COMPUTER EMULATION WITH NON-STATIONARY GAUSSIAN PROCESSES

SILVIA MONTAGNA AND SURYA T. TOKDAR

Abstract. Gaussian process (GP) models are widely used to emulate propagation uncertainty in computer experiments. GP emulation sits comfortably within an analytically tractable Bayesian framework. Apart from propagating uncertainty of the input variables, a GP emulator trained on finitely many runs of the experiment also offers error bars for response surface estimates at unseen input values. This helps select future input values where the experiment should be run to minimize the uncertainty in the response surface estimation. However, traditional GP emulators use stationary covariance functions, which perform poorly and lead to sub-optimal selection of future input points when the response surface has sharp local features, such as a jump discontinuity or an isolated tall peak. We propose an easily implemented non-stationary GP emulator, based on two stationary GPs, one nested into the other, and demonstrate its superior ability in handling local features and selecting future input points from the boundaries of such features.

1. Introduction

Large scale computer simulation is widely used in modern scientific research to investigate physical phenomena that are too expensive or impossible to replicate directly [28, 9, 31]. Most simulators depend on a handful of tuning parameters and initial conditions, referred to as the input arguments. Often interest focuses on quantifying how uncertainty in the input arguments propagates through the simulator and produce a distribution function over one or many outputs of interest. In this paper we consider only deterministic simulators which when run on the same input twice will produce identical output values.

Quantifying uncertainty propagation will require several runs of a simulator at different input points to learn the input-output map \( Y = f(x) \) accurately over the entire input space. However, computer simulations are very time-consuming, thus running a simulator over a dense grid of input points could be prohibitively expensive. On the other hand, running a simulator over a sparse design chosen in advance may result in insufficient information in vast parts of the input space. Consequently, there is considerable interest in estimating a slow computer simulator with a fast statistical “emulator” [25, 15, 27]. The emulator is fitted to input-output data \( \{x^t, f^t\} \), where \( f^t = \{f(x_1), \ldots, f(x_t)\} \) is obtained from a few preliminary runs of the simulator on design \( x^t = \{x_1, \ldots, x_t\} \), and the fitted model is then used for prediction of \( f \) at input configurations not included in \( x^t \) [25, 6].

For Bayesian emulation, a common practice is to assign \( f \) a Gaussian process prior [25, 8, 29]. Gaussian process (GP) emulation is appealing due to its mathematical tractability and ability to incorporate a wide range of smoothness assumptions. The conditional
posterior distribution of $f$ at future inputs, given data $\{x^t, f^t\}$ and process hyperparameters, remains a GP distribution. The posterior mean of $f(x)$ gives a statistical estimate or surrogate for the simulator output at a new input $x$, whereas the posterior variance at $f(x)$ quantifies how well the simulator has been learned at and around $x$. The latter is a particularly attractive feature of GP emulation as it provides a model-based assessment of the emulator’s accuracy and could be used to actively learn an optimal sequence of input points on which the simulator needs to be run to minimize the uncertainty in posterior surface estimation.

Research on computer emulation has largely focused on stationary GP models [25, 15]. Stationary GPs regard the similarity between $f(x)$ and $f(x + h)$ as a decaying function in $h$ only, known up to global smoothness and decay parameters. This is a strong prior assumption that is not easily washed away by data and may lead to unrealistic emulation for many physical phenomena. In practice, stationary GP emulators run into difficulties when the shape of $f$ has sharp localized features, e.g. abrupt discontinuities or tall peaks, and lead to poor point predictions and selection of future inputs. A simple example is illustrated in the left panel of Figure 1. Three aspects emerge: (i) the discovery of a tall peak in the middle has a rippling effect and creates large oscillations of the predictive mean curve over a large part of the input space, a phenomenon often called “spline tension” effect in the predictor form; (ii) prediction seems overconfident around the peak, where the error bars are too narrow to capture the high variability around $x = 0$; instead, (iii) prediction intervals are quite large where abrupt changes in the function values are not observed, and $f$ is relatively more well-behaved. A sequential design strategy based on uncertainty quantified by the prediction variance would favor the selection of a new input from the whole $x$ domain, with the only exception of the tall peak. Thus, stationary GPs favor the selection of new points in unexplored regions of the input space (exploration), but tend to neglect regions that are deemed important based on the current estimate of $f$ (exploitation).

Extrinsic diagnostics is often used to assess the adequacy of a GP emulator as surrogate for the simulator [3, 3]. For example, one can examine the leave-one-out cross validated (CV) standardized residuals to quantify the emulator’s uncertainty. Either too large or very small CV standardized residuals (as compared to a $N(0,1)$ or a $t_\nu$) at some validating points indicate that the emulator is poorly estimating the predictive uncertainty. Outliers of this kind denote a local fitting problem, which could be improved upon by adding new points in the vicinity. Thus, CV examines the local behavior of $f$, and flags those sub-regions where the simulator has more variations. Therefore, CV leans toward an exploitation-driven sequential design. Although CV is often combined with a stationary GP to better address sequential design, it is difficult to reconcile the exploration-driven predictive variance of a stationary GP with the exploitation-driven flagging of CV, and any combination is ad-hoc. Also, the model remains misspecified: a stationary model is used for a response which is often intrinsically not so [3].

Several approaches to the problem of how to specifying non-stationary GP models can be found in the literature. In the context of computer emulation, [11] propose the Bayesian treed GP model (TGP), which applies independent stationary GPs to subregions of the input space determined by data-driven recursive partitioning parallel to the coordinate axes. Because of the parallel partitioning, TGP adapts well to surfaces having rectangular local features (“axes-aligned” non-stationarity). However, it may run into difficulties when the
Figure 1. Plot of (true) function $f(x) = \sin(x) + 2 \exp(-30x^2)$, $x \in [-2, 2]$ (dashed line). The black dots represent observed data at 15 equally-spaced values of $x$. Left panel: the solid line is the point predictor of $f$, or conditional mean, obtained from a stationary GP emulator fitted to the data. Shaded areas represent the error bars. Right panel: non-stationary GP via latent input augmentation.

nature of the non-stationarity is more general. Also, TGP’s hard partitioning of the input space prevents borrowing of information across partitions and enforces discontinuity on the estimated response surface. [2] decompose $f$ into the sum of two stationary GPs, the first capturing the smooth global trend and the second modeling local details. Other approaches in the context of GP regression include [26, 29, 20]. This literature makes it clear that the main challenges in non-stationary GP modeling are to keep the number of hyperparameters under control to facilitate efficient learning from limited data while allowing for non-stationary features of various geometric shapes and at the same time not to enforce non-stationarity when not needed.

