SIAM/ASA J. Uncertainty Quantification
Vol. 4, pp. 26–47

Computer Emulation with Nonstationary Gaussian Processes*

S. Montagna† and S. 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 suboptimal 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 nonstationary 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.

Key words. Bayesian inference, computer emulation, nonstationary Gaussian process, sequential design, particle learning, uncertainty quantification

AMS subject classifications. 62-XX, 62-07, 62G07, 62K20, 62L05, 62P30

DOI. 10.1137/141001512

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 [37, 12, 40]. 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 produces 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 $Y = f(\boldsymbol{x})$ 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" [35, 23, 42]. The emulator is fitted to input-output data $\{\boldsymbol{x}^t, f^t\}$, where $f^t = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_t)\}$ is obtained from a few preliminary runs of the simulator on design $\boldsymbol{x}^t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$, and the fitted model is then used for prediction of f at input configurations not included in $\boldsymbol{x}^t$ [35, 8].

*Received by the editors December 27, 2014; accepted for publication (in revised form) November 4, 2015; published electronically January 28, 2016.
http://www.siam.org/journals/juq/4/100151.html

†Department of Statistics, The University of Warwick, Coventry CV4 7AL, UK (S.Montagna@warwick.ac.uk).
‡Department of Statistical Science, Box 90251, Duke University, Durham, NC 27708-0251 (tokdar@stat.duke.edu).

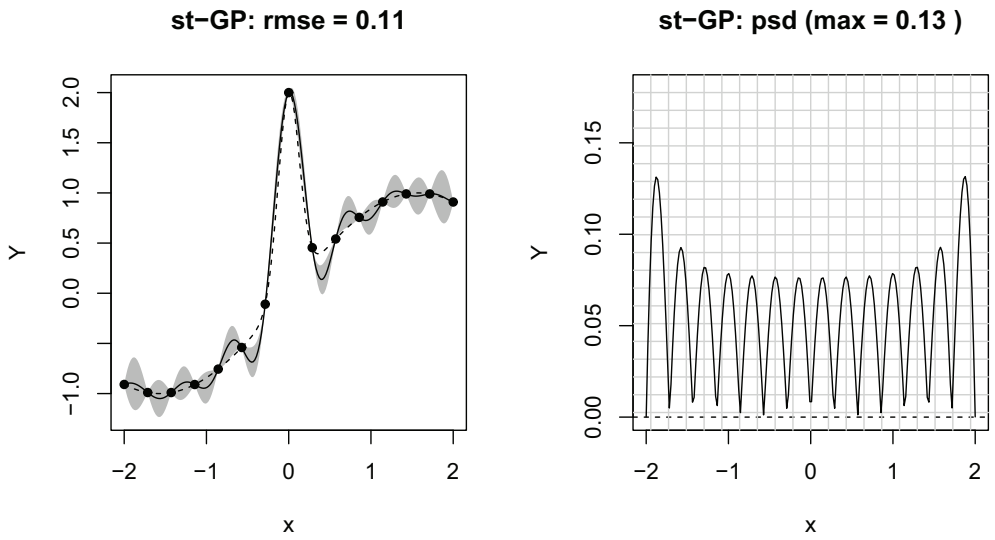

Figure 1. *Plot of (true) function* $f(x) = \sin(x) + 2\exp(-30x^2)$, $x \in [-2, 2]$ *(dashed line). The black dots represent observed data at* 15 *equally spaced values of* x. *Left: The solid line is the point predictor of* f, *or the conditional mean, obtained from a stationary GP emulator fitted to the data. Shaded areas represent the error bars. Right: Standard deviation evaluated at* 200 *predictive locations.*

For Bayesian emulation, a common practice is to assign f a Gaussian process prior [35, 10, 38]. 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 $\{\boldsymbol{x}^t, f^t\}$ and process hyperparameters, remains a GP distribution. The posterior mean of $f(\boldsymbol{x})$ gives a statistical estimate or surrogate for the simulator output at a new input $\boldsymbol{x}$, whereas the posterior variance at $f(\boldsymbol{x})$ quantifies how well the simulator output has been learned at and around $\boldsymbol{x}$. 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 [35, 23]. Stationary GPs regard the similarity between $f(\boldsymbol{x})$ and $f(\boldsymbol{x} + \boldsymbol{h})$ as a decaying function in $\boldsymbol{h}$ 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 a "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. When the training points are equally spaced, the predictive standard deviation given by the stationary GP is almost constant (right panel in Figure 1). Also, the predictive standard deviation gets larger as the distance between the test point and the training data increases. Therefore, a sequential design strategy based on uncertainty as provided by a stationary GP does not naturally favor the *exploitation* of regions that are deemed important based on the current estimate of f but rather a uniform design with selection of new points in unexplored regions of the space (*exploration*).

Extrinsic diagnostics are often used to assess the adequacy of a GP emulator as surrogate for the simulator [5, 4]. 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_ν) 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 subregions 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 [8], 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 [8].

Several approaches proposing nonstationary GP models can be found in the literature. In the context of computer emulation, [17] proposes 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 ("axis-aligned" nonstationarity). However, it may run into difficulties when the nature of the nonstationarity is more general. Also, the sharing of information across partitions is limited and global; i.e., sharing does not have any local-global decay. TGP bears some similarity to the work in [24], which adopts mixtures of GPs defined locally on a Voronoi tesselation. Voronoi tessellations allow for complex partitioning of the input space and are not restricted to being axis-aligned, but have the trade-off of increased complexity. [3] decomposes f into the sum of two stationary GPs, the first capturing the smooth global trend and the second modelling local details. More recently, [16] provides a computation-aware framework for computer experiments by means of local GP approximation. That work modernizes the idea of local neighborhood kriging (e.g., [41]), which has seen a resurgence lately as a tool to simultaneously tackle large data sets and nonstationary modelling. Other approaches in the context of GP regression include those of [36, 38, 29, 11]. This literature makes it clear that the main challenges in nonstationary GP modelling are to keep the number of hyperparameters under control, to facilitate efficient learning from limited data while allowing for nonstationary features of various geometric shapes, and, at the same time, not to enforce nonstationarity when not needed.

[7] proposes a new approach to the modelling of nonstationary processes through dimension expansion. The method is superficially similar to that of image warping in [36], but here the authors retain the locations in the geographical space and achieve greater flexibility by adding extra dimensions. Specifically, the authors introduce extra dimensions for the observed inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$, notated as $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_t$ with $\boldsymbol{Z}_i \in \mathcal{R}^d$, $d > 0$, such that the field $f(\boldsymbol{x}, \boldsymbol{Z})$ is

stationary with a variogram model $\gamma_\phi([\boldsymbol{x}_i\, \boldsymbol{Z}_i] - [\boldsymbol{x}_j, \boldsymbol{Z}_j])$. The latent dimensions are learned nonparametrically through optimization and from information contained within the data. The number of augmented dimensions d is estimated through the use of a group lasso penalty to avoid overfitting and to regularize the optimization problem. Once the $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_t$ are found, a function g is built such that $g(\boldsymbol{x}) \approx \boldsymbol{Z}$. The authors choose thin plate splines, one for each dimension of $\boldsymbol{Z}$. In many situations, however, it is unclear how many additional dimensions are needed to accurately model the spatial process. In this paper, we build on [7] but assume that the process $f(\boldsymbol{x}, Z) : (\boldsymbol{x}, Z) \in \mathcal{R}^{p+1}$ is stationary; that is, we add only one latent dimension to achieve a stationary process. None of the examples considered in this paper lead us to believe that additional latent dimensions were needed, but it is possibly an idea worth exploring in more complicated settings. The latent input, which is inferred from the data, can flag regions of the input space characterized by abrupt changes of the function values and help correct for inadequacies in the fit.

In addition to proposing an emulator which is adaptable to local features of many kinds of shapes, we also want to use our emulator for online learning of an optimal sequence of design points. To address this goal, it is absolutely crucial to have trustworthy judgement of uncertainty of the current estimate of f to concentrate efforts only 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, when performance is measured by the number of simulator runs needed to achieve a certain accuracy. We remark that there is no attempt to emulate computer models or address sequential design in [7].

The proposed method is also attractive from an operational point of view. In [7], the choice of the mapping, learning of the latent inputs, and predictions are performed in isolation, whereas our approach is set within a Bayesian framework which allows uncertainty to be accurately reflected in resulting inferences. Both the latent input dimension and the response function (of the original plus the latent inputs) are individually modelled as stationary GPs controlled by a small number of hyperparameters that can be efficiently learned with sequential Monte Carlo (MC) computing, leveraging the conjugacy properties of GPs. Sequential MC computing seamlessly blends with active learning of the sequential design, in contrast to Markov chain sampling–based nonstationary GP emulators, whose sequential adaptation requires rerunning the whole Markov chain sampler at every iteration.

The remainder of the paper is outlined as follows. Section 2 begins with an overview of GP emulation and stationarity and then introduces our nonstationary GP emulator. Section 3 presents a fast sequential design algorithm for GP emulation. Section 4 examines the performance of different emulators in quantifying uncertainty through a one-dimensional numerical example. In section 5, we investigate sequential design via higher-dimensional examples. Section 6 presents a real data application. Conclusions are reported in section 7.

