(Article begins on next page)

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 minimise 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 [29, 9, 32]. 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 $Y = f(\boldsymbol{x})$ 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" [26, 15, 28]. 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 $\boldsymbol{x}^t$ [26, 6].

For Bayesian emulation, a common practice is to assign f a Gaussian process prior [26, 8, 30]. 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 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 minimise the uncertainty in posterior surface estimation.

Research on computer emulation has largely focused on stationary GP models [26, 15]. 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 localised 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 standard deviation would favour the selection of a new input from the whole x domain, with the only exception of the tall peak (right panel in Figure 1). Thus, stationary GPs favour 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 [4, 3]. For example, one can examine the leave-one-out cross validated (CV) standardised residuals to quantify the emulator's uncertainty. Either too large or very small CV standardised 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 behaviour 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 [6].

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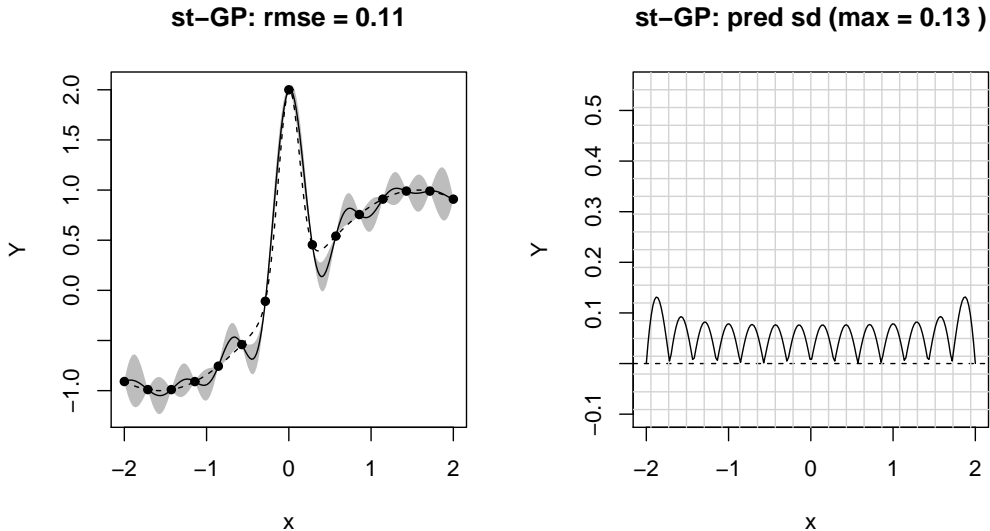
FIGURE 1. Plot of (true) function $f(x) = \sin(x) + 2\exp(-30x^2), x \in [-2, 2]$ (dashed line). The black dots represent observed data at 15 equally-spaced values of x. Left panel: the solid line is the point predictor of f, or conditional mean, obtained from a stationary GP emulator fitted to the data.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

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 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 modelling local details. Other approaches in the context of GP regression include [27, 30, 20]. This literature makes it clear that the main challenges in non-stationary GP modelling 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.

This paper is designed to serve two purposes. First, to introduce a non-stationary GP emulator which is adaptable to local features of many kinds of shape. Second, to use our emulator for online learning of an optimal sequence of design points. 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 characterised by abrupt changes of the function values and help correct for inadequacies in the fit. For sequential design, 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, when performance is measured by the

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 modelled 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.

The remainder of the paper is outlined as follows. Section 2 begins with an overview of GP emulation and stationarity and then introduces our non-stationary 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 one-dimensional numerical examples. 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 $\mathbb{C}\mathrm{ov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ parameterised by $\boldsymbol{\theta}$. Note that we can write the GP emulator as

$$(1) \qquad f(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \epsilon(\boldsymbol{x}; \boldsymbol{\theta}),$$

where $\epsilon(\boldsymbol{x}; \boldsymbol{\theta})$ is a zero-mean GP with covariance function $C_{\boldsymbol{\theta}}(\cdot, \cdot)$. To simplify the notations, 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

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

and variance

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

where $k^\top(\tilde{\boldsymbol{x}})$ is the $t-$vector whose i-th 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 (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 $\mathcal{X}$, thereby we express a belief about the smoothness of f. Although different formulations

are possible [25], in this work we focus on the power family and use the separable power correlation function

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\sum_{l=1}^{p} \phi_l (x_l - x_l')^{p_0}}, \tag{2}$$

which is a standard choice in modelling computer experiments [28]. We fix $p_0 = 2$ (product-Gaussian correlation) and infer the correlation range parameters $\{\phi_l\}_{l=1}^{p}$ as part of our estimation procedure. Thus, the correlation is only function of $\boldsymbol{x} - \boldsymbol{x}'$ (stationarity) and a set of roughness (unknown) parameters. [33] show that the squared-exponential kernel (2) 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.

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, $\sigma^2 \sim \text{IG}(a/2, b/2)$, and a log-normal prior for the correlation parameters, $\phi_l \sim \log \text{N}(\mu_\phi, \nu_\phi)$, but other formulations are possible [11]. The posterior predictive distribution of f at a new input $\tilde{\boldsymbol{x}}$, conditioned on $\boldsymbol{K}$ and data observed up to time t and marginalised with respect to $\{\boldsymbol{\beta}, \sigma^2\}$, is a Student-$t$ distribution with $\hat{\nu} = t - p - 1$ degrees of freedom, mean

$$\hat{f}(\tilde{\boldsymbol{x}} | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{K}) = h(\tilde{\boldsymbol{x}})^\top \tilde{\boldsymbol{\beta}} + k^\top (\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} (\boldsymbol{F} - \boldsymbol{H}^t \tilde{\boldsymbol{\beta}}), \tag{3}$$

and variance

$$\hat{\sigma}^2 (\tilde{\boldsymbol{x}} | \{\boldsymbol{x}, f(\boldsymbol{x})\}_{1:t}, \boldsymbol{K}) = \frac{(b + \Phi)[K(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - k^\top (\tilde{\boldsymbol{x}}) \boldsymbol{K}^{-1} k(\tilde{\boldsymbol{x}})]}{a + \hat{\nu}}, \tag{4}$$

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

$$\Phi = \boldsymbol{F}^\top \boldsymbol{K}^{-1} \boldsymbol{F} - \tilde{\boldsymbol{\beta}}^\top \Psi^{-1} \tilde{\boldsymbol{\beta}}$$

$$\tilde{\boldsymbol{\beta}} = \Psi (\boldsymbol{H}^t \boldsymbol{K}^{-1} \boldsymbol{F})$$

$$\Psi = (\boldsymbol{H}^t \boldsymbol{K}^{-1} \boldsymbol{H}^t)^{-1}.$$

The availability in closed form of the marginalised predictive distribution is crucial for the sequential design algorithm implementing our non-stationary GP emulator (§3).