In this paper we propose a non-stationary GP emulator (Section 2) by equipping a stationary GP with optional non-stationarity. Specifically, non-stationarity is achieved by augmenting the input space with one extra latent input which we infer from the data. The latent input can flag regions of the input space characterized by abrupt changes of the function values and help correct for inadequacies in the fit. In the example above we find the proposed method to give significantly improved performance (Figure 1). Furthermore, several numerical examples (Section 4 and 5) show that our emulator adapts to local features of many kinds of shape and provides a more trustworthy judgement of uncertainty than stationary and other existing non-stationary GP emulators. The latter is a key advantage of our method. When an emulator is used to actively learn an optimal sequence of design points to minimize expensive runs of the simulator, it is absolutely crucial to have trustworthy judgement of uncertainty of the current estimate of $f$ to concentrate efforts only on where needed. Sections 5 and 6 show results from various synthetic and real experiments where a sequential version of our emulator outperforms similar sequential adaptations of existing GP emulators, where we measured performance by number of simulator runs needed to achieve a certain accuracy.
The proposed method is also attractive from an operational point of view. Both the latent input dimension and the response function (of the original plus the latent inputs) are individually modeled as stationary GPs controlled by a small number of hyperparameters that can be efficiently learned with sequential Monte Carlo (MC) computing leveraging on conjugacy properties of GP. Sequential MC computing seamlessly blends with active learning of the sequential design, as opposed to Markov chain sampling based non-stationary GP emulators whose sequential adaptation requires re-running the whole Markov chain sampler at every iteration. We also investigate a two-stage fast approximation of the proposed emulator where the latent input GP is directly learned from data through nonparametric regression and the estimated input surface is plugged in to learn \( f \). Simulations suggest that the two-stage approximation performs at least as well as the sequential MC “full Bayes” counterpart in handling local features and selecting additional inputs from the boundaries of such features. However, the sequential MC version of our emulator often achieves better accuracy given the same number of input points in the design.

2. Gaussian process emulators

2.1. GP emulation and stationarity. The canonical emulator used for the design and analysis of computer experiments is the GP. Specifically, for any finite collection of inputs \( x^t = (x_1, \ldots, x_t), (f(x_1), \ldots, f(x_t))^\top \) is jointly distributed as a multivariate normal distribution with mean \( \mu(x) = h(x)^\top \beta \) and positive definite covariance matrix \( C(x, x') = \sigma^2 K(x, x') \). Although \( h(\cdot) \) may be any function on the input space \( \mathcal{X} \), hereafter we adopt a linear mean in the inputs, \( h(x_i) = [1, x_{i1}, \ldots, x_{ip}]^\top \), with \( \beta \) a vector of unknown parameters. Although the output of a computer model does not generally vary linearly in the inputs, there is often too little prior knowledge on the type of non-linearity.

Clearly, the representation of \( f \) as a Gaussian vector makes the computation conceptually straightforward. The conditional distribution of \( f \) at a new input \( \tilde{x} \), given data \( \{x, f(x)\}_{1:t} \equiv \{X, F\} \) and model parameters \( \theta = \{\beta, \sigma^2, K\} \), is also Gaussian with mean

\[
\hat{f}(\tilde{x}) = \mathbb{E}[f(\tilde{x})|\{x, f(x)\}_{1:t}, \theta] = h(\tilde{x})^\top \beta + k^\top(\tilde{x})K^{-1}(F - X\beta)
\]

and variance

\[
\hat{\sigma}^2(\tilde{x}) = \mathbb{V}[f(\tilde{x})|\{x, f(x)\}_{1:t}, \theta] = \sigma^2 \{K(\tilde{x}, \tilde{x}) - k^\top(\tilde{x})K^{-1}k(\tilde{x})\}
\]

where \( k^\top(\tilde{x}) \) is the \( t \)-vector whose \( i \)-th component is \( K(\tilde{x}, x_i), i = 1, \ldots, t \), and \( K \) is the \( t \times t \) correlation matrix with \( i, j \) element \( K(x_i, x_j) \).

The correlation function is crucial in GP modeling; it is through \( K(x, x') \) that we express a belief about how similar \( f(x) \) and \( f(x') \) should be if \( x \) and \( x' \) were close in \( \mathcal{X} \), thereby we express a belief about the smoothness of \( f \). Although different formulations are possible, in this work we use the separable power correlation function

\[
K(x, x') = e^{-\sum_{l=1}^{p_0} \phi_l (x_l - x'_l)^2}.
\]

We fix \( p_0 = 2 \) (product-Gaussian correlation) and infer the correlation range parameters \( \{\phi_l\}_{i=1}^{p_0} \) as part of our estimation procedure. Thus, the correlation is only function of \( x - x' \) (stationarity) and a set of roughness (unknown) parameters. \[32\] show that the squared-exponential kernel \( \Pi \) can optimally adapt to any smoothness level. \[5\] develop a class of
priors for the correlation range parameters which leads to minimax adaptive rates of posterior concentration.

Bayesian inference for GP emulators proceeds by specifying prior distributions for the model parameters. We use an improper uniform prior on $\beta$, $\beta \propto 1$, to reflect weak prior knowledge about these parameters, an inverse-gamma (IG) prior for the scale, $\sigma^2 \sim \text{IG}(a/2, b/2)$, and a log-normal prior for the correlation parameters, $\phi_l \sim \text{log N}(\mu_\phi, \nu_\phi)$, but other formulations are possible [11]. The (marginalized) distribution of $f$ at a new input $\tilde{x}$, conditioned on $K$ and data observed up to time $t$, is a Student-$t$ distribution with $\hat{\nu} = t - p - 1$ degrees of freedom, mean $\hat{f}(\tilde{x}|\{x, f(x)\}_{1:t}, K)$, and variance $\hat{\sigma}^2(\tilde{x}|\{x, f(x)\}_{1:t}, K)$. See [13] for more details on the derivation of these quantities.

2.2. Non-stationary GP through one latent input. In response to concerns about the adequacy of the stationary assumption, we propose a non-stationary GP that builds upon the concept of spatial deformation as in [26]. Specifically, we write

$$Y = f(x, Z),$$

thus modeling the simulator as function of both the $p$-dimensional (known) vector of inputs, $x \in \mathbb{R}^p$, and a latent (unknown) input, $Z = g(x) \in \mathbb{R}$, which we infer from the data.