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 $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)^\top$ the joint distribution of $(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_t))^\top$ is multivariate Gaussian with mean $\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x})$ and positive definite covariance matrix $\mathbb{C}\mathrm{ov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') =$

$\sigma^2 K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ parameterized by $\boldsymbol{\theta}$. Note that we can write the GP emulator as

$$f(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \epsilon(\boldsymbol{x}; \boldsymbol{\theta}), \tag{2.1}$$

where $\epsilon(\boldsymbol{x}; \boldsymbol{\theta})$ is a zero-mean GP with covariance function $C_{\boldsymbol{\theta}}(\cdot, \cdot)$. To simplify the notation, we shall drop the $\boldsymbol{\theta}$ subscript to the covariance and correlation functions hereafter.

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

$$\hat{f}(\tilde{\boldsymbol{x}}) = \mathbb{E}[f(\tilde{\boldsymbol{x}}) | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{\theta}] = \mu(\tilde{\boldsymbol{x}}) + k^\top(\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} (\boldsymbol{F} - \mu(\boldsymbol{X})) \tag{2.2}$$

and variance

$$\hat{\sigma}^2(\tilde{\boldsymbol{x}}) = \mathbb{V}[f(\tilde{\boldsymbol{x}}) | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{\theta}] = \sigma^2 \{K(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - k^\top(\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} k(\tilde{\boldsymbol{x}})\}, \tag{2.3}$$

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

The mean field $\mu(\boldsymbol{x})$ in (2.1) is typically given the linear model structure $\mu(\boldsymbol{x}) = h(\boldsymbol{x})^\top \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a vector of unknown parameters. Although $h(\cdot)$ may be any function on the input space $\mathcal{X}$, we adopt a linear mean in the inputs, $h(\boldsymbol{x}) = [1, x_1, \ldots, x_p]^\top$. This seems to be a natural choice with little prior information about the input-output relationship and helps to control overfitting. The correlation function is crucial in GP modelling; it is through $K(\boldsymbol{x}, \boldsymbol{x}')$ that we express a belief about how similar $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ should be if $\boldsymbol{x}$ and $\boldsymbol{x}'$ were close in $\mathcal{X}$, and thereby we express a belief about the smoothness of f. Although different formulations are possible [34], in this work we focus on the power family and use the separable squared-exponential correlation function

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\sum_{l=1}^p \phi_l (x_l - x_l')^2}. \tag{2.4}$$

Parameters $\{\phi_l\}_{l=1}^p$ are inferred as part of our estimation procedure. Thus, the correlation is a function only of $\boldsymbol{x} - \boldsymbol{x}'$ (stationarity) and a set of inverse range (unknown) parameters.

We embed our approach in a Bayesian framework and proceed by specifying prior distributions for the model parameters. Hereafter, we use an improper uniform prior $\boldsymbol{\beta} \propto 1$ as conventional representation of weak prior information about $\boldsymbol{\beta}$; an inverse-gamma (IG) prior for the scale, $\sigma^2 \sim \mathrm{IG}(a/2, b/2)$; and a log-normal prior for the inverse range parameters, $\phi_l \sim \log \mathrm{N}(\mu_\phi, \nu_\phi)$, but other formulations are possible [17]. The posterior predictive distribution of f at a new input $\tilde{\boldsymbol{x}}$, conditioned on data $\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}$ and correlation matrix $\boldsymbol{K}$ and marginalized with respect to $\{\boldsymbol{\beta}, \sigma^2\}$, is a Student-$t$ distribution with $\nu = t - p - 1$ degrees of freedom, mean

$$\hat{f}(\tilde{\boldsymbol{x}} | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{K}) = h(\tilde{\boldsymbol{x}})^\top \tilde{\boldsymbol{\beta}} + k^\top(\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} (\boldsymbol{F} - \boldsymbol{H}^t \tilde{\boldsymbol{\beta}}), \tag{2.5}$$

and variance

$$\hat{\sigma}^2(\tilde{\boldsymbol{x}} | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{K}) = \frac{(b + \Phi)[K(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - k^\top(\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} k(\tilde{\boldsymbol{x}})]}{a + \hat{\nu}}, \tag{2.6}$$

where $\boldsymbol{H}^t$ is the $t \times (p+1)$ matrix which contains $h(\boldsymbol{x}_i)^\top$ in its rows and

$$\begin{aligned}\Phi &= \boldsymbol{F}^\top \boldsymbol{K}^{-1} \boldsymbol{F} - \tilde{\boldsymbol{\beta}}^\top \Psi^{-1} \tilde{\boldsymbol{\beta}},\\ \tilde{\boldsymbol{\beta}} &= \Psi(\boldsymbol{H}^t \boldsymbol{K}^{-1} \boldsymbol{F}),\\ \Psi &= (\boldsymbol{H}^t \boldsymbol{K}^{-1} \boldsymbol{H}^t)^{-1}.\end{aligned}$$

The availability in closed form of the marginalized predictive distribution is crucial for the sequential design algorithm implementing our nonstationary GP emulator (section 3).

2.2. Nonstationary GP through latent input augmentation. In computer emulation, it is not uncommon to observe functions that vary more quickly in some parts of the input space than in others [17]. In response to concerns about the adequacy of the stationary assumption for GP emulators, we build on the concept of spatial deformation [36] and model $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \epsilon([\boldsymbol{x}, Z]; \boldsymbol{\theta}), \tag{2.7}$$

where $\mu(\boldsymbol{x}) = h(\boldsymbol{x})^\top \boldsymbol{\beta}$ and $\epsilon([\boldsymbol{x}, Z]; \boldsymbol{\theta})$ is a zero-mean GP whose covariance function depends smoothly on the p-dimensional (known) vector of inputs, $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^p$ a latent (unknown) input Z which we infer from the data, and a handful of model parameters $\boldsymbol{\theta}$. Specifically, we adopt an "augmented" squared-exponential correlation form for $K = \sigma^{-2} C$:

$$K[(\boldsymbol{x}_i, Z_i), (\boldsymbol{x}_j, Z_j)] = \exp\left\{ -\sum_{l=1}^{p} \phi_l (x_{il} - x_{jl})^2 - \phi_{p+1} (Z_i - Z_j)^2 \right\}. \tag{2.8}$$

Several alternative approaches to the modelling of nonstationary simulators can be found in the literature. For example, [34] proposes gathering substantial information about the simulator from experts and uses it to include a large number of regressors in the prior mean field μ while retaining a stationary residual process. However, choosing the best set of regressors is nontrivial, and there is no guarantee that the residual process is stationary. Also, the authors suggest using rougher (but still stationary) correlation functions like the Matérn. While this could lead to a better fit compared to smoother alternatives, it would require specifying an extra smoothing parameter which is hard to infer statistically. Also, the stationary representation does not address the issue that stationary GP processes focus more on exploration rather than exploitation of the input space, thus still leading to a suboptimal sequential design strategy in the presence of local features. [1] let the smoothness parameter of the Matérn model vary with the input $\boldsymbol{x}$ and estimated this and the remaining parameters of the process via a likelihood-based approach. This approach is desirable for functions that are smooth at most $\boldsymbol{x}$ but exhibit lack of smoothness along lower-dimensional sets. It is, however, a rather demanding approach computationally. [29] generalizes Gibbs' construction [13] to obtain nonstationary versions of arbitrary isotropic covariance functions. While their model provides a flexible and general framework, it is computationally demanding and not feasible in high-dimensional spaces. 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. When thinking of emulation of computer models that are characterized by sharp local features, the extra input could tear apart regions of the input

space that are separated by abrupt changes of the function values. The correlation between points at and about a localized feature is weakened since the corresponding distance has been stretched by the latent coordinate.

To this point we have not made any assumptions about the latent input Z. In the following, we model Z as a continuous function of the inputs, $Z_i = g(\boldsymbol{x}_i) \in \mathbb{R}$, using a stationary GP:

$$g \mid \boldsymbol{\theta} \sim \mathrm{GP}(\mathbf{0}, \tilde{\boldsymbol{K}}), \qquad \text{with} \tag{2.9}$$

$$\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\sum_{l=1}^{p} \tilde{\phi}_l (x_{il} - x_{jl})^2\right\}, \tag{2.10}$$

where the scale parameter is fixed to 1.

To summarize, our formulation relies on two stationary GPs, one for the function of interest and one for the latent input,

$$f|\boldsymbol{\theta} \sim \mathrm{GP}(\boldsymbol{\mu}, \boldsymbol{C}) \quad \text{and} \quad g|\boldsymbol{\theta} \sim \mathrm{GP}(\mathbf{0}, \tilde{\boldsymbol{K}}), \tag{2.11}$$

where vector $\boldsymbol{\theta}$ collects all the parameters of both the original and latent processes, $\boldsymbol{\theta} = [\boldsymbol{\beta}, \sigma^2, \{\phi_l\}_{l=1}^{p+1}, \{\tilde{\phi}_l\}_{l=1}^{p}]$. A stochastic process for $f(\boldsymbol{x})$ is achieved by integrating out the regression parameters $\boldsymbol{\beta}$ and Z, and is more adaptive than (2.1) to functions whose smoothness varies with the inputs because it has the capacity to have several length scales. The extended process $f(\boldsymbol{x}, Z)$ is stationary, while the marginal $f(\boldsymbol{x})$ is not. For example, imagine a spatial environmental process which is stationary given the location and elevation but may result in a nonstationary field given only longitude and latitude. In this example, elevation is the latent input and, once learnt, the expanded process becomes stationary. In general, the latent dimension does not need to have a physical meaning, but it is only used to stretch the distance between inputs characterized by abrupt changes in function values.

