and the known boundary $\mathcal{K} : k_6 = 0$, and the third row shows the results when updating by the training points and the two known boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$. Each column from left to right shows emulator means, variances, and diagnostics, respectively.*

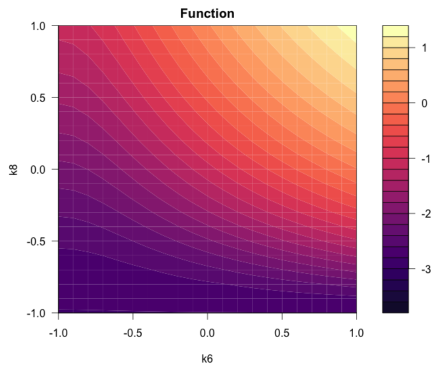


Figure 9. *A $k_6 \times k_8$ cross section of the simulator output with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. This should be compared with the left column of Figures 7 and 8.*

Figure 7 shows that updating using the boundary $k_6 = 0$ results in an updated mean near to the boundary which closely reflects the true function, while further away from the boundary it tends back toward the prior mean. The standard deviation tends to zero at the boundary and increases further away from it, tending back toward the prior standard deviation. The diagnostic plots show that the emulator gives acceptable predictions across the input space, tending to zero at the boundary. Introducing the second boundary results in accurate predictions close to both boundaries and acceptable diagnostic plots. Behavior of the mean and variance tends to the prior specification in the sections of the input space far from both boundaries.

Figure 8 shows that emulator variance modestly decreases when the 60 training points are incorporated, a result which is sensitive to how close any of the training runs are to this particular slice. The emulator mean does show noticeable improvement, but note that the inclusion of the two boundaries $\mathcal{K}$ and $\mathcal{L}$ still has a far more significant effect on the emulator than that of the 60 runs. The diagnostic plots are comparable to those in Figure 7, the most notable difference being the diagnostic values at the top right corner of the input space, which have now been reduced. Diagnostic plots such as these have been compared at several combinations of the other input values with similarly adequate results, and we examine more comprehensive diagnostics below. Since the correlation structure is more heavily influential for updating our beliefs of simulator behavior when known boundaries are utilized, it is even more important to ensure that parameters of the correlation function have been adequately specified, in particular the correlation lengths. Large amounts of poor diagnostics for points near the boundary may indicate that the correlation length has been overestimated—an easy mistake if the function rapidly changes its behavior as it moves away from the boundary.

Figures 7 and 8 demonstrate a major advantage of being able to update simulator beliefs using known boundaries over just using individual points. Individual points are usually large distances away from each other in high dimensions. However, as can be seen from these variance plots (and would similarly be shown by any other slice with different values of the four fixed inputs), the known boundaries here are $d-1$ dimensional objects and hence carry far more information than individual runs (which are 0-dimensional objects), which results in significant variance resolution across substantial amounts of the input space for very little computational cost.

We now perform a more detailed comparison by evaluating the emulators over a fixed set of 2000 diagnostic points, which form a maximin Latin hypercube. Figure 10 shows $f(x)$ against $\mathrm{E}_D[f(x)]$ for the set of 2000 diagnostic points for the six scenarios outlined above. We divide the points according to their (k_6, k_8) coordinates (each scaled to $[-1, 1]$) as follows: blue points are such that $k_6 > -0.5$ and $k_8 > -0.5$, green points have $k_6 < -0.5$ and $k_8 > -0.5$, purple points have $k_6 > -0.5$ and $k_8 < -0.5$, and orange points have $k_6 < -0.5$ and $k_8 < -0.5$. The red line is the function $y = x$. Panel 2 shows that updating the emulator mean by the single boundary $\mathcal{K} : k_6 = 0$ results in larger changes in the mean prediction toward the true value for (green and orange) points close to that boundary. We notice that, although they are affected, there are relatively large numbers of blue and purple points for which the prediction is largely unchanged from the prior specification. Panel 3 shows that incorporating the second boundary into the emulation process results in (purple) points close to that boundary having greatly altered emulator mean values toward the true simulator

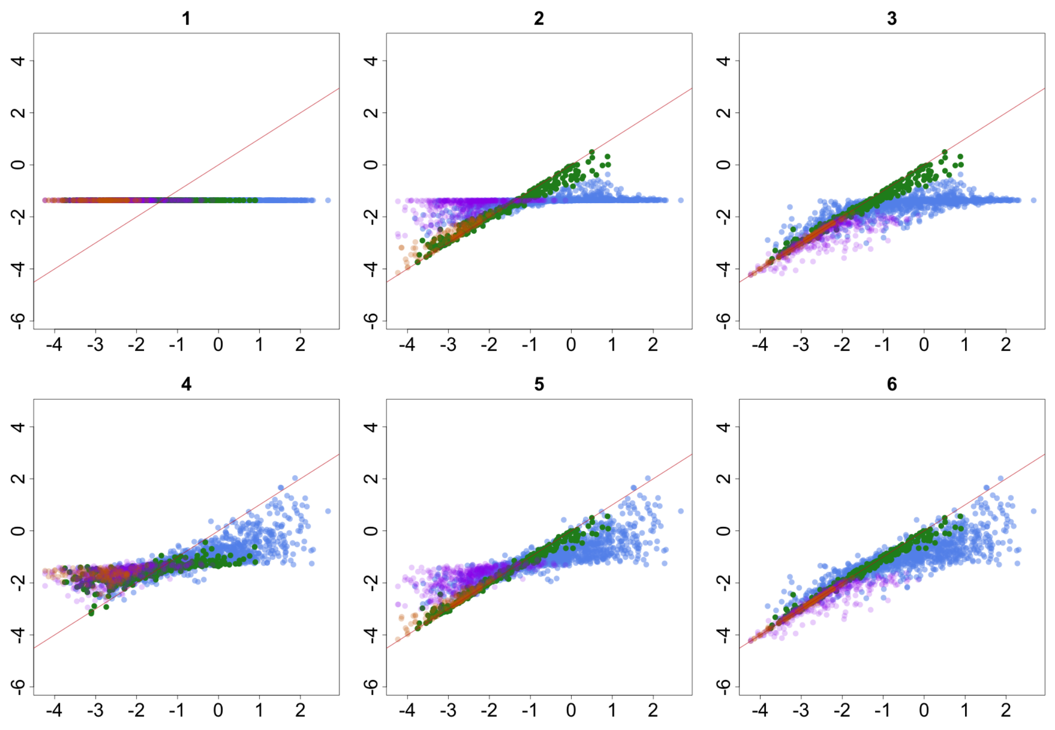


Figure 10. *$f(x)$ against $\mathrm{E}_D[f(x)]$ for the diagnostic set of 2000 points. Blue points are such that $k_6 > -0.5$ and $k_8 > -0.5$, green points are such that $k_6 < -0.5$ and $k_8 > -0.5$, purple points are such that $k_6 > -0.5$ and $k_8 < -0.5$, and orange points are such that $k_6 < -0.5$ and $k_8 < -0.5$. The red line is the function $y = x$. The columns (from left to right) show the results of emulating without boundaries, with one boundary and with two boundaries. The rows show the results of emulating without training points on the top and with training points on the bottom.*

values. Orange points, which are close to both boundaries, have their accuracy increased even further, with many of them lying very close to the line $y = x$. Panels 4 to 6 show the effect of using training points in the construction of the emulators. The effect of updating our beliefs about the simulator output at any particular point in the input space depends on its location relative to the training points. In the case when beliefs have been updated using both boundaries, subsequent updating using training points informs us most about the blue points, namely, those which are far from the boundaries. This suggests that training point design should be affected by knowledge of boundary behavior such that a greater increase in accuracy in those areas largely unaffected by (that is far from) the boundaries is obtained. We now explore these design issues in section 5.3. We also give further diagnostic analysis of the 2000 points in Appendix D.