Our formulation relies on two stationary GPs, one for the function of interest and one for the latent input. Specifically, we assume

$$f|\theta \sim \text{GP}(\mu_\theta, C_\theta), \quad \text{and} \quad g|\theta \sim \text{GP}(0, \tilde{K}_\theta)$$

where $\theta$ denotes a vector of model parameters. We model $\mu_\theta$ as $\mu_\theta(x) = (1, x)^\top \beta$, and an “augmented” product-Gaussian correlation form is assumed for $K_\theta = \sigma^{-2} C_\theta$:

$$K_\theta\{x_i, x_j\} = \exp\left\{ -\sum_{l=1}^{p} \phi_l(x_{il} - x_{jl})^2 - \phi_{p+1}(Z_i - Z_j)^2 \right\}.$$

Thus, (4) corresponds to the standard (squared-exponential) correlation function of a stationary GP [1] indexed by $p + 1$ inputs. Furthermore, we model the correlation function of the GP on $g$, i.e. the correlation of the latent level GP, as

$$\tilde{K}_\theta(x_i, x_j) = \exp\left\{ -\sum_{l=1}^{p} \tilde{\phi}_l(x_{il} - x_{jl})^2 \right\},$$

with the scale parameter fixed to 1.

The idea behind the spatial deformation approach is a non-linear transformation of the locations $x$ into a latent space within which the correlation structure is stationary [26]. The mapping is done through a GP prior as in [29]. However, our construction differs from the one in [29], where $K_\theta$ is chosen to correspond to a mixture of Gaussian correlation functions, each of which depends on the Euclidean distance between the latent inputs $Z$'s only. While retaining an elegant formulation, our construction eases a more intuitive interpretation of the problem. We expect the latent process to mimic, at least qualitatively, the behavior of the response. The correlation between points near a sharp, localized feature is weakened since the corresponding distance has been stretched by the latent coordinate.

Clearly, the problem of modeling a non-stationary simulator could be tackled in different ways, e.g. one could proceed to a direct definition a non-stationary covariance function as in
As opposed to this approach, the latent extension of the input space guarantees positive definiteness of the covariance between observations in the original space and enhances an intuitive interpretation of the problem.

For completeness, we remark that a similar methodological idea was independently developed by [22] in the context of GP regression for non-stationary processes.

3. Implementation

This section presents a sequential MC implementation of our non-stationary GP. The approach relies on particle learning (PL) [17], which naturally blends with active learning of the design. The discussion of a two-stage approximation of the proposed emulator is deferred to Section 7.

First, we need to identify particles \( \{S_t^{(i)}\}_{i=1}^N \), which contain all the sufficient information about the uncertainties given data up to time \( t \), with \( N \) denoting the total number of particles. The sufficient information necessarily depends upon \([x_1, f(x_1)], \ldots, (x_t, f(x_t))] \), thus \( \{S_t^{(i)}\}_{i=1}^N = \{(Z_{1:t}, K_t, \hat{K}_t^{(i)})\} \), with \( Z_{1:t} \equiv (Z_1, \ldots, Z_t)^T \). The correlation functions have been indexed by \( t \) to stress their dependency to the data collected up to time \( t \). Particles do not contain \( \beta \) nor \( \sigma^2 \) as these parameters can be marginalized out within our Bayesian construction [13].

Particles are initialized at time \( t_0 > p + 1 \) with a sample of the unknown parameters from their prior distributions. The algorithm for updating particles \( \{S_t^{(i)}\}_{i=1}^N \) to \( \{S_{t+1}^{(i)}\}_{i=1}^N \) cycles through the following steps:

- **Resample** Generate index \( \zeta \sim \text{Multinomial}(w, N) \), with

\[
   w^{(i)} = \frac{\pi(f(x_{t+1})|S_t^{(i)})}{\sum_{i=1}^N \pi(f(x_{t+1})|S_t^{(i)})}, \quad i = 1, \ldots, N,
\]

where \( \pi(f(x_{t+1})|S_t^{(i)}) = \pi(f(x_{t+1})|[x, f(x)]_{1:t}, K_t^{(i)}) \) denotes the probability of observing \( f(x_{t+1}) \) under a Student-\( t \) distribution [13].

- **Propagate** \( S_t^{(i)} \) to \( S_{t+1}^{(i)} \): propagate each particle \( S_t^{(i)} \) to account for \( [x_{t+1}, f(x_{t+1})] \)

  - The first step requires constructing the “propagated” correlation function of the latent GP, which will be used to sample the latent coordinate at the new input \( x_{t+1} \). Thus, we build \( \tilde{K}_{t+1} \) from \( \hat{K}_t \) and \( \tilde{\kappa}_t^{(i)}(x_{t+1}) = \hat{K}^{(i)}(x_{t+1}, x_j) \), with \( j = 1, \ldots, t \)

\[
   \tilde{K}_{t+1} = \begin{bmatrix} \hat{K}_t & \tilde{\kappa}_t^{(i)}(x_{t+1}) \\ \hat{\kappa}_t^{(i)\top}(x_{t+1}) & \tilde{K}^{(i)}(x_{t+1}, x_{t+1}) \end{bmatrix}
\]

  - We obtain \( Z_{t+1}^{(i)} = g^{(i)}(x_{t+1}) \) from its predictive distribution \( g^{(i)}(x_{t+1})g^{(i)}(x_{1:t}), \tilde{K}_t^{(i)} \sim N(\mu^{(i)}, \tilde{\kappa}^{(i)}) \), where the mean and covariance are obtained via standard kriging equations.

\footnote{For coherence, we remark one should write \( f(x_1, Z_1), \ldots, f(x_t, Z_t) \) since the simulator is regarded as function of both the known and latent inputs. In the remainder, however, we will write \( f(x_1), \ldots, f(x_t) \) to simplify the notation.}
We construct the “propagated” correlation function of \( f \). We build \( K_t^{(i)} \) from \( K_t^{(i)}(x_{t+1}, x_j) \), \( j = 1, \ldots, t \), as

\[
K_{t+1}^{(i)} = \begin{bmatrix}
K_t^{(i)} & k_t^{(i)}(x_{t+1}) \\
k_t^{(i)\top}(x_{t+1}) & K_t^{(i)}(x_{t+1}, x_{t+1})
\end{bmatrix}
\]

Notice that the three sub-steps above can be performed in parallel across particles, with considerable gain in terms of computational speed.

The correlation range parameters and the latent input could be deterministically propagated by copying them from \( S_t^{(i)} \) to \( S_{t+1}^{(i)} \) since they do not change in \( t \). Although this strategy is fast, it could lead to particle depletion in future resampling steps. To avoid degeneracy, we include a “rejuvenate” step which applies Markov Chain Monte Carlo (MCMC) moves to the particles after the propagating step [10, 24]. The update is done via elliptical slice sampling [19].