The predictive formulas for our latent input model are similar to (2.2) and (2.3). In addition to averaging over the set of hyperparameters $\boldsymbol{\theta}$, it is also necessary to average over the posterior distribution of the latent inputs to obtain

$$\begin{aligned}\mathbb{E}[f(\tilde{\boldsymbol{x}})|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \tilde{\boldsymbol{x}}] = \int_{\mathcal{Z}} \int_{\Theta} &\{\mu(\tilde{\boldsymbol{x}}) + k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})^\top \boldsymbol{K}(\boldsymbol{\theta}, Z_{1:t})^{-1}[\boldsymbol{F} - \mu(\boldsymbol{X})]\} \\ &\times p(\boldsymbol{\theta}, Z_{1:t}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})\, d\boldsymbol{\theta} dZ\end{aligned} \tag{2.12}$$

and

$$\begin{aligned}\mathbb{V}\mathrm{ar}[f(\tilde{\boldsymbol{x}})|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \tilde{\boldsymbol{x}}] &= \mathbb{E}_{\boldsymbol{\theta},Z}[\mathbb{V}\mathrm{ar}(f(\tilde{\boldsymbol{x}})|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \tilde{\boldsymbol{x}}, \boldsymbol{\theta}, Z_{1:t})] \\ &\quad + \mathbb{V}\mathrm{ar}_{\boldsymbol{\theta},Z}[\mathbb{E}(f(\tilde{\boldsymbol{x}})|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \tilde{\boldsymbol{x}}, \boldsymbol{\theta}, Z_{1:t})] \\ &= \int_{\mathcal{Z}} \int_{\Theta} \sigma^2\{K(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}; \boldsymbol{\theta}, \tilde{Z}) - k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})^\top \boldsymbol{K}(\boldsymbol{\theta}, Z_{1:t})^{-1} k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})\} \\ &\quad \times p(\boldsymbol{\theta}, Z_{1:t}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})\, d\boldsymbol{\theta} dZ_{1:t} d\tilde{Z} \\ &= \int_{\mathcal{Z}} \int_{\Theta} [\{\mu(\tilde{\boldsymbol{x}}) + k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})^\top \boldsymbol{K}(\boldsymbol{\theta}, Z_{1:t})^{-1}[\boldsymbol{F} - \mu(\boldsymbol{X})]\} - \mathbb{E}\{f(\tilde{\boldsymbol{x}})|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \tilde{\boldsymbol{x}}\}]^2 \\ &\quad \times p(\boldsymbol{\theta}, Z_{1:t}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})\, d\boldsymbol{\theta} dZ_{1:t} d\tilde{Z},\end{aligned} \tag{2.13}$$

where $p(\boldsymbol{\theta}, Z_{1:t}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})$ is the posterior joint density of the latent variables and hyperparameters; $\boldsymbol{K}(\boldsymbol{\theta}, Z_{1:t})$ is the correlation matrix of f, which depends on $\boldsymbol{\theta}$ and $Z_{1:t}$; and $k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})$ is the vector of correlations of the response at the new input $\tilde{\boldsymbol{x}}$ with the responses in the design. This last quantity depends on $\tilde{Z} = g(\tilde{\boldsymbol{x}})$, the latent variable at the new input. Since we do not observe $g(\tilde{\boldsymbol{x}})$, we draw it from its predictive distribution $g(\tilde{\boldsymbol{x}}) \mid g(\boldsymbol{x}_{1:t}), \tilde{\boldsymbol{K}} \sim \mathrm{N}(\boldsymbol{\mu}^*, \tilde{\boldsymbol{K}}^*)$, where mean and variance are obtained via standard kriging equations. In general, the predictive variance of the response for points in a test set (see (2.13)) depends on the value of the latent input at such points via $k(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}, Z_{1:t}, \tilde{Z})$. The nonstationary correlation structure thus allows predictive variances to change across the region even though the marginal variance of the process is constant.

As mentioned above, [36] first pioneered an approach to the problem of nonstationarity and anisotropy in environmental datasets through a nonlinear transformation of the sampling space into a latent space with stationary and isotropic spatial structure. The mapping was done via multidimensional scaling. Further, the authors used thin-plate splines to estimate realizations of f at predictive locations while keeping the estimates of the latent process fixed and without taking into account any measure of uncertainty about the mapping [38]. The approach we propose in (2.11) is similar in flavor to the construction in [38], which built on [36] and implemented spatial deformation via a GP prior. However, our construction differs from that of [38], where K is chosen to correspond to a mixture of Gaussian correlation functions, each of which depends on the Euclidean distance between the latent inputs Z only. Also, those authors infer their deformation from an observation of a sample covariance matrix. The idea of achieving nonstationarity by latent input extension can also be found in [31], which presents two approaches for approximate Bayesian inference in GP regression models. The first method relies on a discrete latent input and is implemented in a Markov chain Monte Carlo (MCMC) sampling scheme, whereas the second method estimates a continuous latent mapping by evidence maximization. We remark that [36, 38, 29, 31] are not attempting to estimate a deterministic model or perform sequential design.

3. Implementation.

We apply our adaptive nonstationary GP emulator to the sequential design of computer experiments [42]. Sequential design is crucial to keeping designs small and saving on expensive runs of the simulator while guaranteeing adequate learning of the input-output map. In this paper, we adopt a technique known as particle learning (PL) to obtain a quick update of the emulator after each sequential design iteration. An introduction to PL is beyond the scope of this paper. The unfamiliar reader can refer to [25] for an introduction and to [20] for its application to the online updating of GP regression models. The implementation outlined here follows along the same lines of [20], but a few augmentations are required to accommodate the extra latent variable and the coupling of the two GPs. These augmentations will be specified below as appropriate.

PL provides a simulation-based approach to sequential Bayesian computation. Central to PL is the identification of *essential state vectors* or particles, $\{S_t^{(i)}\}_{i=1}^N$, that are tracked sequentially, with N denoting the total number of particles. These particles contain all the sufficient information about the uncertainties given the data up to time t and are used to approximate the posterior distribution, $\{S_t^{(i)}\}_{i=1}^N \sim \pi(S_t \mid \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})$. PL provides a method for updating the particles from t to $t+1$.

We start by identifying the quantities each particle includes. The sufficient information necessarily depends upon $[(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_t, f(\boldsymbol{x}_t))]$;[1] thus $\{S_t^{(i)}\}_{i=1}^N = \{(\boldsymbol{Z}_{1:t}, \boldsymbol{K}_t, \tilde{\boldsymbol{K}}_t)^{(i)}\}$, with $\boldsymbol{Z}_{1:t} \equiv (Z_1, \ldots, Z_t)^\top$. The correlation functions have been indexed by t to stress their dependence on data collected up to time t. Particles do not contain $\boldsymbol{\beta}$ or σ^2, as these parameters can be marginalized out within our Bayesian construction [20]. As opposed to [20], in our implementation each particle needs to store two additional quantities related to the latent process, namely $\boldsymbol{Z}_{1:t}$ and $\tilde{\boldsymbol{K}}_t$.

Suppose we start with $(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_{t_0}))^\top$ obtained from $t_0 > p+1$ preliminary runs of the simulator at design points $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t_0})^\top$. The initial design can be chosen as, e.g., a Latin hypercube design (LHD) or a uniform design. Particles are initialized at time t_0 with a sample of the unknown parameters from their prior distributions. In addition to the priors in section 2.1, we also sample $\{\tilde{\phi}_l\}_{l=1}^p$ from $\tilde{\phi}_l \sim \log \mathrm{N}(m_{\tilde{\phi}}, v_{\tilde{\phi}})$.

Suppose the sampler has been run t times and the current design is $[\boldsymbol{x}, f(\boldsymbol{x})]_{1:t}$, where $\boldsymbol{x}_t$ is the latest point included in the design and chosen at the previous iteration. Next, we describe the backbone steps of PL that are used to update particles $\{S_t^{(i)}\}_{i=1}^N$ to $\{S_{t+1}^{(i)}\}_{i=1}^N$ at iteration $t+1$:

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

$$w^{(i)} = \frac{\pi(f(\boldsymbol{x}_t)|S_{t-1}^{(i)})}{\sum_{i=1}^N \pi(f(\boldsymbol{x}_t)|S_{t-1}^{(i)})}, \qquad i = 1, \ldots, N,$$

  where $\pi(f(\boldsymbol{x}_t)|S_{t-1}^{(i)}) = \pi(f(\boldsymbol{x}_t)|[\boldsymbol{x}, f(\boldsymbol{x})]_{1:(t-1)}, \boldsymbol{K}_{t-1}^{(i)})$ denotes the probability of observing $f(\boldsymbol{x}_t)$ under a Student-t distribution with $\nu = (t-1)-p-1$ degrees of freedom and mean and variance given by (2.5)–(2.6), respectively. Therefore, each particle is resampled with probability proportional to the likelihood of observing $\boldsymbol{x}_t$.
- *Input selection via active learning MacKay (ALM) heuristics* [26]: After particles have been resampled, the algorithm performs prediction at a set of candidate input configurations based on the posterior predictive distribution. We first need to obtain an estimate of the latent input $g(\tilde{\boldsymbol{x}})$ at every candidate point via standard kriging equations conditional on $\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}$ and the latent process-specific parameters. Then, given the latent input, we derive $\hat{f}(\tilde{\boldsymbol{x}}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})$, which predicts f at $\tilde{\boldsymbol{x}}$, and $\hat{\sigma}^2(\tilde{\boldsymbol{x}}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})$, which quantifies the uncertainty at $\tilde{\boldsymbol{x}}$. The cost is $O(t)$ for computing the predictive mean, and $O(t^2)$ for the predictive variance for each test case given $\boldsymbol{K}_t^{-1}$. Candidate points can then be ordered based on their predictive variance, and the point with largest uncertainty in predicted output is chosen as the next input, thus leading to the new pair $[\boldsymbol{x}_{t+1}, f(\boldsymbol{x}_{t+1})]$. This step is not specific to PL, but it is used here and in [20] to use the surrogate model's fit for an "informed" selection of new inputs, thus combining PL with sequential design.
- *Propagate*: Update each resampled particle $S_t^{\zeta(i)}$ to $S_{t+1}^{(i)}$ to account for $[\boldsymbol{x}_{t+1}, f(\boldsymbol{x}_{t+1})]$, via the following steps:

[1] To stress the dependence of f on both known and latent inputs within our approach, we should write $f(\boldsymbol{x}, Z)$. In the remainder, however, we will omit Z and write $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_t)$ to simplify the notation.