5.3. Simulation study of KBE design. We now compare emulators constructed using various training point designs, introduced in section 4, that exploit the known boundaries. We wish to explore the improvements to the emulators due to such designs, compared to the improvements seen from just using the known boundaries directly, as were examined in the previous section. We do this by comparing the use of several designs using the root mean

Table 3

A table of RMSEs of the 2000 *diagnostic points using emulators constructed with and without both the known boundaries $\mathcal{K}$ and $\mathcal{L}$ for a maximin Latin hypercube design and the warped version of this design for two choices of correlation length θ. The numbers in bold correspond to the preferred strategy for the given knowledge of the boundaries.*

θ	Known boundaries	Maximin LH	Warped maximin LH
0.7	Without	**0.9247**	0.9489
	With	0.6763	**0.5886**
1.2	Without	**0.4427**	0.6601
	With	0.2986	**0.2530**

square error (RMSE) of the 2000 diagnostic training points obtained in section 5.2.2 under knowledge of the boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$. First, we demonstrate that a warped maximin Latin hypercube is preferable to a standard maximin Latin hypercube. We then compare the chosen design of three V-optimality design procedures; two take account of the known boundaries and one doesn't.

5.3.1. Warped maximin Latin hypercube designs.

We generated 1000000 Latin hypercubes of size 60 over the six-dimensional input space and chose the one with maximal minimum distance between any two of its points. We compare the emulator constructed using this design with that constructed using the warped version of this design, constructed using (4.2), (4.3), and (4.4). Table 3 shows the RMSEs of the 2000 diagnostic points for emulators constructed with and without both the known boundaries for each of these two designs for two choices of correlation length parameter θ.

We observe that the RMSE is greatly improved when the known boundaries are incorporated into the construction process of the emulator, relative to when they are not included, as expected. It is also the case that the RMSE shows noticeable improvement for the emulators constructed using the warped design, compared to the standard design, when the known boundaries are utilized, for both choices of correlation length. This suggests that a warped maximin Latin hypercube is indeed a reasonable general purpose design to use if known boundaries are present, which maintains the good projection properties of a Latin hypercube, as discussed in section 4.

5.3.2. Approximate V-optimality designs.

Finding global V-optimal designs such as shown in Figure 5 is extremely computationally demanding and is impractical for a moderate to high run number. We instead investigate three approximately V-optimal 60-point designs, constructed as follows, the first being constructed without including the known boundaries and the second two including them.

For the first design, which ignores the known boundaries, we iteratively chose individual design points that optimize the current V-optimal criteria, that is, the ith point was chosen to optimize $\text{trace}(\text{Var}_{D_i \cup D_{i-1}}[f(X)])$, given that the previous $(i-1)$ points, as represented by D_{i-1}, had already been chosen in the previous iterations. This is highly unlikely to lead to a global solution but should result in designs which are quick to generate and that have high V-optimality criteria that are good enough for our purposes. The second design was created by warping the first design, using (4.2), (4.3), and (4.4) as in the previous section. The third

Table 4

A table of RMSEs of the 2000 *diagnostic points using emulators constructed with and without the known boundaries $\mathcal{K}$ and $\mathcal{L}$ for three designs, namely, a standard iterative V-optimal design without the known boundaries, the warped version of this design, and an iterative V-optimal design which takes account of the known boundaries.*

θ	Known boundaries	Iterative V-optimal	Warped iterative V-optimal	Iterative V-optimal with known boundaries
0.7	Without	**0.8166**	0.9013	0.9700
	With	0.5815	**0.5091**	**0.5101**
1.2	Without	**0.4476**	0.6687	0.9028
	With	0.2830	**0.2340**	**0.2414**

design used the known boundaries $\mathcal{K}$ and $\mathcal{L}$ and was generated in a similar iterative manner to the first design, but now the ith point was chosen to optimize trace($\text{Var}_{D_i \cup D_{i-1} \cup K \cup L}[f(X)]$). In this case we expect the points to land further away from the two boundaries, similar to the designs shown in Figure 5. In all three cases X was chosen to be a 6^6 grid across the six dimensional input space, which represents a pragmatic approximation to limit the design calculation time.

Table 4 shows the RMSEs of the 2000 diagnostic points for emulators constructed with and without the known boundaries $\mathcal{K}$ and $\mathcal{L}$ for each of the above three designs: iterative V-optimal, warped iterative V-optimal, and iterative V-optimal with known boundaries. We give the results for two values of the correlation length θ. The RMSE numbers in bold correspond to the appropriate design for that scenario, with the other numbers provided for a fair comparison, for example, if we are not aware of the known boundaries we would use the first design, but if we include them we would use either the second or the third design. We observe that there is a substantial drop in RMSE when known boundaries are incorporated into the construction of the emulator, as expected. For example, when using the standard iterative V-optimal design the RMSE drops from 0.8166 to 0.5815 when known boundaries are included. We also see a further drop in RMSE when the existence of the known boundaries is used in the design process, for example, the RMSE drops from 0.5815 (iterative V-optimal) to 0.5091 (warped iterative V-optimal) and to a similar 0.5101 for the full iterative V-optimal with known boundaries design. We note that the second and third designs give similar RMSEs in the bold cases, up to the noise resulting from the finite size of the 2000 diagnostic runs (Table 5 gives the calculated V-optimality criteria $c(X_D)$ for each of the cases in Table 4 and shows that this criterion is very similar for the second and third designs in these cases). Comparing Tables 3 and 4 we can see that the approximate V-optimal designs have lower RMSEs than their Latin hypercube counterparts, which is mainly due to their better space filling properties, justifying their use, provided we are not too concerned about their projection properties, as discussed in section 4. This improvement is less for the larger value of $\theta = 1.2$.

The results of this design simulation study suggest that knowledge of known boundaries should affect our choice of training point design, which can lead to substantial benefits in addition to those obtained by the direct incorporation of the boundaries into the emulator.