We remark that each particle returns an estimate of predictive mean surface, \( \hat{f}^{(i)} \), and predictive standard deviation, \( \hat{\sigma}^{(i)} \). Likely, some of these particles will provide higher fidelity surfaces than others. We will take the average of the point-wise predictive distribution for each of the particles, the posterior mean predictive curve, as our prediction of \( f \) at new inputs

\[
\hat{f} = \mathbb{E}(f|S^{(i)}) = \frac{1}{N} \sum_{i=1}^{N} \hat{f}^{(i)},
\]

whereas the estimate for the predictive standard deviation is obtained as

\[
\sqrt{\hat{\sigma}^2} = \mathbb{E}(\{\hat{\sigma}^{(i)}\}^N_{i=1}) + \text{Var}(\{\hat{f}^{(i)}\}^N_{i=1}).
\]

The authors are currently working on an efficient C++ implementation of the PL algorithm to be used in an R package. The prototype R code is available upon request.

3.1. Adaptive sequential design. Several authors have developed specific criteria for sequentially selecting new input points. For instance, [14] proposed an expected improvement criterion to estimate the global minimum of a computer simulator via the maximum likelihood estimator for the emulator parameters. Equivalently popular approaches are the so-called “active learning” criteria such as ALM - active learning MacKay [18] and ALC - active learning Cohn [7]. [30] compared ALM and ALC and observed that ALC often performs better than ALM. For example, the ALM criterion embedded into a stationary GP emulator favors the selection of new points along the boundary of the input space in that the predictive variance is largest beyond the points which are already in the design [18]. However, the ALC criterion is more intensive to implement, therefore we will adopt ALM in our numerical examples for computational feasibility.

ALM-based selection of future inputs sits comfortably within our PL implementation. After particles have been resampled, the algorithm performs prediction at a set of candidate input configurations based on the posterior predictive distribution (see [13] for more details). ALM induces an ordering among candidate points based on their predictive standard deviation and the point with largest standard deviation in predicted output is chosen as the next input \( x_{t+1} \). Consequently, particles are propagated with the new pair \([x_{t+1}, f(x_{t+1})]\), and the sequence is iterated until some pre-specified stopping criterion is met, e.g. the largest...
Figure 2. Comparison between stationary GP (first panel), non-stationary GP via latent input augmentation (second panel), TGP (third panel), and CGP (fourth panel). Estimates (and RMSE) are obtained at 200 equally spaced test points. The dashed line corresponds to the true function \((f)\), the solid black line is the posterior mean predictive curve, and grey areas denote the error bars.

4. Case studies

4.1. Learning local features. We consider a spatially inhomogeneous smooth function:

\[
f(x) = \sin(x) + 2 \exp(-30x^2),
\]

which is evaluated at 15 equally spaced points in \(\Omega = [-2, 2]\).

For PL, we use \(N = 1000\) particles initialized at time \(t_0 = 4\) with a randomly selected subset of size 4 of the original 15 points. \(\{\phi_1, \phi_2\}\) and \(\tilde{\phi}_1\) are assigned log-normal priors distributions, and 0.5 and 0.25 are chosen as the prior mean and prior variance of the corresponding Normal distribution on \(\{\log \phi_1, \log \phi_2, \log \tilde{\phi}_1\}\). Also, a rather uninformative inverse-gamma prior is chosen for \(\sigma^2\), \(\sigma^2 \sim IG(2, 1)\).

Figure 2 shows the posterior mean predictive curve together with error bars computed as \(\hat{f} \pm 2\sqrt{\hat{\sigma}^2}\). We also show the results of fitting Bayesian TGP \([11]\) and composite GP (CGP) \([2]\) models. The limitations resulting from fitting a stationary GP to function \((8)\) were outlined in Section 1. In comparison, the three non-stationary emulators (panels 2-4 in Figure 2) give significantly improved performance, i.e. the spline tension effect is eliminated, or strongly attenuated. However, TGP’s most evident feature is the large uncertainty in the estimates as quantified by very wide error bars, which could be taken as indicator of an inadequate representation of the simulator. The error bars obtained with our non-stationary GP and CGP are more consistent with the local variability of the underlying surface. In terms of root mean squared error (RMSE), our emulator improves the accuracy of TGP and CPG by 25% and 40%, respectively.

4.2. Quantifying the emulator’s uncertainty. The simulator is typically expected to be within two or three standard deviations from the predictive mean \([3]\). While an isolated
outlier might be ignored, several large standardized residuals, e.g. more than 1% or 5% of the total number of validating points, may denote a problem to be further investigated. For example, large standardized residuals systematically observed in correspondence of a particular input suggest that the emulator is not learning the local behavior of the process [6]. Further, they indicate that the emulator is under-estimating the predictive uncertainty. Ultimately, one wants to acquire an accurate knowledge of $f$ with as least simulator’s runs as possible. The emulator can be used to quickly identify those regions of the input space where the simulator exhibits more variations, thus help determine where the simulator’s runs should concentrate. However, this goal can be achieved only if the emulator’s estimate of uncertainty is trustworthy. If not, a sequential design strategy based on uncertainty will lead to a sub-optimal selection of input points.

Here we examine how model-based evaluations (Figure 2 and first row in Figure 3) combine with extrinsic diagnostics (second row in Figure 3). According to the exploration-driven predictive standard deviation of a stationary GP, one is basically equally likely to locate the new point anywhere in $[-2, 2]$ (first panel in Figure 3). Instead, CV strongly favors the selection of a new input around $x = 0$ (exploitation-driven CV) to learn the local behavior of $f$. Thus, model-based evaluations and extrinsic diagnostics are inconsistent, and the latter shows that uncertainty is being under-estimated around the peak. Incongruent conclusions with CGP: if one trusts the model-based estimate of uncertainty, then the next input will be chosen around the peak; if one relies on CV, the next input will be chosen at the boundaries of the input space. For these two emulators, the problem of how to combine
different diagnostic results emerges clearly. Instead, both model-based evaluations and CV for our non-stationary GP and TGP identify that the next point is needed around $x = 0$. As opposed to our emulator, the conclusion with TGP is however made much more evident by extrinsic diagnostics (a strikingly large CV standardized residual at $x = 0$) rather than by the predictive standard deviation.

4.3. 1D discontinuous function. We now consider a simple discontinuous function:

$$f(x) = \begin{cases} 
0, & x \leq 0 \\
1, & x > 0
\end{cases}$$

with $x \in [-1, 1]$. We evaluate (9) at 10 equally spaced points in $\Omega = [-1, 1]$, and the initial design for PL is a randomly selected subset of size 4 of the same grid of points. This function is particularly suited to TGP because of the vertical, axis-aligned nature of localized feature.