– Construct the "propagated" correlation function of the latent GP. We build $\tilde{\boldsymbol{K}}_{t+1}^{(i)}$ from $\tilde{\boldsymbol{K}}_t^{(i)}$ and $\tilde{\boldsymbol{k}}_t^{(i)}(\boldsymbol{x}_{t+1}) = \tilde{\boldsymbol{K}}^{(i)}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_j)$, with $j = 1, \ldots, t$:

$$\tilde{\boldsymbol{K}}_{t+1}^{(i)} = \begin{bmatrix} \tilde{\boldsymbol{K}}_t^{(i)} & \tilde{\boldsymbol{k}}_t^{(i)}(\boldsymbol{x}_{t+1}) \\ \tilde{\boldsymbol{k}}_t^{(i)\top}(\boldsymbol{x}_{t+1}) & \tilde{K}^{(i)}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) \end{bmatrix}.$$

– Obtain the latent input at the newly included design point $\boldsymbol{x}_{t+1}$, $g^{(i)}(\boldsymbol{x}_{t+1})$. With predictions carried out at the previous step (*ALM heuristics*), one just needs to store the value of $g^{(i)}(\boldsymbol{x}_{t+1})$ previously estimated, and no further computation is involved.

– Construct the "propagated" correlation function of f. We build $\boldsymbol{K}_{t+1}^{(i)}$ from $\boldsymbol{K}_t^{(i)}$ and $\boldsymbol{k}_t^{(i)}(\boldsymbol{x}_{t+1}) = \boldsymbol{K}^{(i)}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_j)$, $j = 1, \ldots, t$, as

$$\boldsymbol{K}_{t+1}^{(i)} = \begin{bmatrix} \boldsymbol{K}_t^{(i)} & \boldsymbol{k}_t^{(i)}(\boldsymbol{x}_{t+1}) \\ \boldsymbol{k}_t^{(i)\top}(\boldsymbol{x}_{t+1}) & \boldsymbol{K}^{(i)}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) \end{bmatrix}.$$

Of these three substeps, the last corresponds to the propagate step described in [20]. The first two substeps are specific to our implementation and are needed to propagate the latent process as well as f.

- *Rejuvenate*: The inverse range parameters and the latent input could be deterministically propagated by copying them from $S_t^{\zeta(i)}$ to $S_{t+1}^{(i)}$ since they do not change in t. Although this strategy is fast, it could lead to particle depletion. To avoid degeneracy in the path space caused by successive resampling steps, we apply MCMC moves ($O((t+1)^3)$) to the particles [14, 33], and update all parameters $[\{\phi_l\}_{l=1}^{p+1}, \{\tilde{\phi}_l\}_{l=1}^{p}, \boldsymbol{Z}_{1:t}]$ via elliptical slice sampling [27]. Again, the difference with [20] in this "rejuvenate" step is related to the additional updates required for the latent process-specific parameters and extra coordinate. Given the updated set of parameters, we need to rebuild the correlation matrices for f and the latent process, with complexities $(p+1)(t+1)^2$ and $p(t+1)^2$, respectively, and find their Cholesky decompositions, with complexity $(t+1)^3$.

This sequence of steps is iterated until some prespecified stopping criterion is met, e.g., the largest predictive variance falls below a certain threshold, or a total number, T, of points has been included in the design. We remark that all but the resampling step can be performed in parallel across particles; thus one can benefit from multicore or other distributed computing structures to speed up the computation.

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 take the average of the pointwise predictive distribution for each of the particles, the posterior mean predictive curve, as our prediction of f at new inputs,

$$\hat{f} = \mathbb{E}(f|S^{(i)}) = \frac{1}{N}\sum_{i=1}^{N} \hat{f}^{(i)}, \tag{3.1}$$

whereas the estimate for the predictive standard deviation is obtained as

$$\sqrt{\hat{\sigma}^2} = \mathbb{E}\big(\{\hat{\sigma}^{(i)}\}_{i=1}^N\big) + \mathbb{V}\mathrm{ar}\big(\{\hat{f}^{(i)}\}_{i=1}^N\big), \tag{3.2}$$

where expressions for $\hat{f}^{(i)}$ and $\hat{\sigma^2}^{(i)}$ are given by (2.5) and (2.6), respectively. In order for sequential design to produce an optimal sequence of points, it is necessary to have a trustworthy judgement of uncertainty; that is, we need to have faith in the model-based estimate of $\hat{\sigma}^2(\tilde{\boldsymbol{x}}|\{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t})$, which cannot be either underestimated or overestimated. Simulation experiments (section 4) show that $\hat{\sigma}^2$ can be poorly estimated by a stationary GP when f presents local features, thus requiring extrinsic diagnostics (e.g., examination of standardized residuals) to help towards the selection of future inputs [8]. To avoid any ad hoc procedures, it is necessary to rely on an adaptive emulator that can properly represent the simulator. In section 4, we compare a stationary GP to our adaptive emulator in assessing uncertainty in presence of local features.

To conclude this section, we remark that several alternative criteria for sequentially selecting new input points can be found in the literature. For instance, [22] proposes an expected improvement criterion to estimate the global minimum of a computer simulator via the maximum likelihood estimator for the emulator parameters. The ALM method that we adopt here falls into the class of "active learning" criteria, which also includes active learning Cohn (ALC) [9]. In general, ALM could be suboptimal if the response surface is heteroscedastic, and one could select points in regions of higher amplitude (or noise) and not necessarily where sharp local features are. However, we showed in section 2.2 that our model accommodates input-dependent predictive variance. [39] 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 [26]. However, the ALC criterion is more intensive to implement, therefore ALM is often preferred in practice.

4. Case studies.

4.1. Learning local features.

We considered a spatially inhomogeneous smooth function:

$$f(x) = \sin(x) + 2\exp(-30x^2), \tag{4.1}$$

which was evaluated at 15 equally spaced points in $\Omega = [-2, 2]$. For PL, we used $N = 1000$ particles initialized at time $t_0 = 4$ with a randomly selected subset of size 4 of the original 15 points. Each particle contained an estimate of the model parameters, which were initialized by sampling from their prior distributions. $\{\phi_1, \phi_2\}$ and $\tilde{\phi}_1$ were assigned log-normal prior distributions, and 0.5 and 0.25 were 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, we chose a weakly informative prior for σ^2, $\sigma^2 \sim \mathrm{IG}(2, 1)$, but flat priors could be used, and particularly the reference prior [6] would retain conjugacy.

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 results obtained with Bayesian TGP [17] and composite GP (CGP) [3]. To implement TGP, we used R package TGP available from CRAN [15]. Following

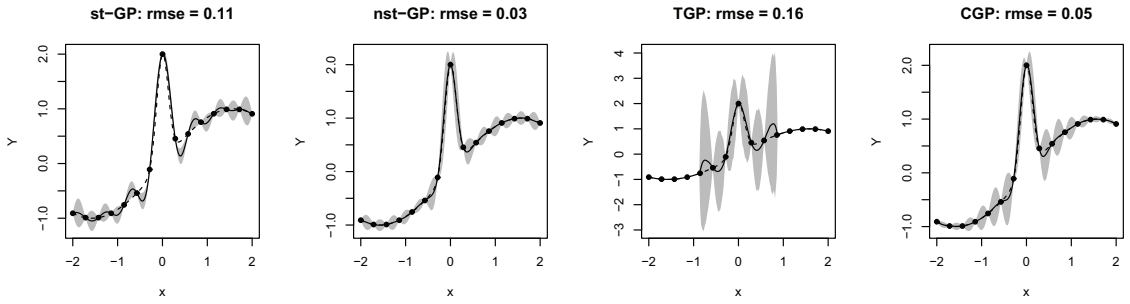

Figure 2. *Comparison between stationary GP (first panel), nonstationary GP via latent input augmentation (second panel), TGP (third panel), and CGP (fourth panel). The dashed line corresponds to the true function* (4.1)*, the solid black line is the posterior mean predictive curve, and grey areas denote the error bars. Estimates (and rmse) are obtained at* 200 *equally spaced test points.*

a referee's remark, we modified the default specifications to allow a split when both leaves have two points only. The limitations resulting from fitting a stationary GP to function (4.1) were outlined in section 1. In comparison, both our emulator and CGP (panels 2 and 3 in Figure 2) give significantly improved performance; i.e., the spline tension effect is eliminated or strongly attenuated. TGP partitions the input space into three regions. The fit is very good in the first and third partition, but the spline tension effect with large error bars remains at and about the peak. This happens because TGP fits a stationary GP in each partition, and the discovery of the central peak creates the same spline tension effect one observes with a stationary GP emulator. The error bars obtained with our nonstationary 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. For illustrative purposes, a plot of the estimated g function obtained with our emulator is reported in Web Appendix A (Figure S1) (100151_01.pdf [local/web 1.45MB]).