2.2. **Non-stationary 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 [11]. In response to concerns about the adequacy of the stationary assumption for GP emulators, we build on the concept of spatial deformation [27] and model $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \epsilon([\boldsymbol{x}, Z]; \boldsymbol{\theta}), \tag{5}$$

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" product-Gaussian correlation form for $K = \sigma^{-2} C$:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp \left\{ -\sum_{l=1}^{p} \phi_l (x_{il} - x_{jl})^2 - \phi_{p+1} (Z_i - Z_j)^2 \right\}. \tag{6}$$

5

Expression (6) corresponds to the standard (squared-exponential) correlation function of a stationary GP (2) indexed by $p+1$ inputs.

Clearly, the problem of modelling a non-stationary simulator could be tackled differently. For example, [25] propose gathering substantial information about the simulator from experts and use 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 non-trivial 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 sub-optimal sequential design strategy in presence of local features. [20] generalise Gibbs' construction to obtain non-stationary 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 characterised 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 localised 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

(7)
$$g \mid \boldsymbol{\theta} \sim \mathrm{GP}(0, \tilde{K}), \qquad \text{with}$$

(8)
$$\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{ -\sum_{l=1}^{p} \tilde{\phi}_l (x_{il} - x_{jl})^2 \right\},$$

where the scale parameter is fixed to 1.

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

(9)
$$f|\boldsymbol{\theta} \sim \mathrm{GP}(\mu, C), \qquad \text{and} \qquad g|\boldsymbol{\theta} \sim \mathrm{GP}(0, \tilde{K}),$$

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 (1) to functions whose smoothness varies with the inputs because it has the capacity to have several length scales.

As mentioned above, [27] 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 realisations 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 [30]. The approach we propose in (9) is similar in flavour to the construction in [30], who build

on [27] and implement spatial deformation via a GP prior. However, our construction differs from [30], 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's only. Also, the 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 [22], who present two approaches for approximate Bayesian inference in GP regression models. The first method relies on a discrete latent input and is implemented in an Markov Chain Monte Carlo (MCMC) sampling scheme, whereas the second method estimates a continuous latent mapping by evidence maximisation. We remark that [27, 30, 20, 22] are not attempting to estimate a deterministic model or perform sequential design.

3. IMPLEMENTATION

We apply our adaptive non-stationary GP emulator to the sequential design of computer experiments [28]: start with a parsimonious design; choose a new input $\boldsymbol{x}$ according to some criterion derived from the emulator fit; update the emulator conditional on the new pair $\{\boldsymbol{x}, f(\boldsymbol{x})\}$; and repeat until some stopping criterion is met. Sequential design is crucial to keep designs small and save on expensive runs of the simulator while guaranteeing adequate learning of the input-output map. We adopt a sequential MC technique known as particle learning (PL) to obtain a quick update of the emulator after each sequential design iteration (§3.1) and discuss some sequential design criteria in §3.2. An introduction to PL is beyond the scope of this paper. The unfamiliar reader can refer to [17] for an introduction and to [13] for its application to the online updating of GP regression models.

3.1. **PL for GP emulation.** 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 to update the particles from t to $t+1$.

We start by identifying the quantities particles include. The sufficient information necessarily depends upon $[(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_t, f(\boldsymbol{x}_t))]$ [1], thus $\{S_t^{(i)}\}_{i=1}^N = \{(Z_{1:t}, K_t, \tilde{K}_t)^{(i)}\}$, with $Z_{1:t} \equiv (Z_1, \ldots, Z_t)^T$. The correlation functions have been indexed by t to stress their dependence to data collected up to time t. Particles do not contain $\boldsymbol{\beta}$ nor σ^2 as these parameters can be marginalised out within our Bayesian construction [13].

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). Particles are initialised at time t_0 with a sample of the unknown parameters from their prior distributions. In addition to the priors in §2.1, we also sample $\{\tilde{\phi}_l\}_{l=1}^p$ from $\tilde{\phi}_l \sim \log N(m_{\tilde{\phi}}, v_{\tilde{\phi}})$. The core components of PL for updating particles $\{S_t^{(i)}\}_{i=1}^N$ to $\{S_{t+1}^{(i)}\}_{i=1}^N$ are the following two steps:

---

[1]To stress the dependence of f to 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.

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

$$w^{(i)} = \frac{\pi(f(\boldsymbol{x}_{t+1})|S_t^{(i)})}{\sum_{i=1}^{N} \pi(f(\boldsymbol{x}_{t+1})|S_t^{(i)})}, \qquad i = 1, \ldots, N,$$

  where $\pi(f(\boldsymbol{x}_{t+1})|S_t^{(i)}) = \pi(f(\boldsymbol{x}_{t+1})|[\boldsymbol{x}, f(\boldsymbol{x})]_{1:t}, K_t^{(i)})$ denotes the probability of observing $f(\boldsymbol{x}_{t+1})$ under a Student-t distribution with $\hat{\nu} = t - p - 1$ degrees of freedom and mean and variance given by Equations (3)–(4), respectively.

- *Propagate* each resampled particle $S_t^{\zeta(i)}$ to $S_{t+1}^{(i)}$ to account for $[\boldsymbol{x}_{t+1}, f(\boldsymbol{x}_{t+1})]$

  - Construct the "propagated" correlation function of the latent GP, which will be used to sample the latent coordinate at the new input $\boldsymbol{x}_{t+1}$. Thus, we build $\tilde{\boldsymbol{K}}_{t+1}^{(i)}$ from $\tilde{\boldsymbol{K}}_t^{(i)}$ and $\tilde{k}_t^{(i)}(\boldsymbol{x}_{t+1}) = \tilde{K}^{(i)}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_j)$, with $j = 1, \ldots, t$

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

  - Obtain $g^{(i)}(\boldsymbol{x}_{t+1})$ from its predictive distribution $g^{(i)}(\boldsymbol{x}_{t+1}) \mid g^{(i)}(\boldsymbol{x}_{1:t}), \tilde{\boldsymbol{K}}_{t+1}^{(i)} \sim N(\mu^{*(i)}, \tilde{K}^{*(i)})$, where mean and variance are obtained via standard kriging equations

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

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

  The three sub-steps above can be performed in parallel across particles.

The correlation 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 include a "*rejuvenate*" step which applies MCMC moves to the particles after the propagating step [10, 24]. The update is done via elliptical slice sampling [19].