As opposed to the stationary GP, the point predictions made by the three non-stationary emulators are not (or less) distorted by the spline tension effect (Figure 4), and the intervals seem more consistent with what might be guessed about the function from observing the data points. Again, TGP identifies large uncertainty everywhere in $\Omega$.

Model-base evaluations for our non-stationary GP and CGP suggest to pick new points at (and around) $x = 0$ to exploit the local feature (top row in Figure 5). Extrinsic and model-based evaluations are still inconsistent for the stationary GP, whereas extrinsic diagnostics show that TGP is likely to be under-estimating the uncertainty at the jump. Although this would lead to a sequential design strategy consistent with the one suggested by TGP’s model-based evaluation, it is preferable to observe the more apparent pattern in our emulators’s predictive standard deviation, which drops quickly when departing from $x = 0$.

In general, it is not clear how to reconcile model-based and extrinsic diagnostics whenever these lead to different evaluations. In particular, it is not obvious in what measure to favor the exploration-driven predictive standard deviation over the exploitation-driven CV. An emulator whose model-based evaluations reconcile with extrinsic diagnostics is preferred in that it automatically learns to create a good balance between exploration and exploitation, and one does not have to resort to ad-hoc combinations. Our emulator seems to accomplish this balance adequately.
Figure 5. Standard deviation at 200 predictive locations and leave-one-out CV standardized residuals for function (9): comparison between stationary GP, non-stationary GP via latent input, TGP, and CGP.

Figure 6. True functions for the 2-dimensional numerical examples.

5. High-dimensional examples and sequential design

5.1. Two-dimensional functions with local features. In this Section, we provide three test functions possessing non-stationary features (Figure 6). The second function ("building") is naturally suited to TGP because of the axis-aligned non-stationarity.

First, we compare the performance of the emulators trained on the same set of input points. We use a 40 Latin Hypercube Design (LHD – blue points in Figure 6), which allows the emulators gather knowledge on the overall shape of \( f \) because of its “space-filling” nature.
Figure 7. Building function estimates at $T = 40$. The quality of the prediction is assessed at a collection of 900 points in $[0,1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes. RMSE and maximum predictive standard deviation (max psd) based on the test points are also reported. Blue points are a randomly selected subset of the LHD used for initialization of PL.

Figure 1 in Web Appendix A and Figures 7-8 show the posterior predictive mean surface, $\hat{f}$, and the predictive standard deviation, $\hat{\sigma}$, for the menhir, building, and well functions, respectively. For the menhir function, the initial LHD does not include points at or nearby the peak, and this affects the estimates of the four emulators which can not recover the central spike. Note, however, how both our emulator and CGP identify higher uncertainty in the central part of the input space. This is also true for TGP, but the reason is likely to be related to the partitioning scheme rather than to learning the geometry of the feature. The most distinctive feature that emerges from both Figure 7 and Figure 8 is that our emulator is learning the geometry of the local features, as shown by the evident patterns in predictive standard deviation. This does not appear to be the case for the other emulators.

Next, we want to assess whether the emulators can correct for inadequacies in the fit. In other terms, we want to examine whether the emulators can learn about, and thus concentrate exploration in, the most interesting or complicated regions of the input space. Therefore, we let the emulators select 20 additional points (60 for well) sequentially via ALM.

Figures 2 and 3 in Web Appendix A show $\hat{f}$ and $\hat{\sigma}$ for the menhir and well functions at $T = 60$ and $T = 100$, respectively, and Figure 9 refers to the building function at $T = 60$. Regardless of the function being examined, our non-stationary emulator favors the sampling of new points from the boundaries of the features. Therefore, it strikes a good balance between exploration (initial LHD) and exploitation (newly selected points). This is not necessarily true for the other emulators across different functions, i.e. CGP tends to select new
Figure 8. Well function estimates at $T = 40$. The quality of the prediction is assessed at a collection of 900 points in $[0,1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes. RMSE and maximum predictive standard deviation (max psd) based on the test points are also reported. Blue points are a randomly selected subset of the LHD used for initialization of PL.

points at the center of the input space of the menhir function, but no pattern is observed for building and well functions. TGP’s selection is driven by the partitioning scheme in that the predictive standard deviation is generally higher at the edges between consecutive partitions. Thus, TGP seems to concentrate in learning the partition rather than the local feature.

For a more quantitative numerical comparison among the emulators, Figure 10 shows the progression of the RMSE as additional inputs are being selected. Our non-stationary emulator performs at least as well as CGP on the menhir function, and outperforms the other emulators on building and, in particular, well functions.

To conclude, our emulator is learning and concentrating the exploration in interesting areas of the input space. Furthermore, it compares favorably both in cases of axis-aligned non-stationarity (building) and in situations where the type of non-stationarity is more general (well).

5.2. Six-dimensional examples. We consider two 6D examples, which constitute an extension of the 2D building and well functions. The 6D building has true function:

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \begin{cases} \sum_{i=1}^{6} (\frac{1}{2})^2 x_i, & \text{if } x_1, x_2, x_3, x_4, x_5, x_6 > 0.25 \\ 0, & \text{otherwise} \end{cases}$$
Figure 9. Building function estimates at 60 design points. Blue points denote the additional inputs selected via the ALM criterion.

Figure 10. Progression of the RMSE as additional input points are being selected for the 2D functions.

on the hypercube $X = [0, 1]^6$. The 6D well has true function:

$$f(x_1, \ldots, x_6) = \begin{cases} 
1, & \text{if } \sum_{i=1}^{4}(x_i - 0.5)^2 > 0.025 \text{ and } \sum_{i=1}^{4}(x_i - 0.5)^2 < 0.25 \\
0, & \text{otherwise}
\end{cases}$$

on the hypercube $X = [0, 1]^6$. Therefore, $f$ in (11) is constant in $x_5$ and $x_6$.

For PL, $N = 1000$ particles are trained on a 120 LHD. Emulators then select 80 additional points from a 1000 candidate LHD according to ALM. Similar to the 2D examples, our non-stationary emulator outperforms the others in terms of reduction of RMSE (left and central panels in Figure 11). Therefore, inactive covariates which add noise to the process (6D well) do not affect the performance of our emulator. Additional summaries are reported in Web
Figure 11. Progression of the RMSE as additional input points are being selected. Comparison among stationary GP, non-stationary GP via latent input augmentation, TGP, and CGP. Left and central panels: 6D examples; Right panel: LGBB CFD experiment.

Appendix B.