4.2. Quantifying the emulator's uncertainty. The simulator f is typically expected to be within two or three standard deviations from the predictive mean $\hat{f}$ [4]. 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 at and around a particular input value suggest that the emulator is not learning the local behavior of the process [8]. Further, they indicate that the emulator is underestimating the predictive uncertainty. Ultimately, one wants to acquire an accurate knowledge of f with as few simulator runs as possible. The emulator can be used to quickly identify those regions of the input space where the simulator exhibits more variations, thus helping to determine new input configurations where the simulator should be run. However, this goal can be achieved only if the emulator's estimate of uncertainty is trustworthy. A sequential design strategy based on an unreliable estimate of uncertainty will otherwise lead to a suboptimal selection of input points.

Here we examine how model-based evaluations (Figure 2 and the first row in Figure 3) combine with extrinsic diagnostics (second row in Figure 3). For extrinsic diagnostics, we examine the leave-one-out CV standardized residuals. According to the *exploration*-driven

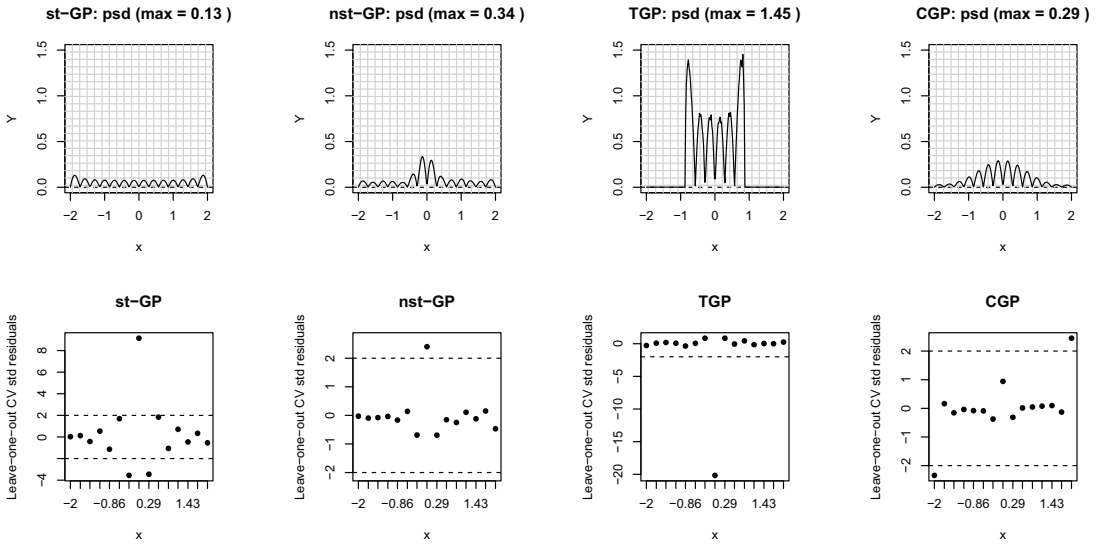

Figure 3. *Estimated standard deviation at* 200 *predictive locations (top panels) and leave-one-out CV standardized residuals (bottom panels) for the peak function* (4.1)*: comparison between stationary GP (st-GP), nonstationary GP via latent input (nst-GP), TGP, and CGP. Top panels also report the maximum estimate of predictive standard deviation.*

predictive standard deviation of a stationary GP, one is essentially equally likely to locate the new point anywhere in $[-2, 2]$ (top left panel in Figure 3). Instead, CV captures well the misfit around 0 (*exploitation*-driven CV). Thus, model-based evaluations and extrinsic diagnostics are inconsistent, and the latter shows that uncertainty is being underestimated around the peak. Model-based evaluations and extrinsic diagnostics also fail to reconcile with CGP (top and bottom right panels in Figure 3), with the predictive standard deviation suggesting that one should choose the next input around the peak ($x = 0$), and CV suggesting that one should locate it at either boundary of the input space ($x = -2$ or $x = 2$). As for TGP, the maximum a posteriori tree corresponds to three partitions, and as a result, one obtains in the central region the same fit of a stationary GP (compare the third plot in the top row of Figure 3 to the top left plot). TGP definitely restricts the sampling region of the next most suitable point as compared to the stationary emulator, but all points remain almost uniformly likely to be selected within the central partition. Both model-based evaluations and CV for our emulator (nst-GP) identify that the next point is needed around $x = 0$.

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 achieve this balance adequately.

5. High-dimensional examples and sequential design.

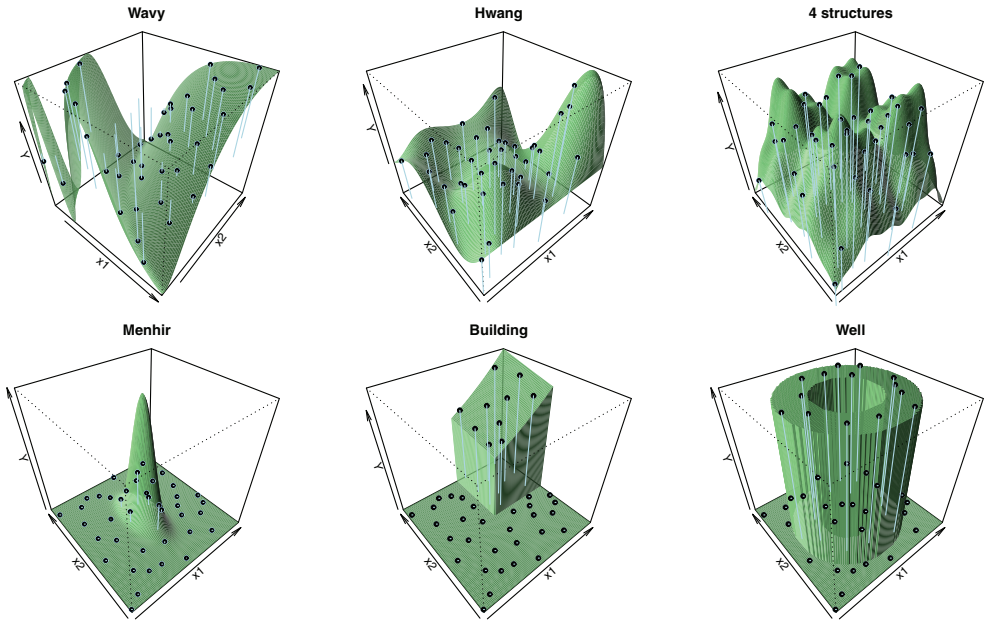

Figure 4. *True functions for the two-dimensional numerical examples and initial* 40 *observation LHD (black points) used to train the emulators.*

5.1. Two-dimensional functions with local features. In this section, we examine several test functions possessing nonstationary features (Figure 4). Most of these surfaces are continuous, whereas the last two examples are of surfaces with discontinuities. The first function (wavy) fluctuates rapidly when x_1 or x_2 is small, but gradually becomes smooth as x_1 and x_2 increase toward 1. The second function appears in [21]. The fifth function ("building") is naturally suited to TGP because of the axis-aligned nonstationarity. To shorten our presentation, we show here results obtained on two continuous (wavy and with four structures) and one discontinuous (well) functions, and defer results on the remaining examples to Web Appendix B (100151_01.pdf [local/web 1.45MB]).

We first compare the performance of the emulators when trained on a common and fixed set of input points (no sequential design). For this purpose, we use a 40 observation LHD (black points in Figure 4), which allows the emulators to gather knowledge on the overall shape of f because of its "space-filling" nature. As an alternative to LHD, one could also start with 40 random draws from the uniform distribution over the hypercube or, more generally, any other space-filling design. Figure 5 shows the posterior predictive mean surface, $\hat{f}$, and the predictive standard deviation, $\hat{\sigma}$, for wavy (first and second rows), four-structure (third and fourth rows), and well (fifth and sixth rows) functions. TGP's recursive rectangular partitioning of the input space challenges the learning of the wavy function (one can notice a ridge at $x_1 = 0.5$). Our emulator and CGP's maps in predictive standard deviations show higher uncertainty for small values of x_1 and x_2, thus calling for more points in the third quadrant of the space. No significant differences among emulators emerge on the four-structure function. As for the well function, our emulator is learning the geometry, as shown by a distinctive circular pattern