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 point-wise predictive distribution for each of the particles, the posterior mean predictive curve, as our prediction of f at new inputs

$$(10) \qquad \hat{f} = \mathbb{E}(f|S^{(i)}) = \frac{1}{N} \sum_{i=1}^{N} \hat{f}^{(i)},$$

whereas the estimate for the predictive standard deviation is obtained as

$$(11) \qquad \sqrt{\hat{\sigma}^2} = \mathbb{E}(\{\hat{\sigma}^{(i)}\}_{i=1}^{N}) + \mathbb{V}\text{ar}(\{\hat{f}^{(i)}\}_{i=1}^{N}),$$

where expressions for $\hat{f}^{(i)}$ and $\hat{\sigma^2}^{(i)}$ are given by Equations (3)–(4), respectively.

Sequential MC computing seamlessly blends with active learning of the sequential design (§3.2). In addition to PL, we also investigated 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. A comparison between the two implementations is deferred to Web Appendix D of the Supplementary Materials.

3.2. **Adaptive sequential design.** The adaptive and sequential selection of input points to include in the design sits comfortably within our PL implementation. After particles have been resampled, and before proceeding with the propagation step, the algorithm performs prediction at a set of candidate input configurations based on the posterior predictive distribution. For every candidate point, 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 $\tilde{\boldsymbol{x}}$. Candidate points can be ordered based on their predictive variance and the point with largest uncertainty in predicted output is chosen as the next input $\boldsymbol{x}_{t+1}$. Consequently, particles are propagated with the new pair $[\boldsymbol{x}_{t+1}, f(\boldsymbol{x}_{t+1})]$, and the sequence is iterated until some pre-specified 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. In order for this procedure 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 can not be either underestimated or overestimated. Simulation experiments (§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 standardised residuals) to help towards the selection of future inputs. To avoid any ad-hoc procedures, it is necessary to rely on an adaptive emulator that can represent properly the simulator. In §4, we compare a stationary GP to our adaptive emulator in assessing uncertainty in presence of local features.

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]. The procedure we implement and described above corresponds to ALM. [31] compared ALM and ALC and observed that ALC often performs better than ALM. For example, the ALM criterion embedded into a stationary GP emulator favours 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 ALM is often preferred in practice.
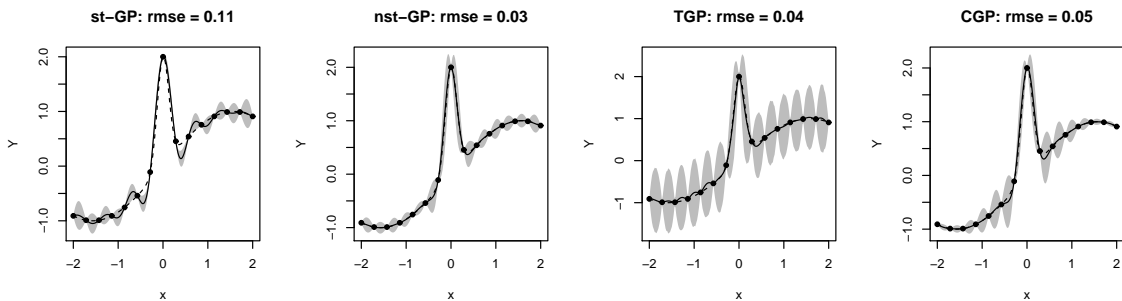
FIGURE 2. Comparison between stationary GP (first panel), non-stationary GP via latent input augmentation (second panel), TGP (third panel), and CGP (fourth panel). The dashed line corresponds to the true function (12), the solid black line is the posterior mean predictive curve, and grey areas denote the error bars. Estimates (and RMSE) are obtained at 200 equally spaced test points.

4. CASE STUDIES

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

$$(12) \qquad\qquad f(x) = \sin(x) + 2\exp(-30x^2),$$

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

For PL, we use $N = 1000$ particles initialised at time $t_0 = 4$ with a randomly selected subset of size 4 of the original 15 points. Each particle contains an estimate of the model parameters, which are initialised by sampling from their prior distributions. $\{\phi_1, \phi_2\}$ 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 σ^2, $\sigma^2 \sim \mathrm{IG}(2, 1)$.

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 results obtained with Bayesian TGP [11] and composite GP (CGP) [2]. The limitations resulting from fitting a stationary GP to function (12) 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 f is typically expected to be within two or three standard deviations from the predictive mean $\hat{f}$ [3]. While an isolated outlier might be ignored, several large standardised 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 standardised residuals systematically observed at and around a particular
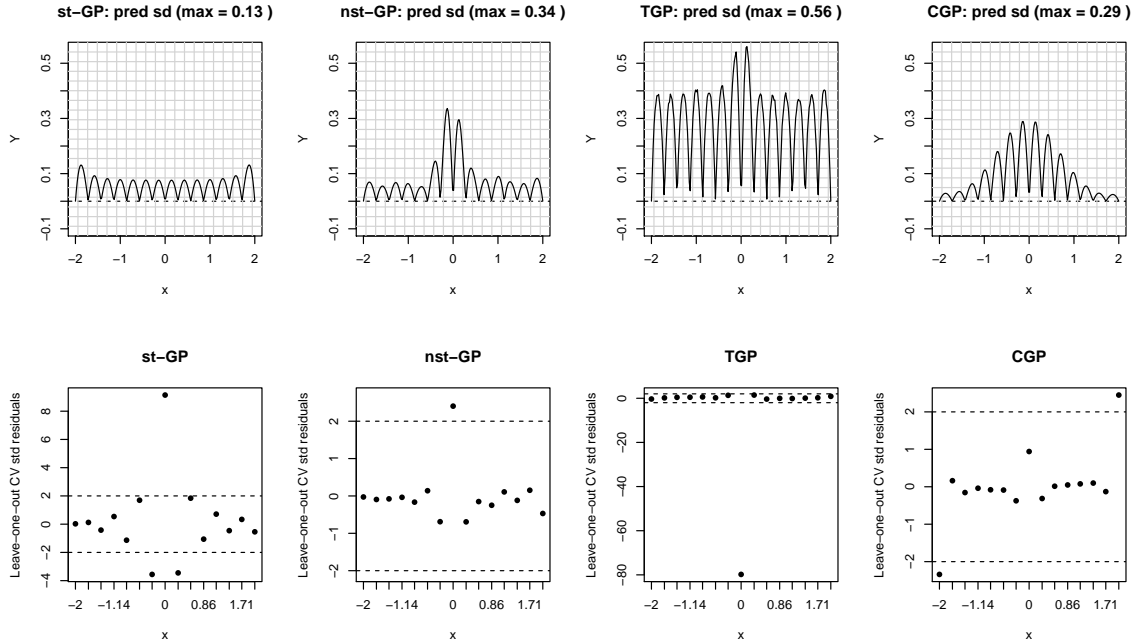
FIGURE 3. Estimated standard deviation at 200 predictive locations (top panels) and leave-one-out CV standardised residuals (bottom panels) for the peak function (12): comparison between stationary GP (st-GP), non-stationary GP via latent input (nst-GP), TGP, and CGP. Top panels also report the maximum estimate of predictive standard deviation.