6. Conclusion.

We have discussed how improved emulation strategies have the potential to benefit multiple scientific areas, allowing more accurate analyzes with lower computational

Table 5

A table of the V-optimality criterion values $c(X_D)$ of the 6^6 grid of points, using emulators constructed with and without the known boundaries for three designs, namely, a standard iterative V-optimal design without the known boundaries, the warped version of this design, and an iterative V-optimal design which takes account for the known boundaries.

θ	Known boundaries	Iterative V-optimal	Warped iterative V-optimal	Iterative V-optimal with known boundaries
0.7	Without	**55605**	56018	56464
	With	28673	**27707**	**27675**
1.2	Without	**33668**	36839	40015
	With	7993	**6627**	**6607**

cost, and therefore if additional prior insight into the physical structure of the model is available, it is of real importance that emulator structures capable of incorporating such insights are indeed developed.

Here it was shown that if a simulator has boundaries or hyperplanes in its input space where it can be either analytically solved or just evaluated far more efficiently, these known boundaries can be formally incorporated into the emulation process by Bayesian updating of the emulators with respect to the information contained on the boundaries. It was also shown that this is possible for a large class of emulators, for multiple boundaries of various forms[1] and, most importantly, for trivial extra computational cost. This analysis also demonstrated how to include known boundaries when using standard black box Gaussian process software (for users who do not have access to alter the code), by simply incorporating all the projections of the input points of interest and the simulator runs into the emulator update. This method is simple to implement but is of course less powerful than direct implementation of the fully updated emulator equations that we have developed here, especially if one needs to evaluate the emulator at a large number of points.

The design problem of how to choose an efficient set of runs of the full simulator, given that we are aware of the existence of one or more known boundaries, was then examined. V-optimal and warped Latin hypercube designs were suggested as reasonable choices in this context, and their relative strengths and weaknesses were explored. Finally, we applied this approach to a model of hormonal crosstalk in *Arabidopsis*, an important model in systems biology, which possesses two perpendicular known boundaries, and analyzed the improvements to the emulator of the $[PLSp]$ output, first due to the known boundaries, and then due to the use of more careful designs of simulator runs.

Obviously, the applicability of this approach depends on whether any such boundaries can be found for the complex model in question. We note that in some scenarios the input space of interest $\mathcal{X} \subset \mathbb{R}^d$ may be defined such that $\mathcal{X}$ does not contain a known boundary $\mathcal{K}$; however, a boundary may exist just outside of $\mathcal{X}$ (for example, when some physical parameter was set to zero, but the lower limit of $\mathcal{X}$ for that parameter is just above zero), such that were $\mathcal{K}$ to

[1]We note that although for a general product correlation structure we require the boundary to be a hyperplane perpendicular to one (or more for lower dimensional boundaries) input direction, for the Gaussian correlation structure the boundary can be a hyperplane in any orientation. Our analysis also naturally extends to lower dimensional boundaries.

be included in the emulation process, the resulting emulator would still be improved over a significant proportion of $\mathcal{X}$, with the extent of the improvement dependent on the correlation parameters and on the distance from $\mathcal{K}$ to $\mathcal{X}$. This may in fact be fairly common as when specifying $\mathcal{X}$ the domain expert may be aware that the boundary $\mathcal{K}$ is not of primary physical interest, as the more complex model that is employed away from $\mathcal{K}$ has been constructed for a reason, and hence they may not have originally included $\mathcal{K}$ within $\mathcal{X}$. As the benefits of using $\mathcal{K}$ in the emulation process come with trivial computational cost, all such boundaries should be included.

A further point we would make is that, as was seen in the application to the *Arabidopsis* model, known boundaries may exist only for a subset of the simulator outputs. This can still be useful, especially in applications of emulation such as history matching [41, 42] whereby the input parameter space $\mathcal{X}$ is searched to find acceptable matches between model and observed data, by iteratively discarding regions of $\mathcal{X}$ that seem unlikely to lead to good matches based only on subsets of the outputs that are easy to emulate. Further outputs are included as the history match progresses and are usually found to be easier to emulate in later iterations, after $\mathcal{X}$ has been substantially reduced (see [41, 42, 24] for extended discussions and examples of this). Therefore the iterative structure of the history matching approach may still allow substantial exploitation of known boundaries that improve the emulation of only a subset of the outputs.

The results presented here can of course be extended in several directions, for example, to collections of multiple parallel or perpendicular boundaries of varying dimension, or to multivariate emulators providing suitable product correlation structures are used [33]. Extensions to the case of uncertain regression parameters (the β_j in (2.7)) are also possible, although the formal update would now depend on the specific form of the correlation function $r_1(a)$ which may not be tractable for many choices. Curved boundaries of various geometries could of course be incorporated, provided both that suitable transformations were found to convert them to hyperplanes and that we were happy to adopt the induced transformed product correlation structure as our prior beliefs. We leave these considerations to future work.

Appendix A. Two perpendicular boundary emulator derivations.

Here we provide the full derivation of the expectation and covariance of $f(x)$, adjusted by perpendicular boundaries L and K as discussed in section 3.4.

We perform the update by K using the results of section 3.1. We then use a proof analogous to that of (3.4) but now applied to the vector L after performing the update[2] for K:

$$\begin{aligned} &\mathrm{Var}_K[L]\,\mathrm{Var}_K[L]^{-1} = I_{(m+1)} \\ \Rightarrow\quad &\mathrm{Cov}_K\left[f(x^L), L\right]\mathrm{Var}_K[L]^{-1} = (1, 0, \ldots, 0). \end{aligned} \tag{A.1}$$

We can also use (3.7), which is a direct consequence of the product correlation structure, which still holds after the update by K, now with K replaced by L to give

$$\mathrm{Cov}_K[f(x), L] = r_2(b)\,\mathrm{Cov}_K\left[f(x^L), L\right], \tag{A.2}$$

[2]We are assuming there are no problems here due to the nonempty $\mathcal{K} \cap \mathcal{L}$. In fact the full Bayes linear update would instead use the generalized inverse if L contains points on $\mathcal{K}$ (which would possess zero variance), but (A.1) will remain the same.