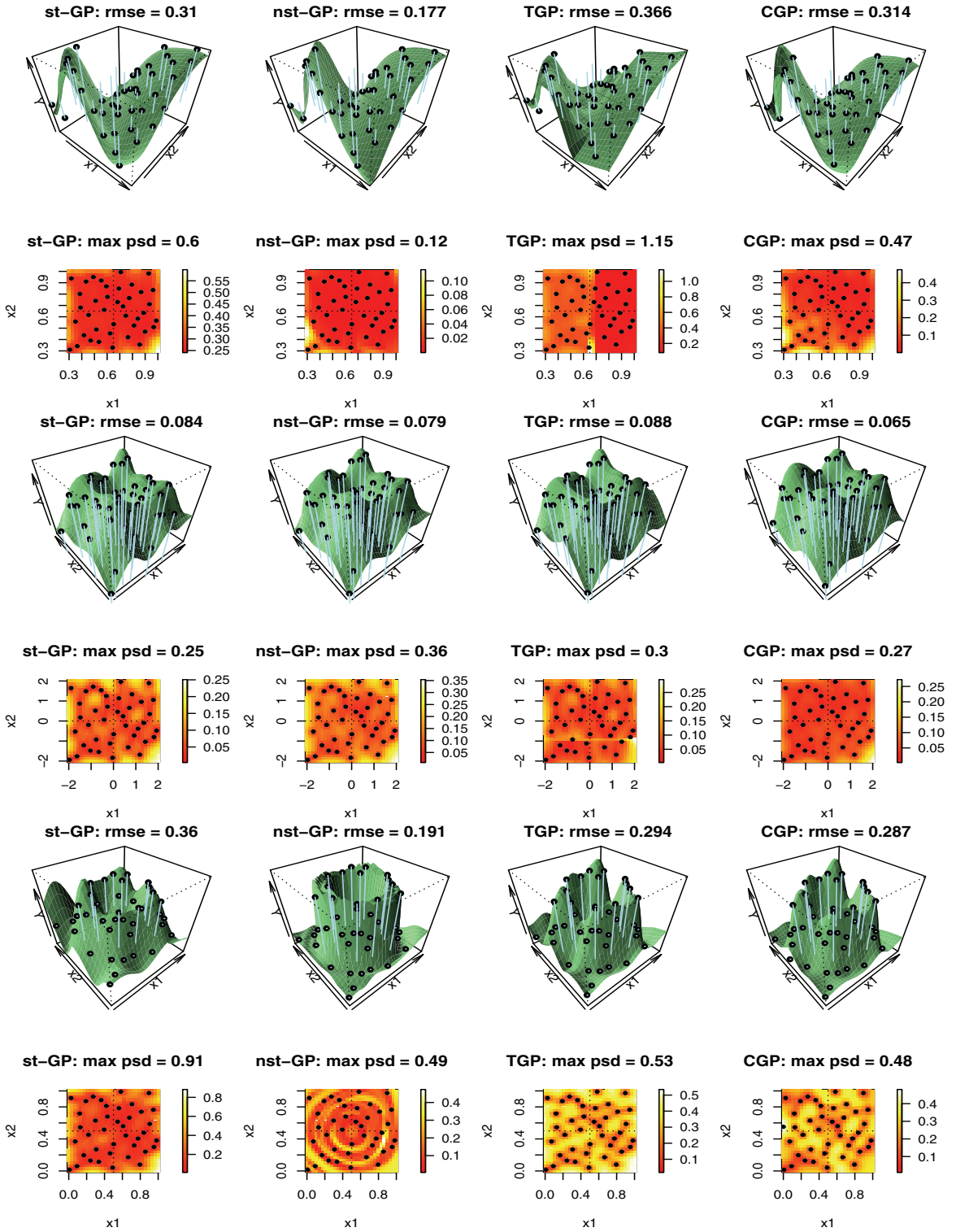

Figure 5. *Comparison between stationary GP (st-GP), nonstationary GP via latent input augmentation (nst-GP), TGP, and CGP on the two-dimensional wavy, four-structure, and well functions. The emulators are fit to a common design corresponding to a* 40 *LHD (black points). First, third, and fifth rows: posterior mean predictive surface,* $\hat{f}$*, and rmse. Second, fourth, and sixth rows: predictive standard deviation,* $\hat{\sigma}$*, and maximum predictive standard deviation (max psd). The quality of the prediction is assessed at a collection of* 900 *points, i.e., an expanded grid of* 30 *equally spaced points along each coordinate axes.*

in predictive standard deviation which is not traceable in the map produced by the other emulators.

Next, we address the issue of sequential design and 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 four-structure and well) sequentially based on their model-based estimate of uncertainty (ALM). The resulting final designs will, therefore, be different across emulators.

Figure 6 shows $\hat{f}$ and $\hat{\sigma}$ for wavy, four-structure, and well functions. For wavy and well functions, we observe that the predictive standard deviation maps obtained with our emulator inform us about the shape of the features, thus favoring the sampling of new points at and about them. Therefore, our emulator strikes a good balance between *exploration* (initial LHD) and *exploitation* (newly selected points). This behavior is not necessarily observed with the other emulators across different functions. For example, TGP and CGP sample more points where the wavy function fluctuates rapidly, but no particular sampling pattern appears on the well function. 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. No particular pattern emerges, however, across emulators on the four-structure function.

For a more quantitative numerical comparison among emulators, we calculated the rmse of predictive means to the truth on a random LHD of size 500. This was repeated 10 times, each with new LHD training (size 40, as above) and test sets. Figure 7 shows the progression of the rmse with error bars as additional inputs are being selected. Our emulator outperforms the others on wavy and well functions, but CGP is best on the four-structure function. This function is a "two-frequency" truth in that it has two global features, one that varies smoothly and one that has finer order variations. Therefore, it is an ideal example for CGP but not for our emulator, which locally behaves likes a stationary GP.

To conclude, our emulator tends to concentrate the selection of new points in interesting areas of the input space. From a comparison of several different numerical examples here and in Web Appendix B (100151_01.pdf [local/web 1.45MB]), we can conclude that no emulator is consistently the best, and the performance somehow depends on the characteristics of the function examined. Some patterns of preference emerge, though. In particular, it appears that our emulator is particularly suited to functions that change rapidly in the input space or present discontinuities. It performs as well as TGP on functions with axis-aligned nonstationarity (building), and early on, with fewer data points to train on, our emulator does in fact a better job of identifying feature localization. Also, it has the advantage over TGP of adapting better to functions whose nonstationarity is more general (wavy and well). On functions that are smoother or have two frequencies, CGP appears to be preferred. It is possible that a combination of ours and CGP's modelling assumptions and methods will perhaps be the most ideal route for real applications.

5.2. Six-dimensional examples. We consider two six-dimensional examples, which are an extension of the two-dimensional building and well functions. The six-dimensional building

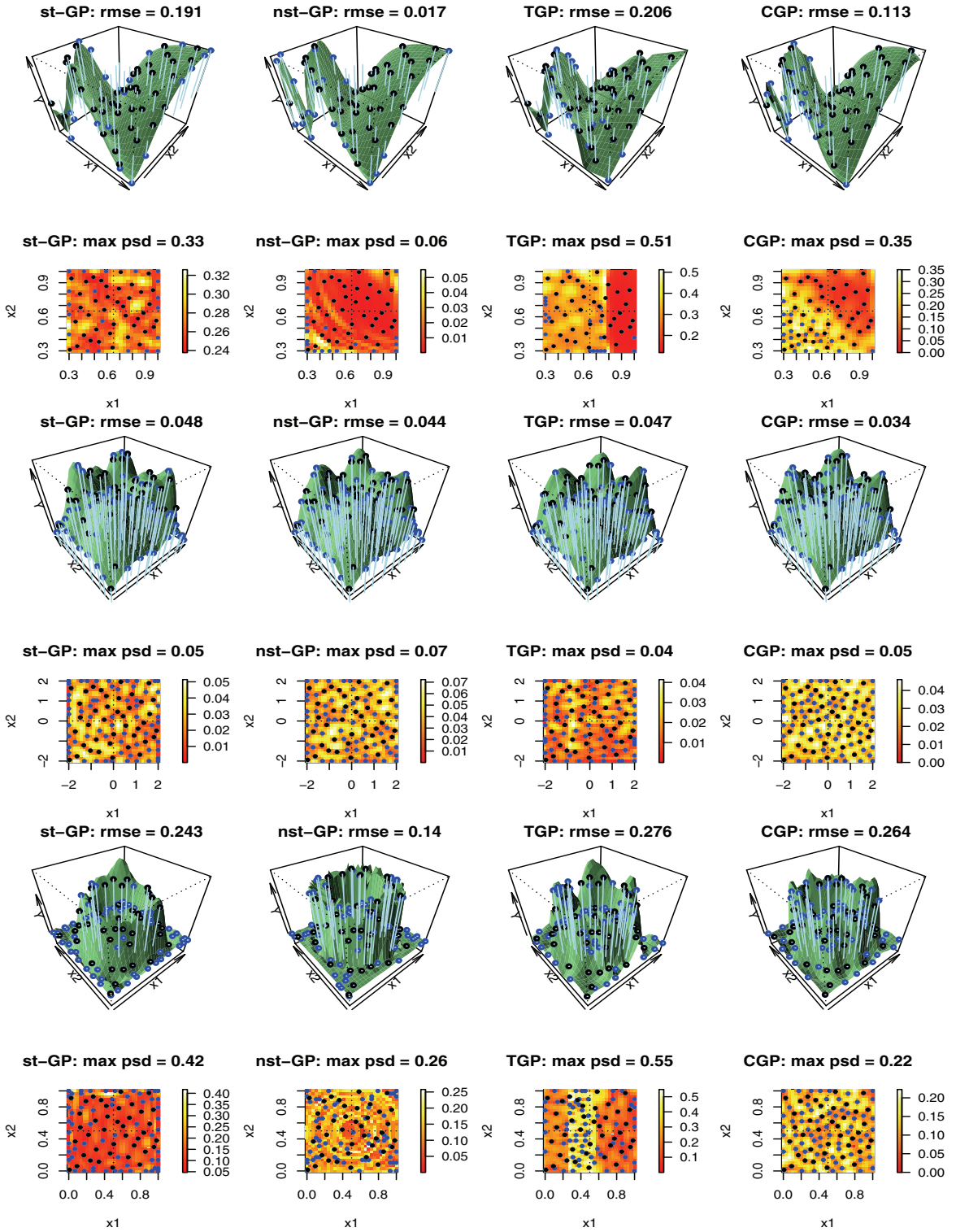

Figure 6. *Comparison between stationary GP (st-GP), nonstationary GP via latent input augmentation (nst-GP), TGP, and CGP on the two-dimensional wavy function at* $T = 60$*, and on the four-structure and well functions at* $T = 100$*. Black points: initial* 40 *LHD (common to all emulators). Blue points: additional points selected via ALM. First, third, and fifth rows: posterior mean predictive surface,* $\hat{f}$*, and rmse. Second, fourth, and sixth rows: predictive standard deviation,* $\hat{\sigma}$*, and maximum predictive standard deviation (max psd). The quality of the prediction is assessed at a collection of* 900 *points, i.e., an expanded grid of* 30 *equally spaced points along each coordinate axis.*