6. LGBB CFD EXPERIMENT

This Section presents an application to a computational fluid dynamics (CFD) simulator of a proposed reusable NASA rocket booster vehicle, the Langley Glide-Back Booster (LGBB). The interest is in learning about the response in several flight characteristics of the LGBB as a function of three inputs (speed in Mach number, angle of attack, and slide-slip angle) when the vehicle reenters the atmosphere. See [12] for more details on the study.

The CFD simulation involves the iterative integration of systems of inviscid Euler equations and each run of the solver for a given set of parameters takes on the order of 5–20 hours on a high-end workstation [12]. Therefore, the interest in adaptively design the experiment to concentrate sampling in those regions where the response is more interesting (e.g., higher uncertainty or richest structure) emerges clearly. As [12] show, the most interesting region occurs near Mach 1 and for large angle of attack (refer to Figure 6 in Web Appendix C which shows the “lift” response as function of Mach and Alpha). The ridge in response at Mach equal to 1 separates subsonic flows and supersonic flows. The behavior of the response is quite different in the two regions, with lift appearing mostly homogeneous in the supersonic region.

Following [12], we examine the lift response as a function of speed (Mach) and angle of attack (Alpha) with the side-slip angle (Beta) fixed at zero. We obtain a linear interpolation onto a 30×30 grid over Mach and Alpha, and use the interpolated lift as our truth. Figure 12 shows a slice of the posterior mean predictive surface as a function of Mach and Alpha. The distinction between subsonic and supersonic flows is well captured by the non-stationary emulators, which tend to select new input points with small Mach, particularly for large Alpha. The stationary GP focuses mostly on a uniform exploration of the space and will require ad-hoc extrinsic diagnostics to focus around the ridge.
The third panel in Figure 11 shows the progression of RMSE to the interpolated truth. Our emulator performs as well as TGP on a surface that favors the latter because of the axis-aligned local feature, and improves the accuracy over stationary GP and CGP by 35% and 48%, respectively, at $T = 100$.

![Graph showing progression of RMSE to the interpolated truth with different emulators.]

**Figure 12.** LGBB slice of mean posterior predictive surface of the lift response as a function of Mach (speed) and Alpha (angle of attack) with Beta (side-slip angle) fixed at 0. The design was initialized with 20 randomly selected points from a 30×30 grid (black points), and 80 new points were selected via ALM (blue).

### 7. Two-stage empirical Bayes approximation of non-stationary GP emulator

#### 7.1. Implementation.** The modeling approach we have investigated in the previous Sections is formulated around an effort at joint modeling of both $f$ and $Z$. The latent input GP essentially becomes a vector of model parameters that can not be marginalized out within our construction, thus needs to be learnt. Bearing in mind that this GP is latent and the vector $Z$ is of sequentially increasing dimension, the learning of $Z$ can face challenges and a large amount of data may be needed to learn it well. Alternatively to this “full Bayes” approach, we elect here a cruder but much simpler strategy to approximate our non-stationary emulator. Specifically, we can rely on a two-stage approach where the first stage focuses on the estimation of the latent predictor, and the second stage focuses on learning $f$ assuming that the latent predictor is known and fixed at the level estimated in the first stage. Although several methods exist to obtain an estimate of the latent input GP at first stage, we investigate here an MCMC-based nonparametric regression approach.

Suppose we are given an initial design $\{x_i, f(x_i)\}_{i=1}^t$. We consider $Z = g(x)$ and model
$g$ by a stationary GP indexed by the $p$-dimensional vector of known inputs, $\mathbf{x}$, and a vector of model parameters, $\theta$:

$$g|\theta \sim \text{GP}(\tilde{\mu}_\theta, \tilde{K}_\theta).$$

As discussed above, several choices are available for the GP mean $\tilde{\mu}_\theta$, including the constant-zero mean. The correlation function $\tilde{K}_\theta$ corresponds to (5). If we consider a unit scale and fix $\tilde{\mu}_\theta \equiv 0$, (12) reduces to the GP prior on $g$ presented in (3). One can estimate $g$ from a smooth GP (noisy) regression:

$$f(\mathbf{x}_i) = g(\mathbf{x}_i) + \epsilon_i, \quad \text{with} \quad \epsilon_i \sim \text{N}(0, \tau^2), \quad \text{and} \quad i = 1, \ldots, t.$$  

Overall, the model in (12)-(13) is equivalent to assuming a GP prior on $f$:

$$f|\theta, \tau^2 \sim \text{GP}(\tilde{\mu}_\theta, \tilde{K}_\theta + \tau^2 \delta_{j,k}),$$

where $\delta_{\cdot, \cdot}$ is the Kronecker delta function. Bayesian inference of (14) proceeds via MCMC: realizations are drawn from the joint posterior distribution of the model parameters and, for all $\mathbf{x}$ of interest and using the parameter values drawn from the posterior distribution, we can estimate $Z$ at $\mathbf{x}$ as $\hat{g}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})|\mathbf{x}, \theta, \tau^2]$, the point predictor of $f$ at $\mathbf{x}$. Steps are repeated a large number of times, and the average of the point predictors is used as estimate of $g$. At the second stage, we consider $f|\hat{g}(\mathbf{x}), \theta \sim \text{GP}(\tilde{\mu}_\theta, \tilde{K}_\theta)$, where $f$ is now a stationary GP indexed by a $p+1$-dimensional vector of inputs $\{\mathbf{x}, \hat{g}(\mathbf{x})\}$ as in (3)-(4) under the fiction that the latent input is known. Similar to the first stage, inference for $\theta$ proceeds via MCMC. Similar to the “full Bayes” version of our method, prediction is made at a set of candidate points using the the parameter values drawn from the posterior distributions, and the estimated predictive uncertainty is used to guide the selection of new inputs. The two-stage, MCMC-based inference is perfectly coherent and comes closest to a full Bayesian treatment of the problem since it takes into account uncertainty in estimating the hyperparameters and the latent input GP at first stage. However, MCMC-based inference is ill-suited to sequential design, as the chain must be restarted and iterated until convergence when the design is augmented with a new pair $[\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1})]$. Fits from previous iterations can only guide the initialization of the new Markov Chain.

7.2. Simulation studies. We evaluate the performance of the two-stage approximation on the sequential experiments presented in the previous Sections. Figure 7 in Web Appendix D shows the performance of the two-stage approximation on the 2D examples of Section 5.1 given the initial 40 LHD. Estimating the latent input surface at first stage considerably improves the performance of our emulator on the Menhir function. Similar to what observed with the “full Bayes” implementation, the key feature is that the emulator is learning the geometry of the different features. This helps select new inputs from the edges of such features (Figure 13). Figure 14 shows a comparison between two-stage approximation and “full Bayes” in terms of progression of RMSE. No implementation is preferred in terms of predictive accuracy across functions or number of input points. The “full Bayes” approach is preferred on the well function, whereas two-stage seems to be preferred on the Menhir function for smaller designs. This is probably due to the ability of the two-stage approximation in better learning the latent input GP through the first noisy regression, which results into quicker learning of $f$. However, the predictive accuracy of the “full Bayes” approximation of our emulator considerably improves when one point is selected at the center of the input space (this happens at $t = 48$), and eventually reconciles with two-stage approximation.
Figure 8 in Web Appendix D shows that the full version of our emulator outperforms the two-stage approximation on the 6D and NASA experiments.