where b is the perpendicular distance from x to $\mathcal{L}$, as shown in Figure 3 (left panel). The expectation as given by (3.25) can now be calculated using the sequential update equation (3.18) as

$$\begin{aligned}
\mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}_K\left[f(x)\right] + \mathrm{Cov}_K\left[f(x),L\right]\mathrm{Var}_K\left[L\right]^{-1}\left(L-\mathrm{E}_K\left[L\right]\right)\\
&= \mathrm{E}_K\left[f(x)\right] + r_2(b)(1,0,\ldots,0)(L-\mathrm{E}_K\left[L\right])\\
&= \mathrm{E}_K\left[f(x)\right] + r_2(b)(f(x^L)-\mathrm{E}_K\left[f(x^L)\right]) \qquad \text{(A.3)}\\
&= \mathrm{E}\left[f(x)\right] + r_1(a)(f(x^K)-\mathrm{E}\left[f(x^K)\right]) + r_2(b)f(x^L)\\
&\quad - r_2(b)(\mathrm{E}\left[f(x^L)\right] + r_1(a)(f(x^{LK})-\mathrm{E}\left[f(x^{LK})\right]))\\
&= \mathrm{E}\left[f(x)\right] + r_1(a)(f(x^K)-\mathrm{E}\left[f(x^K)\right]) + r_2(b)(f(x^L)-\mathrm{E}\left[f(x^L)\right])\\
&\quad - r_1(a)r_2(b)(f(x^{LK})-\mathrm{E}\left[f(x^{LK})\right])\\
&= \mathrm{E}\left[f(x)\right] + r_1(a)\Delta f(x^K) + r_2(b)\Delta f(x^L) - r_1(a)r_2(b)\Delta f(x^{LK}), \qquad \text{(A.4)}
\end{aligned}$$

where we have also used (3.8) for $\mathrm{E}_K[f(x)]$, defined $\Delta f(.) \equiv f(.) - \mathrm{E}\left[f(.)\right]$, and denoted the projection of x^L onto $\mathcal{K}$ as x^{LK}, which is just the perpendicular projection of x onto $\mathcal{L}\cap\mathcal{K}$. The corresponding expression for the covariance as shown in (3.27), adjusted by L and K, is

$$\begin{aligned}
\mathrm{Cov}_{L\cup K}\left[f(x),f(x')\right] &= \mathrm{Cov}_K\left[f(x),f(x')\right] - \mathrm{Cov}_K\left[f(x),L\right]\mathrm{Var}_K\left[L\right]^{-1}\mathrm{Cov}_K\left[L,f(x')\right]\\
&= \mathrm{Cov}_K\left[f(x),f(x')\right] - r_2(b)(1,0,\ldots,0)\mathrm{Cov}_K\left[L,f(x')\right]\\
&= \mathrm{Cov}_K\left[f(x),f(x')\right] - r_2(b)\mathrm{Cov}_K\left[f(x^L),f(x')\right]\\
&= r_2(b-b')\mathrm{Cov}_K\left[f(x^L),f(x'^L)\right] - r_2(b)\mathrm{Cov}_K\left[f(x^L),f(x'^L)\right]r_2(b') \qquad \text{(A.5)}\\
&= (r_2(b-b')-r_2(b)r_2(b'))\mathrm{Cov}_K\left[f(x^L),f(x'^L)\right]\\
&= \sigma^2 R_1(a,a')\,R_2(b,b')\,r_{-1,-2}(x^{LK}-x'^{LK}),
\end{aligned}$$

where we have defined the correlation function of the projection of x and x' onto $\mathcal{L}\cap\mathcal{K}$ as

$$r_{-1,-2}(x^{LK}-x'^{LK}) = \prod_{i=3}^{d} r_i(x_i^{LK}-x_i'^{LK}) = \mathrm{Cov}\left[f(x^{LK}),f(x'^{LK})\right].$$

The limiting behavior looks to be as expected in the large and small a, a', b, b' limits, e.g.,

$$\begin{aligned}
&\lim_{b\to 0}\mathrm{E}_{L\cup K}\left[f(x)\right] = f(x^L), &&\lim_{b\to 0}\mathrm{Var}_{L\cup K}\left[f(x)\right] = 0,\\
&\lim_{b\to\infty}\mathrm{E}_{L\cup K}\left[f(x)\right] = \mathrm{E}_K\left[f(x)\right], &&\lim_{b\to\infty}\mathrm{Var}_{L\cup K}\left[f(x)\right] = \mathrm{Var}_K\left[f(x)\right],
\end{aligned}$$

and similarly for the covariances

$$\begin{aligned}
\lim_{b\to 0}\mathrm{Cov}_{L\cup K}\left[f(x),f(x')\right] &= \lim_{b'\to 0}\mathrm{Cov}_{L\cup K}\left[f(x),f(x')\right] = 0,\\
\lim_{b,b'\to\infty}\mathrm{Cov}_{L\cup K}\left[f(x),f(x')\right] &= \mathrm{Cov}_K\left[f(x),f(x')\right], \quad b-b' \text{ finite.}
\end{aligned}$$

Appendix B. Two parallel boundary emulator derivations. Here we now provide the full derivation of the expectation and covariance of $f(x)$, adjusted by parallel boundaries L and K, as discussed in section 3.5.

First we need to find the analogous version of (3.23), which relates $\mathrm{Cov}_K[f(x), L]$ to $\mathrm{Cov}_K[f(x^L), L]$. Noting that

$$\mathrm{Cov}_K\left[f(x^L), f(z^{(j)})\right] = \sigma^2 R_1(c,c)\, r_{-1}(x^K - y^{(j)}) \tag{B.1}$$

it follows that

$$\begin{aligned}\mathrm{Cov}_K\left[f(x), f(z^{(j)})\right] &= \sigma^2 R_1(a,c)\, r_{-1}(x^K - y^{(j)})\\ &= \frac{R_1(a,c)}{R_1(c,c)}\,\sigma^2 R_1(c,c)\, r_{-1}(x^K - y^{(j)})\\ &= \frac{R_1(a,c)}{R_1(c,c)}\,\mathrm{Cov}_K\left[f(x^L), f(z^{(j)})\right]\\ &= R_1^{(2)}(a,c)\,\mathrm{Cov}_K\left[f(x^L), f(z^{(j)})\right], \qquad \text{(B.2)}\end{aligned}$$

where we have defined $R_1^{(2)}(a,c) = R_1(a,c)/R_1(c,c)$. Therefore we have

$$\mathrm{Cov}_K\left[f(x), L\right] = R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), L\right]. \tag{B.3}$$

Here (3.22) holds as before, implying we can again avoid explicit evaluation of the intractable $\mathrm{Var}_K[L]^{-1}$ term. Hence the adjusted expectation, as given by (3.31), can be calculated using the sequential update equation (3.18) as

$$\begin{aligned}\mathrm{E}_{L\cup K}\left[f(x)\right] &= \mathrm{E}_K\left[f(x)\right] + \mathrm{Cov}_K\left[f(x), L\right]\mathrm{Var}_K\left[L\right]^{-1}\left(L - \mathrm{E}_K\left[L\right]\right)\\ &= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), L\right]\mathrm{Var}_K\left[L\right]^{-1}\left(L - \mathrm{E}_K\left[L\right]\right)\\ &= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)(1,0,\ldots,0)(L - \mathrm{E}_K\left[L\right])\\ &= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)(f(x^L) - \mathrm{E}_K\left[f(x^L)\right]) \qquad \text{(B.4)}\\ &= \mathrm{E}\left[f(x)\right] + r_1(a)(f(x^K) - \mathrm{E}\left[f(x^K)\right])\\ &\quad + \frac{r_1(b) - r_1(a)r_1(c)}{1 - r_1^2(c)}\left\{f(x^L) - \left(\mathrm{E}\left[f(x^L)\right] + r_1(c)(f(x^K) - \mathrm{E}\left[f(x^K)\right])\right)\right\}\\ &= \mathrm{E}\left[f(x)\right] + \left[\frac{r_1(a) - r_1(b)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^K) + \left[\frac{r_1(b) - r_1(a)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^L), \qquad \text{(B.5)}\end{aligned}$$