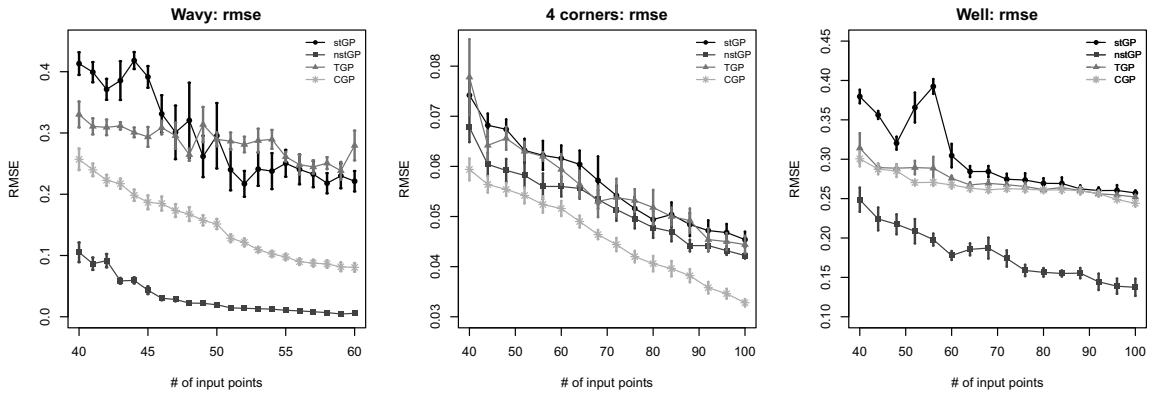

Figure 7. *Progression of the rmse as additional input points are being selected for the two-dimensional functions with error bars.*

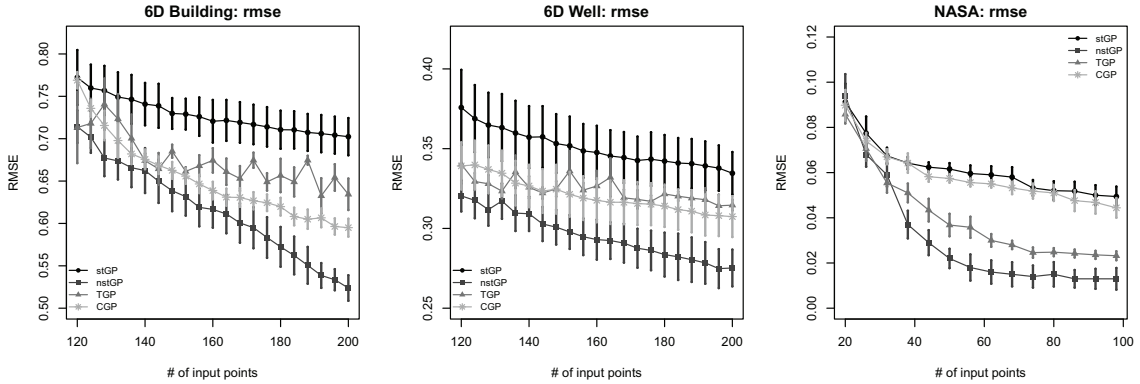

Figure 8. *Comparison among stationary GP (stGP), nonstationary GP via latent input augmentation (nstGP), TGP, and CGP in terms of progression of the rmse as additional input points are being selected. Left: six-dimensional building function (5.1). Center: six-dimensional well (5.2). Right: LGBB CFD experiment (see section 6).*

has true function

$$f(x_1,x_2,x_3,x_4,x_5,x_6) = \begin{cases} e^{\sum_{i=1}^{6}\left(\frac{1}{i}\right)^2 x_i} & \text{if } x_1,x_2,x_3,x_4,x_5,x_6 > 0.25, \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

on the hypercube $\boldsymbol{X} = [0,1]^6$. The six-dimensional well has true function

$$f(x_1,\ldots,x_6) = \begin{cases} 1 & \text{if } \sum_{i=1}^{4}(x_i-0.5)^2 > 0.025 \text{ and } \sum_{i=1}^{4}(x_i-0.5)^2 < 0.25, \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

on the hypercube $\boldsymbol{X} = [0,1]^6$. Therefore, f in (5.2) is constant in x_5 and x_6.

All emulators are trained on an identical 120-observation LHD. Emulators then select 80 additional points from a 1000-candidate LHD according to ALM. This is repeated five times, each with new LHD training and test sets. Similar to the two-dimensional examples, our emulator outperforms the others in terms of reduction of rmse (left and central panels in Figure 8). Therefore, inactive covariates adding noise to the process, such as x_5 and x_6

for function (5.2), do not affect the performance of our emulator. Additional summaries are reported in Web Appendix C of the Supplementary Materials (100151_01.pdf [local/web 1.45MB]).

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 lift force as a function of the speed of the vehicle at reentry (measured by Mach number), the angle of attack (the alpha angle), and the sideslip angle (the beta angle). [17] remarks that the sideslip angle is quantized in the experiments, so it is run only at six particular levels. Here, we examine the lift response as a function of Mach and alpha with the sideslip angle fixed at zero. See [17] for more details on the study. The CFD simulation involved an iterative integration of systems of inviscid Euler equations, each run taking 5–20 hours on a high-end workstation available to the scientists [18].

The lift response hypersurface indeed exhibits quite a bit of heterogeneity and local features. It is smooth over vast parts of the input space, but has an interesting ridge-like structure near the Mach $= 1$ hyperplane and a series of wrinkles along the angle of attack (refer to Web Appendix D of the Supplementary Materials (100151_01.pdf [local/web 1.45MB]) for a plot of lift as function of Mach and alpha). The ridge in response at Mach $= 1$ separates subsonic flows and supersonic flows.

We obtain an interpolation onto a 30×30 grid over Mach and alpha, and use the interpolated lift as our truth. All emulators are trained on a fixed and common initial design given by 20 randomly selected points from a 30×30 grid (Mach $\in [0, 6]$, alpha $\in [-5, 30]$), then select 80 additional points via ALM. Figure 9 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 all nonstationary emulators, which tend to select new points at small Mach number, and our emulator and CGP do so particularly for large alpha. A stationary GP focuses mostly on a uniform exploration of the space and will require ad hoc extrinsic diagnostics to focus around the ridge.

We run ten replicates randomizing over the initial designs (20 points randomly selected from the original dataset) and candidate sets (400 points randomly selected from the original dataset at each replicate). The third panel in Figure 8 shows the progression of rmse with error bars. Our emulator performs better than TGP on a surface that favors the latter because of the axis-aligned local feature, and improves the accuracy over stationary GP and CGP at $T = 100$.

7. Discussion. In this work we describe a nonstationary GP model to be used as an emulator in the sequential design of computer experiments. Our modelling approach is a hybrid between, and extension of, the latent variable modelling of [7] and sequential MC modelling of stationary GPs of [20] to nonstationary GP emulation. To accommodate sequential design, we rely on active learning heuristics [39]. 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, thus producing better designs. Further, it adapts well to both cases of axis-aligned nonstationarity and in situations where

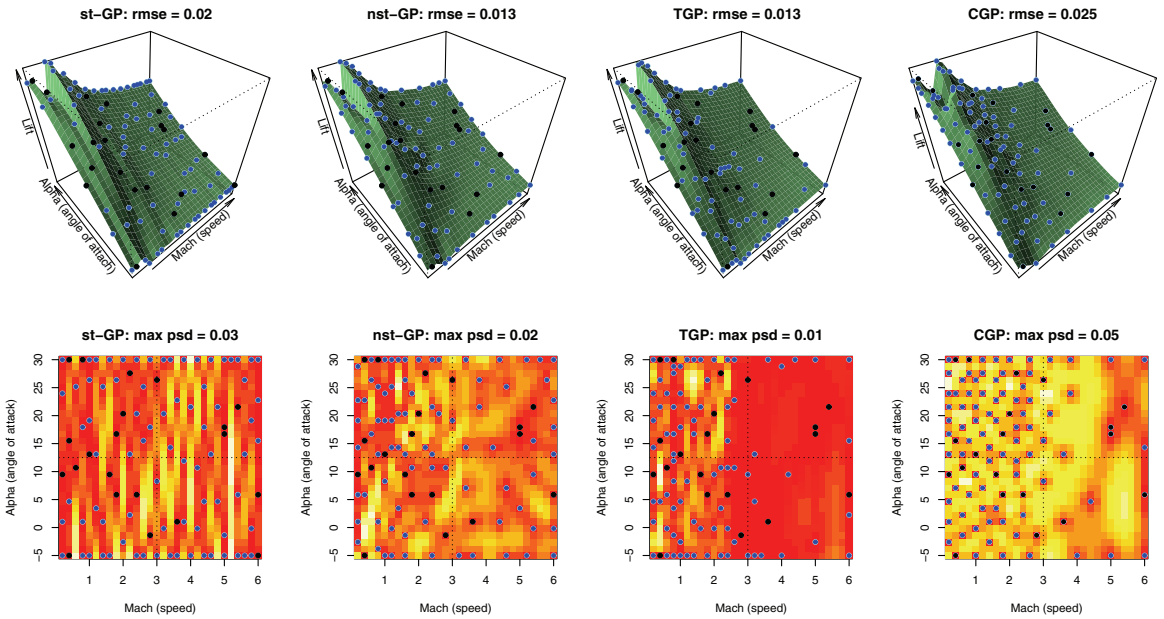

Figure 9. *Comparison between stationary GP (st-GP), nonstationary GP via latent input augmentation (nst-GP), TGP, and CGP on the LGBB experiment with lift response as a function of Mach (speed) and alpha (angle of attack) with beta (sideslip angle) fixed at* 0. *Black points:* 20 *points used as initial design (common to all emulators). Blue points:* 80 *additional points selected via ALM. Top row: posterior mean predictive surface,* $\hat{f}$*, and rmse. Bottom row: predictive standard deviation,* $\hat{\sigma}$*, and maximum predictive standard deviation (max psd). The quality of the prediction is assessed at a collection of* 900 *points, i.e., an expanded grid of* 30 *equally spaced points along each coordinate axis.*

the nonstationarity is more general. The methodology presented in this paper is applied to single-output codes; that is, $Y = f(\boldsymbol{x})$ is a scalar. The approach could be applied to high-dimensional computer models by emulating each output individually, but at the expense of potentially losing important information about correlations between outputs. The emulation of multioutput computer models via, e.g., a q-dimensional GP emulator is an interesting topic to be investigated in future research. 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 nonstationary modelling in the analysis of social, biological, and ecological data collected over spatial domains. The extension to nonparametric regression is straightforward [38, 32, 24, 30].