To conclude, the two-stage approximation of our emulator preserves some good features of the “full Bayes” version, namely learning the geometry of different types of shape and increasing the sampling frequency of new inputs along important input dimensions. Therefore, it constitutes a valid alternative to the full Bayesian implementation for adaptive design selection and function approximation. However, the “full Bayes” version often achieves lower RMSE, in particular for larger designs or in higher dimensions.

Figure 13. Two-stage approximation: the quality of the prediction is assessed at a collection of 900 points in [0, 1]^2, i.e. an expanded grid of 30 equally spaced points along each coordinate axes. Blue points are additional points selected via ALM criterion.

8. Discussion

In this work we describe a non-stationary GP model that can be used as an emulator in the sequential design of computer experiments. To induce non-stationarity, we consider a mapping to a latent space where stationarity holds, and augment the input space by the latent input. The numerical examples show that the extra flexibility introduced by the latent input greatly improves predictions over a stationary GP fit. In particular, the proposed methodology provides more reliable, model-based evaluations as opposed to extraneous explorations done with stationary GPs, and adapts to both cases of axis-aligned non-stationarity and in situations where the non-stationarity is more general. The approach also retains an easy interpretability while building upon a simple but elegant construction. Here we discuss some details in regard to our implementation and computer emulation in general.

The nugget. A “nugget” is a small, positive quantity α often added to the diagonal of the correlation function for f. The resulting covariance function corresponds to the case
where $f$ is observed with additive Gaussian noise with zero mean and variance $\alpha$. Many authors do not include a nugget term on the grounds that computer codes are deterministic. In fact, the nugget introduces a measurement error in the stochastic process. A GP that includes a nugget does not interpolate and assigns non-zero uncertainty to the design data. However, it is not uncommon practice to include a nugget to enhance the numerical stability in factorizing covariance matrices. A typical value of the nugget used in our numerical examples is $\alpha = 10^{-7}$ (PL implementation), the effect of this being the addition of $\alpha$ in the predictive variance of the responses. Although very small, $\alpha$ can have a non-negligible impact on the estimates. For example, it compromises interpolation of the stationary GP on the menhir function (Figure 2 in Web Appendix A). However, the nugget did not seem to significantly affect the estimates of our non-stationary GP. For more details on the inclusion of a nugget in computer emulation, refer to [1, 2].

High-dimensional problems. The application of the proposed methodology to high-dimensional input spaces can be challenging due to the intrinsic difficulty faced in high-dimensional settings by GP models, which try to recover up to the $p$-th level of interaction. We expect that more structure (i.e., additivity, sparse factorization) is needed to handle high dimensional problems. Independently developed research in the context of non-parametric regression suggests that additive GP models could be a promising way to move forward, and they will be investigated in future research.

Applications. Although the model was developed for the analysis of computer experiments, it also has a wide range of uses as a simple and efficient method for non-stationary modeling in the analysis of social, biological, and ecological data collected over spatial domains. The extension to non-parametric regression is straightforward with the inclusion of a nugget [29, 23, 16, 21].

References

[1] Ioannis Andrianakis and Peter G. Challenor. The effect of the nugget on Gaussian process emulators of computer models. *Comput. Stat. Data Anal.*, 56:4215–4228, 2012.
[2] Shan Ba and Roshan Joseph. Composite Gaussian process models for emulating expensive functions. *Annals of Applied Statistics.*, 6:1838–1860, 2012.

[3] L. S. Bastos and A. O’Hagan. Diagnostics for Gaussian process emulators. *Technometrics.*, 51:425–438, 2009.

[4] Maria J. Bayarri, J. O. Berger, R. Paulo, J. Sacks, J. A. Cafoe, J. Cavendish, C.-H. Lin, and J. Tu. A framework for validation of computer models. *Technometrics.*, 49:138–154, 2007.

[5] A. Bhattacharya, D. Pati, and B. D. Dunson. Adaptive dimension reduction with a Gaussian process prior. *arXiv: 1111.1044*, 2011.

[6] D. Busby. Hierarchical adaptive experimental design for Gaussian process emulators. *Reliability Engineering & System Safety.*, 94:1183–1193, 2009.

[7] David A. Cohn. Neural network exploration using optimal experiment design. *Neural Netw.*, 9:1071–1083, 1996.

[8] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *J. Am. Statist. Assoc.*, 86:953–963, 1991.

[9] Yalin Fan, Isaac Ginis, Tetsu Hara, C. Wayne Wright, and Edward J. Walsh. Numerical simulations and observations of surface wave fields under an extreme tropical cyclone. *J. Phys. Oceanogr.*, 39:2097–2116, 2009.

[10] Walter R. Gilks and Carlo Berzuini. Following a moving target: Monte Carlo inference for dynamic Bayesian models. *J. R. Statist. Soc. B.*, 63:127–146, 2001.

[11] Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *J. Am. Statist. Assoc.*, 103:1119–1130, 2008.

[12] Robert B. Gramacy and Herbert K. H. Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics.*, 51:130–145, 2009.

[13] Robert B. Gramacy and Nicholas G. Polson. Particle learning of Gaussian process models for sequential design and optimization. *J. Comput. and Graph. Statist.*, 20:102–118, 2011.

[14] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization.*, 13:455–492, 1998.

[15] Marc C. Kennedy and Anthony O’Hagan. Bayesian calibration of computer models. *J. R. Stat. Soc. B.*, 63:425–464, 2001.

[16] Hyoung-Moon Kim, Bani K. Mallick, and C. Holmes. Analyzing nonstationary spatial data using piecewise Gaussian processes. *J. A. Stat. Assoc.*, 100:653–668, 2005.

[17] H. F. Lopes, C. M. Carvalho, M. S. Johannes, and N. G. Polson. *Particle learning for sequential Bayesian computation*. Oxford University Press, 2011.

[18] David J.C. MacKay. Information-based objective functions for active data selection. *Neural Computation.*, 4:590–604, 1992.

[19] Iain Murray, Ryan P. Adams, and David J. C. MacKay. Elliptical slice sampling. *J. Mach. Learn. Res.*, 9:541–548, 2010.