where we have used the fact that for parallel boundaries the projection of x^L onto $\mathcal{K}$ is just

x^K. Similarly we find the covariance adjusted by L and K to be

$$\begin{aligned}
\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= \mathrm{Cov}_K\left[f(x), f(x')\right] - \mathrm{Cov}_K\left[f(x), L\right]\mathrm{Var}_K\left[L\right]^{-1}\mathrm{Cov}_K\left[L, f(x')\right] \\
&= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)(1,0,\dots,0)\mathrm{Cov}_K\left[L, f(x')\right] \\
&= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), f(x')\right] \\
&= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right]R_1^{(2)}(c,a') \\
&= \sigma^2 R_1(a,a')\, r_{-1}(x^K - x'^K) \\
&\quad - R_1^{(2)}(a,c)\sigma^2 R_1(c,c)\, r_{-1}(x^K - x'^K) R_1^{(2)}(c,a') \\
&= \sigma^2\, r_{-1}(x^K - x'^K)\left\{R_1(a,a') - R_1^{(2)}(a,c)R_1(c,c)R_1^{(2)}(c,a')\right\} \\
&= \sigma^2\, r_{-1}(x^K - x'^K)\left\{R_1(a,a') - \frac{R_1(a,c)R_1(c,a')}{R_1(c,c)}\right\},
\end{aligned} \tag{B.6}$$

which is just a generalized form of (3.11). To make the invariance under interchange of boundaries explicit, we expand out the $R_1(.,.)$ terms giving

$$\begin{aligned}
\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right] &= \sigma^2\,\frac{r_{-1}(x^K - x'^K)}{R_1(c,c)}\left\{(r_1(a-a') - r_1(a)r_1(a'))(1 - r_1^2(c))\right. \\
&\qquad \left. - (r_1(b) - r_1(a)r_1(c))(r_1(b') - r_1(c)r_1(a'))\right\}, \\
&= \sigma^2\,\frac{r_{-1}(x^K - x'^K)}{1 - r_1^2(c)}\left\{r_1(a-a')(1 - r_1^2(c)) - r_1(a)r_1(a') - r_1(b)r_1(b')\right. \\
&\qquad \left. + r_1(c)\left[r_1(a)r_1(b') + r_1(b)r_1(a')\right]\right\},
\end{aligned} \tag{B.7}$$

which is now explicitly invariant under the interchange of the two boundaries $\mathcal{K} \leftrightarrow \mathcal{L}$ (as $c = a + b$ is invariant under $a \leftrightarrow b, a' \leftrightarrow b'$, as is $a - a' = b - b'$).

The emulator variance is given by (3.33), but we note that it is now bounded from above and will attain its maximum at the midpoint between the two parallel boundaries. Setting $a = b = c/2$ in (3.33) shows that this maximum bound is given by

$$\mathrm{Var}_{L\cup K}\left[f(x)\right] \le \sigma^2\left\{1 - \frac{2r_1^2(\frac{c}{2})}{1 + r_1(c)}\right\}. \tag{B.8}$$

In addition, we note that the emulator expectation, covariance, and variance update calculations for a single boundary will easily generalize to a boundary of lower dimension than $d-1$. However, for multiple boundaries there will be some additional constraints, specifically on the dimension and intersection of the boundaries in the perpendicular case (the parallel case is easier), a full discussion of which we leave to future work.

Appendix C. Continuous known boundary proofs.

C.1. Two perpendicular continuous boundaries. For the two perpendicular boundaries continuous case, after the update by boundary $\mathcal{K}$, we use $s_K(z, z')$ to represent the infinite dimensional generalization of $\mathrm{Var}_K[L]^{-1}$, which satisfies the corresponding inverse property:

$$\int_{z'\in\mathcal{L}} \mathrm{Cov}_K\left[f(z), f(z')\right] s_K(z', z'')\, dz' = \delta(z - z'') \qquad \text{for } z, z'' \in \mathcal{L}. \tag{C.1}$$

Then, noting that $\mathrm{Cov}_K[f(x), f(z)] = r_2(b)\,\mathrm{Cov}_K[f(x^L), f(z)]$, the emulator expectation adjusted sequentially by first K and then L becomes

$$\begin{aligned}
&\mathrm{E}_{L\cup K}\left[f(x)\right]\\
&\quad= \mathrm{E}_K\left[f(x)\right] + \int_{z\in\mathcal{L}}\int_{z'\in\mathcal{L}} \mathrm{Cov}_K\left[f(x), f(z)\right]\ s_K(z,z')\ (f(z') - \mathrm{E}_K\left[f(z')\right])dzdz'\\
&\quad= \mathrm{E}_K\left[f(x)\right] + \int_{z\in\mathcal{L}}\int_{z'\in\mathcal{L}} r_2(b)\mathrm{Cov}_K\left[f(x^L), f(z)\right]\ s_K(z,z')\ (f(z') - \mathrm{E}_K\left[f(z')\right])dzdz'\\
&\quad= \mathrm{E}_K\left[f(x)\right] + r_2(b)\int_{z'\in\mathcal{L}} \delta(x^L - z')\ (f(z') - \mathrm{E}_K\left[f(z')\right])dz'\\
&\quad= \mathrm{E}_K\left[f(x)\right] + r_2(b)\ (f(x^L) - \mathrm{E}_K\left[f(x^L)\right]),
\end{aligned}$$

which is identical to (A.3), and the rest of the proof of (A.4) follows as before. Similarly for the covariance we have

$$\begin{aligned}
&\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right]\\
&\quad= \mathrm{Cov}_K\left[f(x), f(x')\right] - r_2(b)\int_{z'\in\mathcal{L}} \delta(x^L - z')\ \mathrm{Cov}_K\left[f(z'), f(x'^L)\right] r_2(b')\ dz'\\
&\quad= r_2(b-b')\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right] - r_2(b)\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right] r_2(b'),
\end{aligned}$$

which agrees with (A.5), and the rest of the proof follows as before.