input value suggest that the emulator is not learning the local behaviour 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 further runs of the simulator should concentrate. 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 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). For extrinsic diagnostics, we examine the leave-one-out cross validated standardised residuals. According to the *exploration*-driven 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 strongly favours the selection of a new input around $x = 0$ (*exploitation*-driven CV) to learn the local behaviour of f. Thus, model-based evaluations and extrinsic diagnostics are inconsistent, and the latter shows that uncertainty is being under-estimated around the peak. It is not immediately clear how to combine inconsistent evaluations when there is no knowledge of the true f, as for real data applications. 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
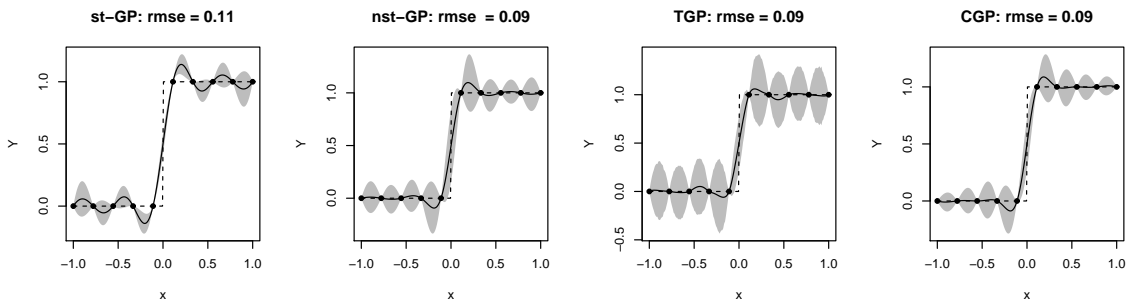
FIGURE 4. Comparison between stationary GP (st-GP), non-stationary GP via latent input augmentation (nst-GP), TGP, and CGP on the 1D discontinuous function. The dashed line corresponds to the true function (13), the solid black line is the posterior mean predictive curve, and grey areas denote the error bars. Estimates (and RMSE) are obtained at 200 equally spaced test points.

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 emulator (nst-GP) and TGP identify that the next point is needed around $x = 0$. As opposed to our emulator, the conclusion is however made much more evident by extrinsic diagnostics (a strikingly large CV standardised residual at $x = 0$) rather than by the predictive standard deviation with TGP.

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

$$(13) \qquad f(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

We evaluate (13) at 10 equally spaced points in $\Omega = [-1, 1]$. For PL, we use $N = 1000$ particles initialised at time $t_0 = 4$ with a randomly selected subset of size 4 of the original 10 points. This function is particularly suited to TGP because of the vertical, axis-aligned nature of localised feature.

Figure 4 shows that the point predictions obtained with the non-stationary emulators are not (or less) distorted by the spline tension effect as opposed to the fit obtained with a stationary GP. Further, 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 Ω.

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). Also, CV shows that the predictive uncertainty is not underestimated. 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 $x = 0$.

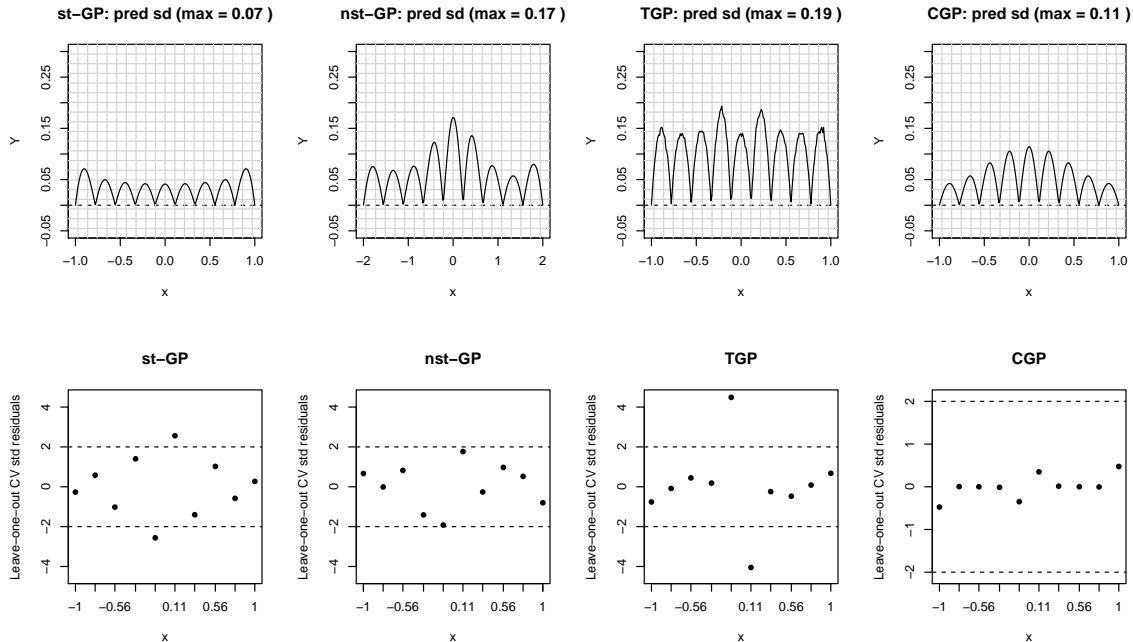In general, it is not clear how to reconcile model-based and extrinsic diagnostics whenever

FIGURE 5. Estimated standard deviation at 200 predictive locations (top panels) and leave-one-out CV standardised residuals (bottom panels) for the jump function (13): comparison between stationary GP (st-GP), non-stationary GP via latent input (nst-GP), TGP, and CGP. Top panels also report the maximum estimate of predictive standard deviation.

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.