[20] Christopher J. Paciorek and Mark J. Schervish. Nonstationary covariance functions for gaussian process regression. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 16, pages 273–280. MIT Press, 2004.

[21] Christopher J. Paciorek and Mark J. Schervish. Spatial modelling using a new class of nonstationary covariance functions. *Environmetrics.*, 17:483–506, 2006.

[22] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary Gaussian process regression using a latent extension of the input space. In *Presented as a poster at the ISBA Eight World Meeting on Bayesian Statistics*, 2006.

[23] Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems*, volume 14, pages 881–888. MIT Press, 2001.

[24] Greg Ridgeway and David Madigan. A sequential Monte Carlo method for Bayesian analysis of massive datasets. *J. Knowl. Discov. and Data Min.*, 7:301–319, 2003.

[25] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statist. Sci.*, 4:409–423, 1989.
[26] Paul D. Sampson and Peter Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *J. A. Stat. Assoc.*, 87:108–119, 1992.

[27] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments.* Springer Series in Statistics. Springer, 2003.

[28] Lars Schade and Kerry Emanuel. The ocean’s effect on the intensity of tropical cyclones: results from a simple coupled atmosphere-ocean model. *J. Atmos. Sci.*, 56:642–651, 1999.

[29] Alexandra Mello Schmidt and Anthony O’Hagan. Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *J. R. Stat. Soc.*, B., 65:745–758, 2000.

[30] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 241–246. IEEE, 2000.

[31] C. Textor, H. Graf, A. Longo, A. Neri, T. E. Ongaro, P. Papale, C. Timmreck, and G. G. J. Ernst. Numerical simulation of explosive volcanic eruptions from the conduit flow to global atmospheric scales. *Ann. of Geophys.*, 48:817–842, 2009.

[32] A.W. van der Vaart and J.H. van Zanten. Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth. *The Ann. of Statist.*, 37:2655–2675, 2009.

Dept. of Statistical Science, Duke University, Box 90251, Durham, NC, USA

E-mail address: sm234@duke.edu

Dept. of Statistical Science, Duke University, Box 90251, Durham, NC, USA

E-mail address: st118@stat.duke.edu
Web-based Supplementary Materials for “Computer emulation with non-stationary Gaussian processes” by Silvia Montagna and Surya T. Tokdar

SILVIA MONTAGNA and SURYA T. TOKDAR
Department of Statistical Science, Duke University, Durham, NC 27708, U.S.A.

Web Appendix A
Two-dimensional numerical examples

Figure 1: Menhir function estimates at $T = 40$ (LHD). Top row: posterior mean predictive surface, $\hat{f}$; bottom row: predictive standard deviation, $\hat{\sigma}$. The quality of the prediction is assessed at a collection of 900 points in $\Omega = [0, 1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes. Blue points correspond to the initial design used for PL for our non-stationary GP. We also report the root mean squared error (rmse) and the maximum predictive standard deviation (max psd), which are computed based on the test points.
Figure 2: Menhir function estimates at 60 design points. Black points denote the initial 40 LHD whereas blue points denote the additional inputs selected via the ALM criterion. Note how the stationary GP does not interpolate at the peak. We defer a discussion on this phenomenon in Section 8 of the paper.
Figure 3: Well function estimates at 100 design points. Black points denote the initial 40 LHD whereas blue points denote the additional inputs selected via the ALM criterion.
Recall that $\phi_j \geq 0 (\tilde{\phi}_j \geq 0)$ controls the sensitivity of $f(g)$ to $x_j$. Thus, $\phi_j = 0 (\tilde{\phi}_j = 0)$ removes $x_j$ (dimension reduction), whereas a larger $\phi_j (\tilde{\phi}_j)$ gives smaller correlation, i.e. $f(x)$ and $f(x')$ ($g(x)$ and $g(x')$) are less related in the $x_j$ direction and the function is more complex.

Figure 4 shows the distribution across particles of the estimated $\{\tilde{\phi}_i\}_{i=1}^6$ in the 6D well example. Note that $\tilde{\phi}_5$ are $\tilde{\phi}_6$ are estimated to be smaller than $\{\tilde{\phi}_i\}_{i=1}^4$, thus showing that our emulator is learning that $f$ is less sensitive to these input dimensions. Besides a few large isolated outliers, the correlation length parameters of $K$ are estimated to be small except for $\phi_7$, which is associated to the latent input $Z$ (Figure 5).

Figure 4: Distribution across particles of the correlation length parameters of $\tilde{K}$ at $T = 200$ in the 6-dimensional well example. The red curve denotes the prior distribution on the latent correlation length parameters $\{\tilde{\phi}_i\}_{i=1}^6$. Specifically, $\log \tilde{\phi}_i \sim N(0.5, 0.25), i = 1, \ldots, 6$. 
Figure 5: Distribution across particles of the correlation length parameters of $K$ at $T = 200$ in the 6-dimensional well example. The red curve denotes the prior distribution on the correlation length parameters $\phi_i$, $i = 1, \ldots, 7$. Specifically, $\log \phi_i \sim N(1, 0.25)$, $i = 1, \ldots, 7$. At every input configuration $\{x_1, x_2, \ldots, x_6\}_t$, corresponds an estimate of the latent input $Z$, where $\{x_i\}_{i=1}^6$ are known inputs. The last two panels show the estimated latent input $Z$ at each design point (initial 40 LHD + points selected via ALM). $Z$ is plotted versus the first and fourth dimension of the corresponding input configuration.
Web Appendix C
LGBB CFD experiment

Figure 6: Interpolated lift surface plotted as a function of Mach (speed) and Alpha (angle of attack) with Beta (side-slip angle) fixed to zero. The ridge at Mach 1 denotes a distinction between subsonic flows and supersonic flows. The upper-left corner of the plot (high angle of attack, low speed) shows a spike which is a result of false convergence of the simulator (Gramacy and Lee, 2008).
Web Appendix D

Two-stage approximation

![Two-stage approximation graphs](image)

**Figure 7:** Predictive surface and standard deviation at a set of 900 predictive points obtained with a two-stage MCMC-based implementation of the non-stationary GP emulator on the 2D numerical examples. Quantitative summaries report the root mean squared error (RMSE) and the maximum predictive standard deviation (pred sd) computed based on the test points. The fit is based on the same 40 LHD (black points) that was used in our 2D numerical examples in Section 5.1.

![Progression of RMSE graphs](image)

**Figure 8:** Progression of the RMSE as additional input points are being selected for the 6D and NASA examples. Comparison between “full Bayes” non-stationary GP (Full nstGP) and two-stage approximation (TS nstGP).