C.2. Two parallel continuous boundaries. The proof for continuous parallel boundaries follows a form similar to the perpendicular case. We use $s_K(z,z')$ as before, which still satisfies (C.1). However, here we have instead from (B.2) that

$$\mathrm{Cov}_K\left[f(x), f(z)\right] = R_1^{(2)}(a,c)\,\mathrm{Cov}_K\left[f(x^L), f(z)\right], \qquad z\in\mathcal{L}.$$

Therefore the emulator expectation adjusted sequentially by first K and then L becomes

$$\begin{aligned}
&\mathrm{E}_{L\cup K}\left[f(x)\right]\\
&\quad= \mathrm{E}_K\left[f(x)\right] + \int_{z\in\mathcal{L}}\int_{z'\in\mathcal{L}} R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), f(z)\right]\ s_K(z,z')\ (f(z') - \mathrm{E}_K\left[f(z')\right])dzdz'\\
&\quad= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)\int_{z'\in\mathcal{L}} \delta(x^L - z')\ (f(z') - \mathrm{E}_K\left[f(z')\right])dz'\\
&\quad= \mathrm{E}_K\left[f(x)\right] + R_1^{(2)}(a,c)\ (f(x^L) - \mathrm{E}_K\left[f(x^L)\right]),
\end{aligned}$$

which is identical to (B.4), and the rest of the proof of (B.5) follows as before. Similarly for the covariance we have

$$\begin{aligned}
&\mathrm{Cov}_{L\cup K}\left[f(x), f(x')\right]\\
&\quad= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)\int_{z'\in\mathcal{L}} \delta(x^L - z')\ \mathrm{Cov}_K\left[f(z'), f(x'^L)\right] R_1^{(2)}(c,a')\ dz'\\
&\quad= \mathrm{Cov}_K\left[f(x), f(x')\right] - R_1^{(2)}(a,c)\mathrm{Cov}_K\left[f(x^L), f(x'^L)\right] R_1^{(2)}(c,a'),
\end{aligned}$$

which agrees with (B.6), and the rest of the proof of (B.7) again follows as before.

Table 6

A table of parameter ranges for Arabidopsis thaliana *(which were square rooted and converted to* $[-1, 1]$ *for the analysis).*

Input rate parameter	Minimum	Maximum
k_4	0	10
k_6	0	1
k_{6a}	0	20
k_7	0	10
k_8	0	10
k_9	0	1

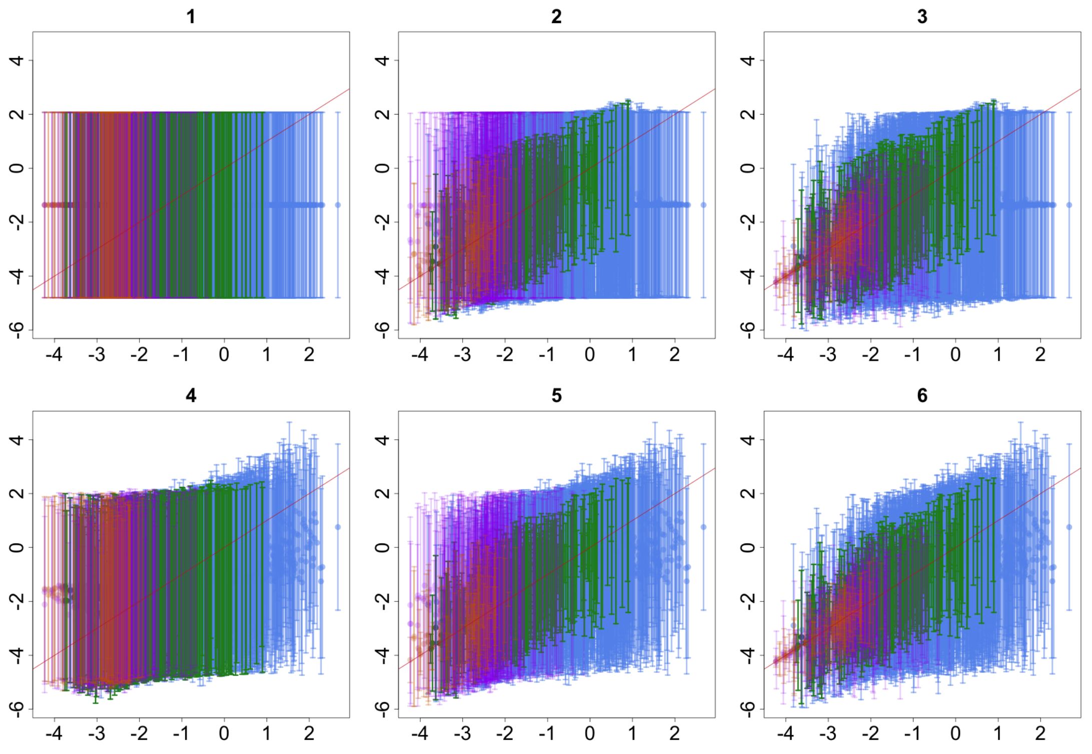


Figure 11. *$f(x)$ against $\mathrm{E}_D[f(x)] \pm 3\sqrt{\mathrm{Var}_D[f(x)]}$ for the diagnostic set of 2000 points. Blue points are such that $k_6 > -0.5$ and $k_8 > -0.5$, green points are such that $k_6 < -0.5$ and $k_8 > -0.5$, purple points are such that $k_6 > -0.5$ and $k_8 < -0.5$, and orange points are such that $k_6 < -0.5$ and $k_8 < -0.5$. The red line is the function $y = x$. The columns (from left to right) show the results of emulating without boundaries, with one boundary and with two boundaries. The rows show the results of emulating without training points on the top and with training points on the bottom.*

Appendix D. *Arabidopsis* model details and emulator diagnostics. Table 6 gives the parameter ranges for the *Arabidopsis thaliana* model, used throughout section 5.

Figure 11 shows $f(x)$ against $\mathrm{E}_D[f(x)] \pm 3\sqrt{\mathrm{Var}_D[f(x)]}$ for each of the six emulator scenarios for the diagnostic set of 2000 points, with the color scheme the same as for Figure 10.