5. HIGH-DIMENSIONAL EXAMPLES AND SEQUENTIAL DESIGN

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

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 LHD (blue points in Figure 6), which allows the emulators to gather knowledge on the overall shape of f because of its "space-filling" nature. Figure 13 in 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. The initial LHD does not include points at or nearby the peak of the menhir function (Figure 13), and this affects the estimates of the four emulators, which can not recover the central spike. Note, however, how both our emulator and CGP
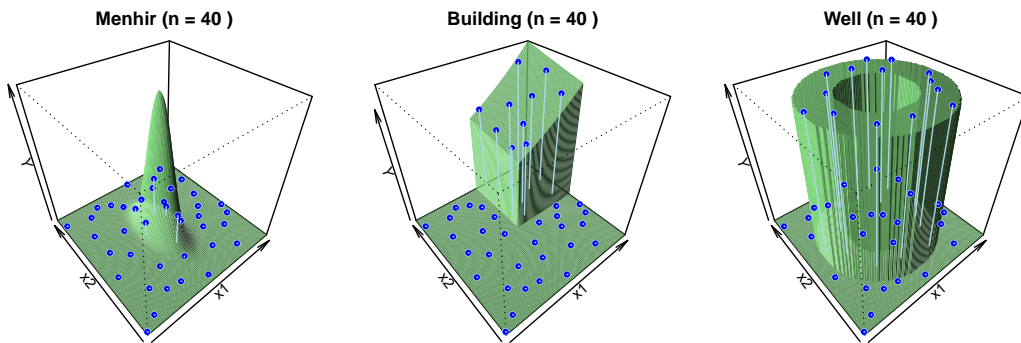
FIGURE 6. True functions for the 2-dimensional numerical examples and initial 40 LHD (blue points) used to train the emulators.

identify higher uncertainty in the central part of the input space. In particular, the pattern in predictive standard deviation seems to indicate that our emulator is learning the circular geometry of the menhir. TGP also identifies higher central uncertainty, 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, which is higher at the edges of the building (Figure 7) and has a distinctive circular pattern for the well function (8). This does not appear to be the case for 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 well) sequentially based on their model-based estimate of uncertainty (ALM). The resulting final designs will, therefore, be different across emulators.

Figures S1 and S2 in Web Appendix A of the Supplementary Materials 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 emulator favours 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 points at the center of the input space for the menhir function, but no pattern is observed for building and well functions. After ALM, TGP seems to have preferred the selection of new points at the edges of the building (Figure 9), but this behaviour is much less evident on the other functions with circular geometry. 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 emulators, Figure 10 shows the progression of the RMSE as additional inputs are being selected. Our emulator performs
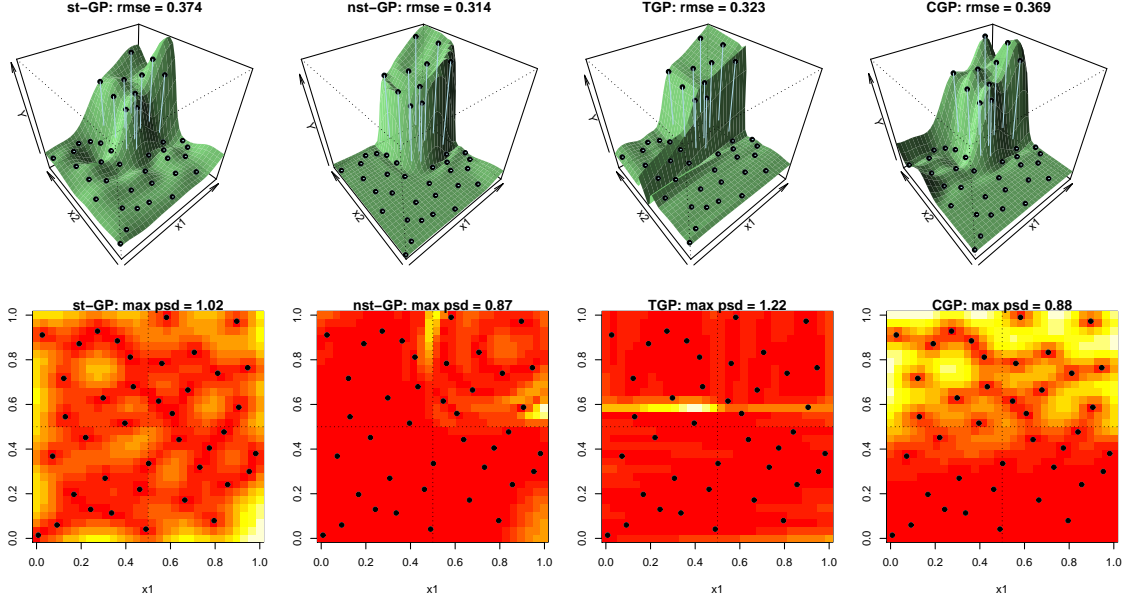
FIGURE 7. Comparison between stationary GP (st-GP), non-stationary GP via latent input augmentation (nst-GP), TGP, and CGP on the 2D building function. The emulators are fit to a common design corresponding to a 40 LHD (black points). Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0,1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes.

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 tends to concentrate the selection of new points in interesting areas of the input space. Furthermore, it compares favourably 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 are an extension of the 2D building and well functions. The 6D building has true function:

$$(14) \qquad f(x_1, x_2, x_3, x_4, x_5, x_6) = \begin{cases} e^{\sum_{i=1}^{6} \left(\frac{1}{i}\right)^2 x_i}, & \text{if } x_1, x_2, x_3, x_4, x_5, x_6 > 0.25 \\ 0, & \text{otherwise} \end{cases}$$

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

$$(15) \qquad f(x_1, \ldots, x_6) = \begin{cases} 1, & \text{if } \sum_{i=1}^{4}(x_i - 0.5)^2 > 0.025 \text{ and } \sum_{i=1}^{4}(x_i - 0.5)^2 < 0.25 \\ 0, & \text{otherwise} \end{cases}$$

on the hypercube $\boldsymbol{X} = [0,1]^6$. Therefore, f in (15) is constant in x_5 and x_6.

All emulators are trained on an identical 120 LHD. Emulators then select 80 additional points from a 1000 candidate LHD according to ALM. Similar to the 2D examples, our emulator outperforms the others in terms of reduction of RMSE (left and central panels in
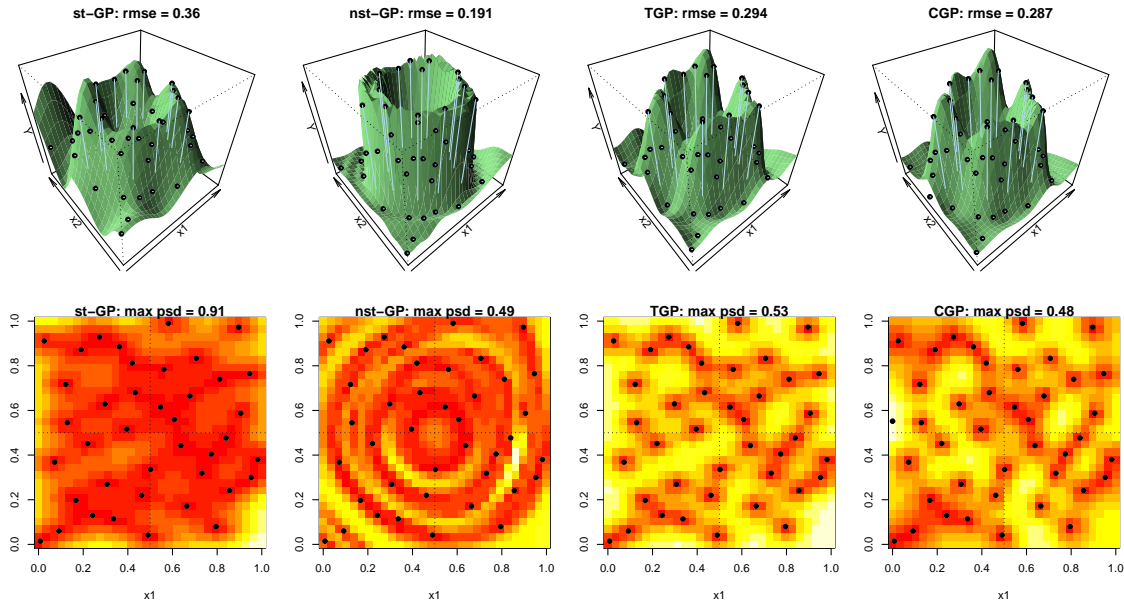
15