We conclude this section with remarks on some important aspects for computer emulation. To run all the examples included in this paper, we added a nugget α to the diagonal of the correlation functions for f and g. Many authors do not include a nugget term, on the grounds that computer codes are deterministic. In fact, the nugget introduces a measurement error and assigns nonzero uncertainty to the design data. However, it is not uncommon practice to include a nugget to enhance the numerical stability in factorizing covariance matrices [17, 2]. Although very small, α can have a nonnegligible impact on the estimates. For example, it compromises interpolation of the stationary GP on the Menhir function (Figure S3 in Web Appendix B (100151_01.pdf [local/web 1.45MB])). However, the nugget did not seem to

significantly affect the estimates of our nonstationary emulator. As an alternative to fixing the nugget to an arbitrary small value, recent literature [19] shows via empirical examples that estimating the (nonzero) nugget can improve the fit of stationary GP emulators (e.g., yielding better predictive accuracy and coverage). Finally, methods from image processing, such as curvelets or compressed sensing, are worth exploring as an alternative to GP models whenever the simulator output presents discontinuities. In particular, compressed sensing has been successfully employed in higher-dimensional settings; see, for example, [28].

REFERENCES

[1] E. B. Anderes and M. L. Stein, *Local likelihood estimation for nonstationary random fields*, J. Multivariate Anal., 102 (2011), pp. 506–520.

[2] I. Andrianakis and P. G. Challenor, *The effect of the nugget on Gaussian process emulators of computer models*, Comput. Statist. Data Anal., 56 (2012), pp. 4215–4228.

[3] S. Ba and R. Joseph, *Composite Gaussian process models for emulating expensive functions*, Ann. Appl. Statist., 6 (2012), pp. 1838–1860.

[4] L. S. Bastos and A. O'Hagan, *Diagnostics for Gaussian process emulators*, Technometrics, 51 (2009), pp. 425–438.

[5] M. J. Bayarri, J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu, *A framework for validation of computer models*, Technometrics, 49 (2007), pp. 138–154.

[6] J. Berger, V. De Oliveira, and B. Sansó, *Objective Bayesian analysis of spatially correlated data*, J. Amer. Statist. Assoc., 96 (2001), pp. 1361–1374.

[7] L. Bornn, G. Shaddick, and J. V. Zidek, *Modeling nonstationary processes through dimension expansion*, J. Amer. Statist. Assoc., 107 (2012), pp. 281–289.

[8] D. Busby, *Hierarchical adaptive experimental design for Gaussian process emulators*, Reliab. Eng. System Safety, 94 (2009), pp. 1183–1193.

[9] D. A. Cohn, *Neural network exploration using optimal experiment design*, Neural Networks, 9 (1996), pp. 1071–1083.

[10] 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.

[11] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, *Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets*, technical report, University of Minnesota, Minneapolis, MN, 2014.

[12] Y. Fan, I. Ginis, T. Hara, C. W. Wright, and E. J. Walsh, *Numerical simulations and observations of surface wave fields under an extreme tropical cyclone*, J. Phys. Oceanogr., 39 (2009), pp. 2097–2116.

[13] M. N. Gibbs, *Bayesian Gaussian Processes for Classification and Regression*, Ph.D. thesis, Inferential Sciences Group of the Cavendish Laboratory, University of Cambridge, UK, 1997.

[14] W. R. Gilks and C. Berzuini, *Following a moving target: Monte Carlo inference for dynamic Bayesian models*, J. Roy. Statist. Soc. Ser. B, 63 (2001), pp. 127–146.

[15] R. B. Gramacy, *tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and resign by treed Gaussian process models*, J. Statist. Softw., 19 (2007), pp. 1–46.

[16] R. B. Gramacy and D. W. Apley, *Local Gaussian process approximation for large computer experiments*, J. Comput. Graph. Statist., 24 (2015), pp. 561–578.

[17] R. B. Gramacy and H. K. H. Lee, *Bayesian treed Gaussian process models with an application to computer modeling*, J. Amer. Statist. Assoc., 103 (2008), pp. 1119–1130.

[18] R. B. Gramacy and H. K. H. Lee, *Adaptive design and analysis of supercomputer experiments*, Technometrics, 51 (2009), pp. 130–145.

[19] R. B. Gramacy and H. K. H. Lee, *Cases for the nugget in modeling computer experiments*, Statist. Comput., 22 (2012), pp. 713–722.

[20] R. B. Gramacy and N. G. Polson, *Particle learning of Gaussian process models for sequential design and optimization*, J. Comput. Graph. Statist., 20 (2011), pp. 102–118.

[21] J. N. Hwang, S. R. Lay, M. Maechler, R. D. Martin, and J. Schimert, *Regression modeling in back-propagation and projection pursuit learning*, IEEE Trans. Neural Netw., 5 (1994), pp. 342–353.
[22] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, J. Global Optim., 13 (1998), pp. 455–492.
[23] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, J. Roy. Statist. Ser. Soc. B, 63 (2001), pp. 425–464.
[24] H.-M. Kim, B. K. Mallick, and C. C. Holmes, *Analyzing nonstationary spatial data using piecewise Gaussian processes*, J. Amer. Statist. Assoc., 100 (2005), pp. 653–668.
[25] H. F. Lopes, C. M. Carvalho, M. S. Johannes, and N. G. Polson, *Particle Learning for Sequential Bayesian Computation*, Oxford University Press, London, 2011.
[26] D. J. C. MacKay, *Information-based objective functions for active data selection*, Neural Comput., 4 (1992), pp. 590–604.
[27] I. Murray, R. P. Adams, and D. J. C. MacKay, *Elliptical slice sampling*, J. Mach. Learning Res., 9 (2010), pp. 541–548.
[28] L. J. Nelson, F. Zhou, G. L. W. Hart, and V. Ozoliņš, *Compressive sensing as a new paradigm in model building*, Phys. Rev. B, 87 (2013), 035125.
[29] C. J. Paciorek and M. J. Schervish, *Nonstationary covariance functions for Gaussian process regression*, in Proceedings of the Conference on Neural Information Processing Systems (NIPS), Vol. 16, MIT Press, Cambridge, MA, 2004, pp. 273–280.
[30] C. J. Paciorek and M. J. Schervish, *Spatial modelling using a new class of nonstationary covariance functions*, Environmetrics, 17 (2006), pp. 483–506.
[31] T. Pfingsten, M. Kuss, and C. E. Rasmussen, *Nonstationary Gaussian Process Regression Using a Latent Extension of the Input Space*, in preparation, 2006.
[32] C. E. Rasmussen and Z. Ghahramani, *Infinite mixtures of Gaussian process experts*, in Advances in Neural Information Processing Systems, Vol. 14, MIT Press, Cambridge, MA, 2001, pp. 881–888.
[33] G. Ridgeway and D. Madigan, *A sequential Monte Carlo method for Bayesian analysis of massive datasets*, J. Knowl. Disc. and Data Min., 7 (2003), pp. 301–319.
[34] 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.
[35] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–423.
[36] P. D. Sampson and P. Guttorp, *Nonparametric estimation of nonstationary spatial covariance structure*, J. Amer. Statist. Assoc., 87 (1992), pp. 108–119.
[37] L. Schade and K. Emanuel, *The ocean's effect on the intensity of tropical cyclones: Results from a simple coupled atmosphere-ocean model*, J. Atmos. Sci., 56 (1999), pp. 642–651.
[38] A. M. Schmidt and A. O'Hagan, *Bayesian inference for nonstationary spatial covariance structure via spatial deformations*, J. Roy. Statist. Soc. Ser. B, 65 (2000), pp. 745–758.
[39] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, *Gaussian process regression: Active data selection and test point rejection*, in Proceedings of the International Joint Conference on Neural Networks (IJCNN), Vol. 3, IEEE Press, Piscataway, NJ, 2000, pp. 241–246.
[40] 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. Geophys., 48 (2009), pp. 817–842.
[41] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, J. Roy. Statist. Soc. Ser. B, 50 (1988), pp. 297–312.
[42] B. J. Williams, T. J. Santner, and W. I. Notz, *Sequential design of computer experiments to minimize integrated response functions*, Statist. Sinica, 10 (2000), pp. 1133–1152.