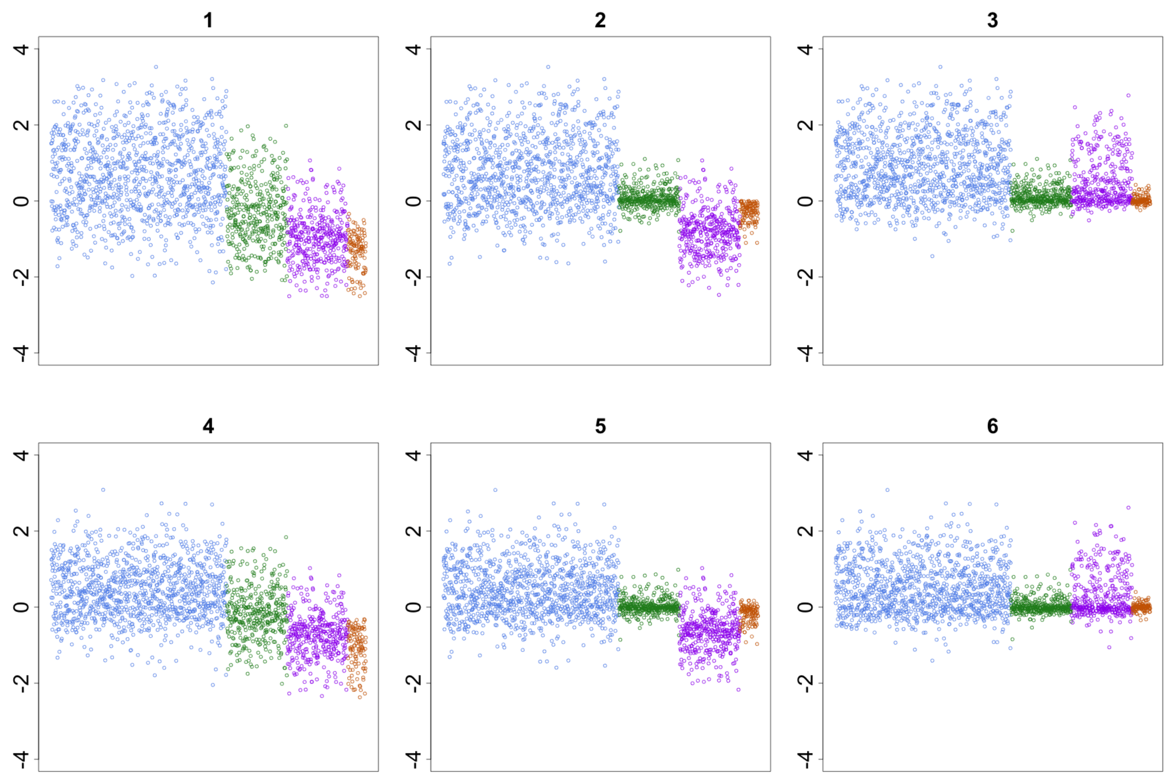


Figure 12. $\frac{(f(x)-\mathrm{E}_D[f(x)])}{\sqrt{\mathrm{Var}_D[f(x)]}}$ *for the diagnostic set of* 2000 *points. The columns (from left to right) show the results of emulating without boundaries, with one boundary and with two boundaries. The top row shows the results of emulating without training points and the bottom row shows the results of emulating with the training points.*

These plots show how the variance at each point is updated in correspondence to its expectation. We observe that the error bars on some of the points decrease as the boundaries get utilized, particularly for points which lie close to at least one or the other of the boundaries. The majority of the error bars still cross the line $y = x$, indicating that the emulator expectation lies within three emulator standard deviations of the true simulator value.

Figure 12 shows $\frac{(f(x)-\mathrm{E}_D[f(x)])}{\sqrt{\mathrm{Var}_D[f(x)]}}$ for each of the six emulator scenarios for the diagnostic set of 2000 points. A value with magnitude greater than 3 is equivalent to the corresponding error bar in Figure 11 not containing the true simulator value. We conclude that the diagostic plots are acceptable for all emulators, with those points far from the boundary $k_6 = 0$ having larger values once the boundaries have been utilized. The small diagnostic values corresponding to points close to the boundary $k_6 = 0$ may suggest that a larger correlation length could be appropriate, particularly in certain dimensions of the input space such as k_6 itself.

REFERENCES

[1] I. Andrianakis, N. McCreesh, I. Vernon, T. McKinley, J. Oakley, R. Nsubuga, M. Goldstein, and R. White, *Efficient history matching of a high dimensional individual based HIV transmission model*, SIAM/ASA J. Uncertain. Quantif., 5 (2017), pp. 694–719.

[2] I. Andrianakis, I. Vernon, N. McCreesh, T. McKinley, J. Oakley, R. Nsubuga, M. Goldstein, and R. White, *Bayesian history matching of complex infectious dizease models using emulation: A tutorial and a case study on HIV in Uganda*, PLoS Comput Biol., 11 (2015), e1003968.

[3] I. Andrianakis, I. Vernon, N. McCreesh, T. J. McKinley, J. E. Oakley, R. N. Nsubuga, M. Goldstein, and R. G. White, *History matching of a complex epidemiological model of human immunodeficiency virus transmission by using variance emulation*, J. R. Stat. Soc. Ser. C Appl. Stat., 66 (2017), pp. 717–740, https://doi.org/10.1111/rssc.12198.

[4] T. S. Bastos and A. O'Hagan, *Diagnostics for Gaussian process emulators*, Technometrics, 51 (2008), pp. 425–438.

[5] M. J. Bayarri, J. O. Berger, E. S. Calder, K. Dalbey, S. Lunagomez, A. K. Patra, E. B. Pitman, E. T. Spiller, and R. L. Wolpert, *Using statistical and computer models to quantify volcanic hazards*, Technometrics, 51 (2009), pp. 402–413, https://doi.org/10.1198/TECH.2009.08018.

[6] J. Bernardo and A. Smith, *Bayesian Theory*, Wiley Ser. Probab. Stat., Wiley, New York, 2006, https://books.google.co.uk/books?id=cl6nAAAACAAJ.

[7] A. Boukouvalas, P. Sykes, D. Cornford, and H. Maruri-Aguilar, *Bayesian precalibration of a large stochastic microsimulation model*, IEEE Trans. Intell. Transportation Syst., 15 (2014).

[8] R. G. Bower, I. Vernon, M. Goldstein, A. J. Benson, C. G. Lacey, C. M. Baugh, S. Cole, and C. S. Frenk, *The parameter space of galaxy formation*, Mon. Not. Roy. Astron. Soc., 96 (2010), pp. 717–729.

[9] P. S. Craig, M. Goldstein, A. H. Seheult, and J. A. Smith, *Bayes linear strategies for history matching of hydrocarbon reservoirs*, in Bayesian Statistics 5, J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, eds., Clarendon Press, Oxford, UK, 1996, pp. 69–95.

[10] P. S. Craig, M. Goldstein, A. H. Seheult, and J. A. Smith, *Pressure matching for hydrocarbon reservoirs: A case study in the use of Bayes linear strategies for large computer experiments (with discussion)*, in Case Studies in Bayesian Statistics, C. Gatsonis, J. S. Hodges, R. E. Kass, R. McCulloch, P. Rossi, and N. D. Singpurwalla, eds., Vol. 3, Springer, New York, 1997, pp. 36–93.

[11] J. A. Cumming and M. Goldstein, *Bayes linear uncertainty analysis for oil reservoirs based on multiscale computer experiments*, in Handbook of Bayesian Analysis, A. O'Hagan and M. West, eds., Oxford University Press, Oxford, UK, 2009.

[12] J. A. Cumming and M. Goldstein, *Small sample bayesian designs for complex high-dimensional models based on information gained using fast approximations*, Technometrics, 51 (2009), pp. 377–388.

[13] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker, *Bayesian prediction of deterministic functions with applications to the design and analysis of computer experiments*, J. Amer. Statist. Assoc., 86 (1991), pp. 953–963.

[14] B. De Finetti, *Theory of Probability*, Vol. 1, Wiley, New York, 1974.