FIGURE 8. Comparison between stationary GP (st-GP), non-stationary GP via latent input augmentation (nst-GP), TGP, and CGP on the 2D well function. The emulators are fit to a common design corresponding to a 40 LHD (black points). Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0,1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes.

Figure 11). Therefore, inactive covariates adding noise to the process, such as x_5 and x_6 for function (15), do not affect the performance of our emulator. Additional summaries are reported in Web Appendix B of the Supplementary Materials.

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 S5 in Web Appendix C of the Supplementary Materials for a plot of the "lift" response as function of Mach and Alpha). The ridge in response at Mach equal to 1 separates subsonic flows and supersonic
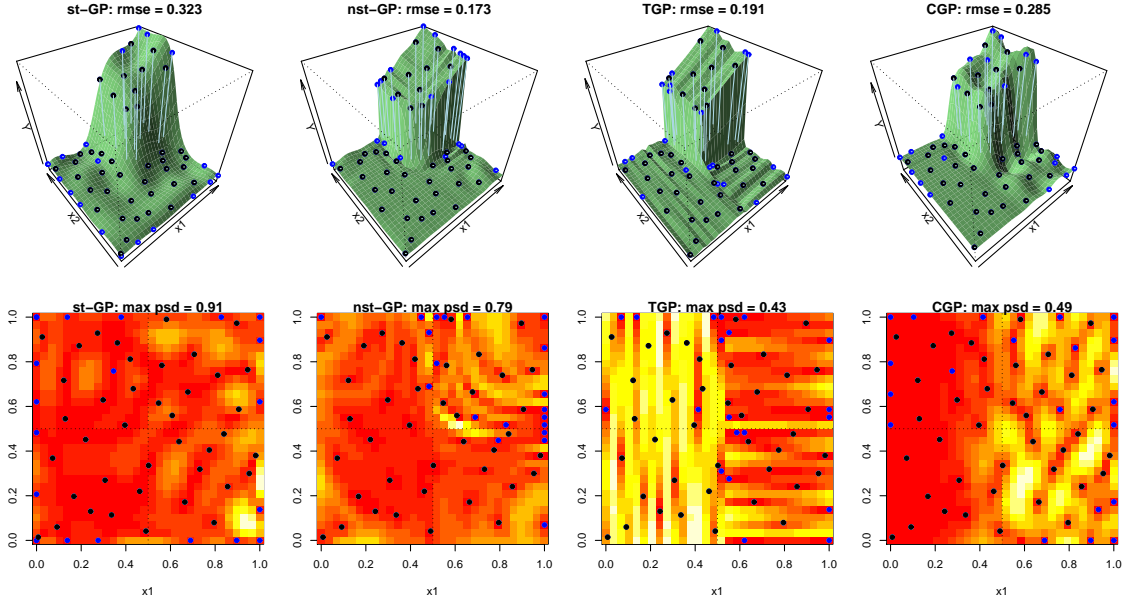
FIGURE 9. Comparison between stationary GP (st-GP) via latent input augmentation (nst-GP), TGP, and CGP on the 2D building function at $T = 60$. Black points: initial 40 LHD (common to all emulators); blue points: 20 additional points selected via ALM. Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0, 1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes.
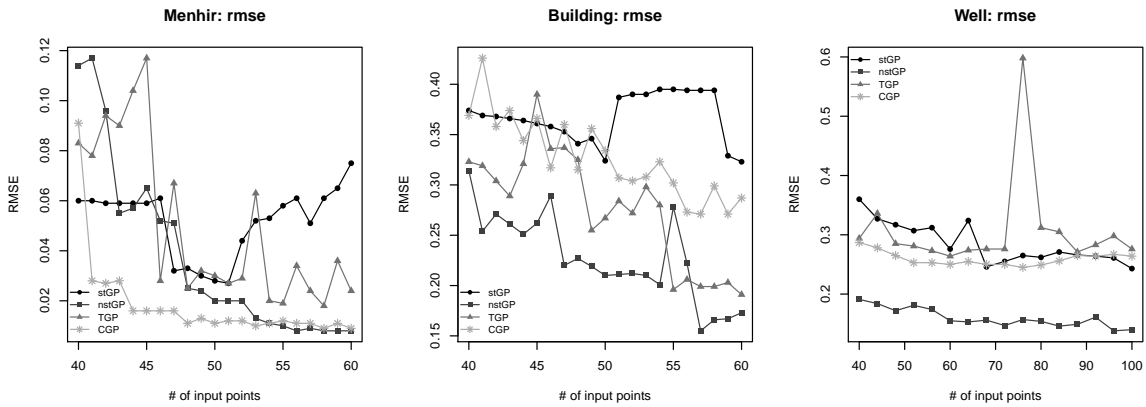
FIGURE 10. Progression of the RMSE as additional input points are being selected for the 2D functions.

flows. The behaviour 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 attach (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. All
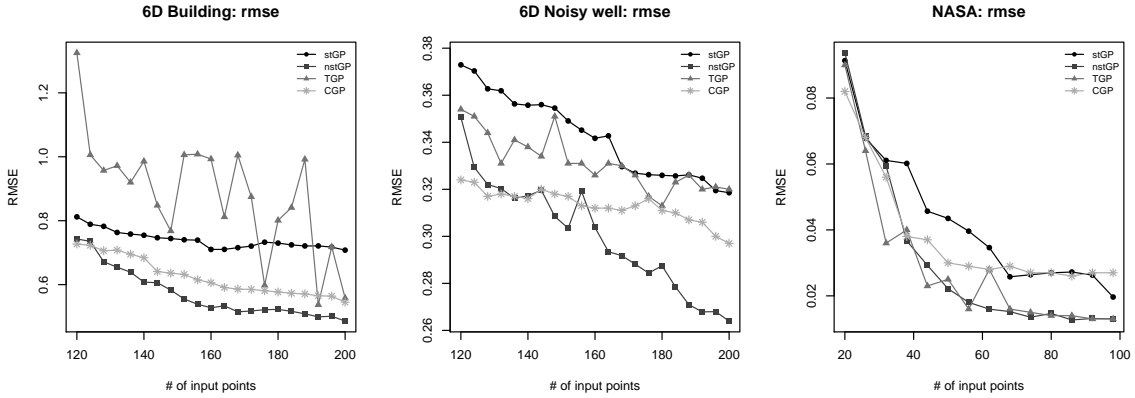
FIGURE 11. Comparison among stationary GP (stGP), non-stationary GP via latent input augmentation (nstGP), TGP, and CGP in terms of progression of the RMSE as additional input points are being selected. Left panel: 6D building function (14). Central panel: 6D well (15); Right panel: LGBB CFD experiment (Section 6).

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 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 all non-stationary emulators, which tend to select new points at small Mach, 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.

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 favours the latter because of the axis-aligned local feature, and improves the accuracy over stationary GP and CGP by 35% and 48%, respectively, at $T = 100$.

7. 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 [1]. The resulting covariance function corresponds to the case where f is observed with additive Gaussian noise with zero mean and variance α. Many
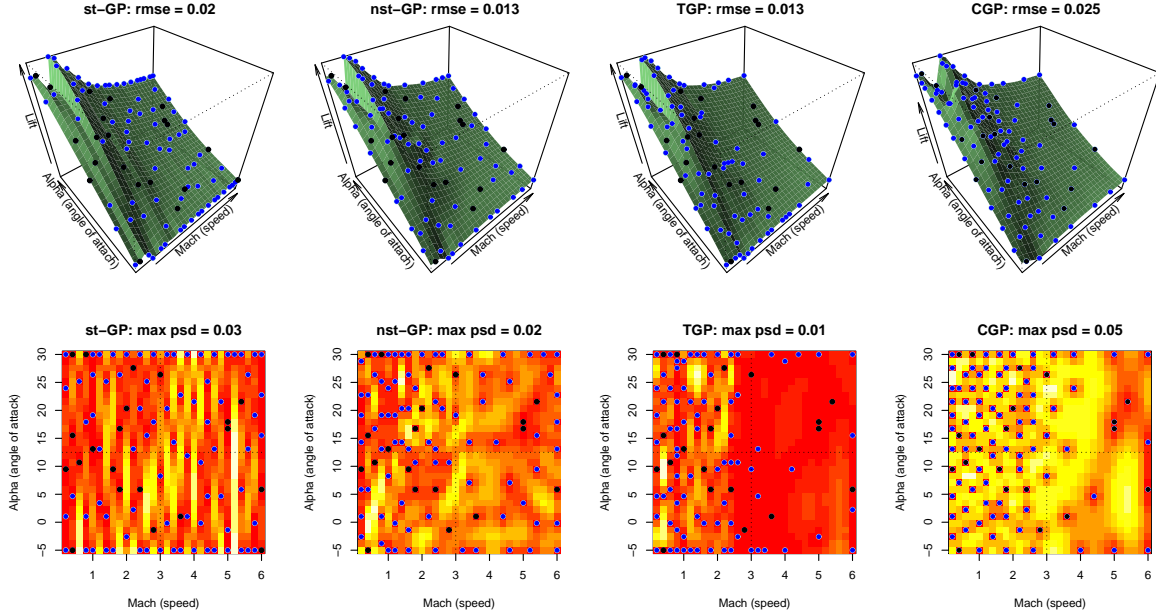
18

FIGURE 12. Comparison between stationary GP (st-GP), non-stationary 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) and Beta (side-slip 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 root mean squared error (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 axes.

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 factorising covariance matrices [11, 1]. A typical value of the nugget used in our numerical examples is $\alpha = 10^{-7}$, the effect of this being the addition of α in the predictive variance of the responses. Although very small, α can have a non-negligible impact on the estimates. For example, it compromises interpolation of the stationary GP on the menhir function (Figure S1 in Web Appendix A). However, the nugget did not seem to significantly affect the estimates of our non-stationary emulator. For more details on the inclusion of a nugget in computer emulation, refer to [1, 2].

Parallelisation. An appealing characteristic of PL is its heavy parallelisable nature: many of the typical calculations on the particles can proceed independently of one another. In particular, the evaluation of the posterior predictive distribution and the propagate step can be performed in parallel for each particle. Resampling requires the particles being synchronised, but this step is fast once the particle predictive densities have been evaluated. Our

code makes extensive use of R's function `lapply` to automatically loop over the particles to evaluate the predictive distribution and propagate the particles. More advanced alternatives such as `snowfall` and `sfCluster` for parallel programming using clusters could lead to computational improvements.

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 factorisation) is needed to handle high-dimensional problems. Independently developed research in the context on 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 modelling 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 [30, 23, 16, 21].

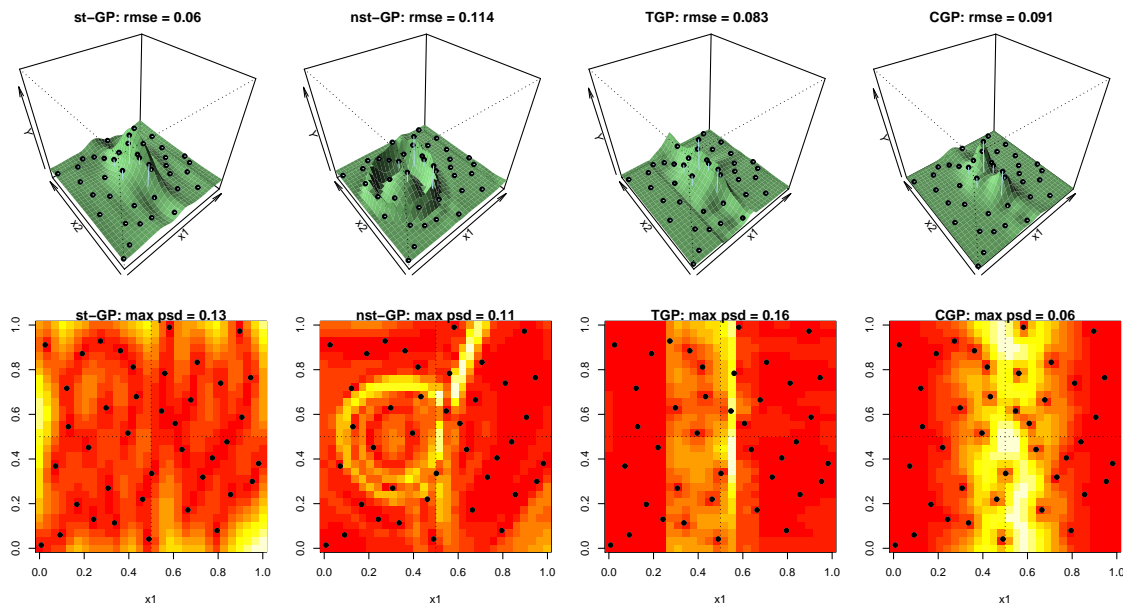## APPENDIX A. MENHIR FUNCTION ESTIMATES AT $T = 40$ (LHD)

FIGURE 13. Comparison between stationary GP (st-GP) via latent input augmentation (nst-GP), TGP, and CGP on the 2D menhir function. The emulators are fit to a common design corresponding to a 40 LHD (black points). Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0,1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes.