[15] M. Goldstein, *Bayes linear analysis*, in Encyclopaedia of Statistical Sciences, S. Kotz, C. B. Reed, N. Balakrishnan, B. Vidakovic, and N. L. Johnson, eds., Wiley, New York, 1999, pp. 29–34.

[16] M. Goldstein, A. Seheult, and I. Vernon, *Accessing model accuracy*, in Environmental Modelling: Finding Simplicity in Complexity, J. Wainwright and M. Mulligan, eds., 2nd ed., Wiley, New York, 2013, https://doi.org/10.1002/9781118351475.ch26.

[17] M. Goldstein and D. A. Wooff, *Bayes Linear Statistics: Theory and Methods*, Wiley, New York, 2007.

[18] *GPy: A Gaussian Process Framework in Python*, http://github.com/SheffieldML/GPy (2012).

[19] R. K. S. Hankin, *Introducing BACCO, an R bundle for Bayesian analysis of computer code output*, J. Statist. Softw., 14 (2005).

[20] K. Heitmann, D. Higdon, M. White, S. Habib, B. J. Williams, E. Lawrence, and C. Wagner, *The coyote universe* II: *Cosmological models and precision emulation of the nonlinear matter power spectrum*, Astrophys. J., 705 (2009), pp. 156–174.

[21] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, *Computer model calibration using high-dimensional output*, J. Amer. Statist. Assoc., 103 (2008), pp. 570–583.
[22] P. B. Holden, N. R. Edwards, J. Hensman, and R. D. Wilkinson, *ABC for climate: Dealing with expensive simulators*, in Handbook of Approximate Bayesian Computation, Chapman & Hall/CRC Press, Boca Raton, FL, 2018.
[23] *Analysis of the genome sequence of the flowering plant* Arabidopsis thaliana, Nature, 408 (2000), pp. 796–815, https://doi.org/10.1038/35048692.
[24] S. E. Jackson, I. Vernon, and J. Liu, *Bayesian uncertainty analysis establishes the link between the parameter space of a complex model of hormonal crosstalk in* Arabidopsis *root development and experimental measurements*, Stat. Appl. Genet. Mol. Biol., arXiv:1801.01538v1 [stat.AP], submitted.
[25] J. S. Johnson, Z. Cui, L. A. Lee, J. P. Gosling, A. M. Blyth, and K. S. Carslaw, *Evaluating uncertainty in convective cloud microphysics using statistical emulation*, J. Adv. Model. Earth Sys., 7 (2015), pp. 162–187.
[26] M. E. Johnson, L. M. Moore, and D. Ylvisaker, *Minimax and maximin distance designs*, J. Statist. Plann. Inference, 26 (1990), pp. 131–148.
[27] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, J. R. Stat. Soc. Ser. B Stat. Methodol., 63 (2001), pp. 425–464.
[28] J. Liu, S. Mehdi, J. Topping, J. Friml, and K. Lindsey, *Interaction of PLS and PIN and hormonal crosstalk in* Arabidopsis *root development*, Front. Plant Science, 4 (2013), https://doi.org/10.3389/fpls.2013.00075.
[29] B. MacDonald, P. Ranjan, and H. Chipman, *GPfit: An R package for fitting a Gaussian process model to deterministic simulator outputs*, J. Statist. Softw., 64 (2015), https://doi.org/10.18637/jss.v064.i12.
[30] N. McCreesh, I. Andrianakis, R. N. Nsubuga, M. Strong, I. Vernon, T. J. McKinley, J. E. Oakley, M. Goldstein, R. Hayes, and R. G. White, *Universal test, treat, and keep: Improving ART retention is key in cost-effective HIV control in Uganda*, BMC Infectious Dizeases, 17 (2017), https://doi.org/10.1186/s12879-017-2420-y.
[31] T. McKinley, I. Vernon, I. Andrianakis, N. McCreesh, J. Oakley, R. Nsubuga, M. Goldstein, and R. White, *Approximate Bayesian computation and simulation-based inference for complex stochastic epidemic models*, Statist. Sci., 33 (2018), pp. 4–18.
[32] L. F. S. Rodrigues, I. Vernon, and R. G. Bower, *Constraints to galaxy formation models using the galaxy stellar mass function*, Mon. Not. Roy. Astron. Soc., 466 (2017), pp. 2418–2435.
[33] J. Rougier, *Efficient emulators for multivariate deterministic functions*, J. Comput. Graph. Statist., 17 (2008), pp. 827–843, https://doi.org/10.1198/106186008X384032.
[34] J. Rougier, *A Representation Theorem for Stochastic Processes with Separable Covariance Functions, and its Implications for Emulation*, arXiv:1702.05599 [math.ST], 2012.
[35] J. Rougier, S. Guillas, A. Maute, and A. D. Richmond, *Expert knowledge and multivariate emulation: The thermosphere–ionosphere electrodynamics general circulation model (tie-gcm)*, Technometrics, 51 (2009), pp. 414–424, https://doi.org/10.1198/TECH.2009.07123.
[36] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–435.
[37] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer, New York, 2003.
[38] M. D. Schneider, L. Knox, S. Habib, K. Heitmann, D. Higdon, and C. Nakhleh, *Simulations and cosmological inference: A statistical model for power spectra means and covariances*, Phys. Rev. D, 78 (2008), 063529, https://doi.org/10.1103/PhysRevD.78.063529.
[39] M. H. Tan, *Gaussian process modeling of a functional output with information from boundary and initial conditions and analytical approximations*, Technometrics, 60 (2018).
[40] M. H. Tan, *Gaussian process modeling with boundary information*, Statist. Sinica, 28 (2018), pp. 621–648.
[41] I. Vernon, M. Goldstein, and R. G. Bower, *Galaxy formation: A Bayesian uncertainty analysis*, Bayesian Anal., 5 (2010), pp. 619–670.
[42] I. Vernon, M. Goldstein, and R. G. Bower, *Rejoinder for Galaxy formation: A Bayesian uncertainty analysis*, Bayesian Anal., 5 (2010), pp. 697–708.
[43] I. Vernon, M. Goldstein, and R. G. Bower, *Galaxy formation: Bayesian history matching for the observable universe*, Statist. Sci., 29 (2014), pp. 81–90.

[44] I. Vernon and J. P. Gosling, *A Bayesian Computer Model Analysis of Robust Bayesian Analyses*, arXiv:1703.01234 [stat.ME], 2017.

[45] I. Vernon, J. Liu, M. Goldstein, J. Rowe, J. Topping, and K. Lindsey, *Bayesian uncertainty analysis for complex systems biology models: Emulation, global parameter searches and evaluation of gene functions*, BMC Systems Biol., 12 (2018).

[46] D. Williamson, M. Goldstein, L. Allison, A. Blaker, P. Challenor, L. Jackson, and K. Yamazaki, *History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble*, Climate Dynamics, 41 (2013), pp. 1703–1729.