20

References

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

[2] S. Ba and R. Joseph, *Composite Gaussian process models for emulating expensive functions*, Annals of Applied Statistics, 6 (2012), pp. 1838–1860.

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

[4] 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.

[5] A. Bhattacharya, D. Pati, and D. B. 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 (2009), pp. 1183–1193.

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

[8] 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. Am. Statist. Assoc., 86 (1991), pp. 953–963.

[9] 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.

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

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

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

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

[14] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, J. of Global Optimization, 13 (1998), pp. 455–492.

[15] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, J. R. Stat. Soc. B, 63 (2001), pp. 425–464.

[16] H.-M. Kim, B. K. Mallick, and C. Holmes, *Analyzing nonstationary spatial data using piecewise Gaussian processes*, J. A. Stat. Assoc., 100 (2005), pp. 653–668.

[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] D. J. MacKay, *Information-based objective functions for active data selection*, Neural Computation, 4 (1992), pp. 590–604.

[19] I. Murray, R. P. Adams, and D. J. C. MacKay, *Elliptical slice sampling*, J. Mach. Learn. Res., 9 (2010), pp. 541–548.

[20] C. J. Paciorek and M. J. Schervish, *Nonstationary covariance functions for gaussian process regression*, in Proc. of the Conf. on Neural Information Processing Systems (NIPS), vol. 16, MIT Press, 2004, pp. 273–280.

[21] ——, *Spatial modelling using a new class of nonstationary covariance functions*, Environmetrics, 17 (2006), pp. 483–506.

[22] T. Pfingsten, M. Kuss, and C. E. Rasmussen, *Nonstationary Gaussian process regression using a latent extension of the input space*, in manuscript, 2006.

[23] C. E. Rasmussen and Z. Ghahramani, *Infinite mixtures of Gaussian process experts*, in Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2001, pp. 881–888.

[24] G. Ridgeway and D. Madigan, *A sequential Monte Carlo method for Bayesian analysis of massive datasets*, J. Knowl. Disco. and Data Min., 7 (2003), pp. 301–319.

[25] 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.

[26] 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.

[27] P. D. Sampson and P. Guttorp, *Nonparametric estimation of nonstationary spatial covariance structure*, J. A. Stat. Assoc., 87 (1992), pp. 108–119.

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

[29] 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.

[30] A. M. Schmidt and A. O'Hagan, *Bayesian inference for nonstationary spatial covariance structure via spatial deformations*, J. R. Stat. Soc., B, 65 (2000), pp. 745–758.

[31] 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, 2000, pp. 241–246.

[32] 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 (2009), pp. 817–842.

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

Dept. of Statistics, University of Warwick, Coventry CV4 7AL, UK
E-mail address: S.Montagna@warwick.ac.uk

Dept. of Statistical Science, Duke University, Box 90251, Durham, NC, USA
E-mail address: st118@stat.duke.edu

Supplementary Materials for "Computer emulation with non-stationary Gaussian processes" by Silvia Montagna and Surya T. Tokdar

SILVIA MONTAGNA

Department of Statistics, University of Warwick, Coventry, CV4 7AL, UK

SURYA T. TOKDAR

Department of Statistical Science, Duke University, Durham, NC 27708, USA
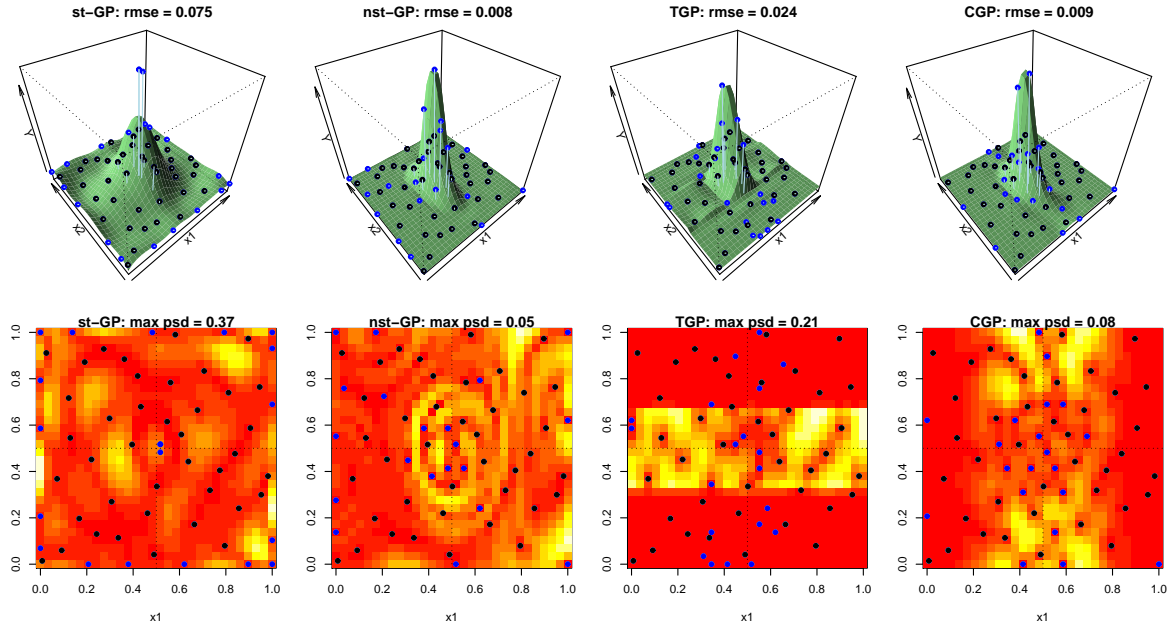
Web Appendix A

Two-dimensional numerical examples

Figure S1. Comparison between stationary GP (st-GP), non-stationary GP via latent input augmentation (nst-GP), TGP, and CGP on the 2D menhir function at $T = 60$. Black points: initial 40 LHD (common to all emulators); blue points: 20 additional points selected via ALM. Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0, 1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes. Note that the stationary GP does not interpolate at the peak. We defer a discussion on this phenomenon to Section 7 of the paper.
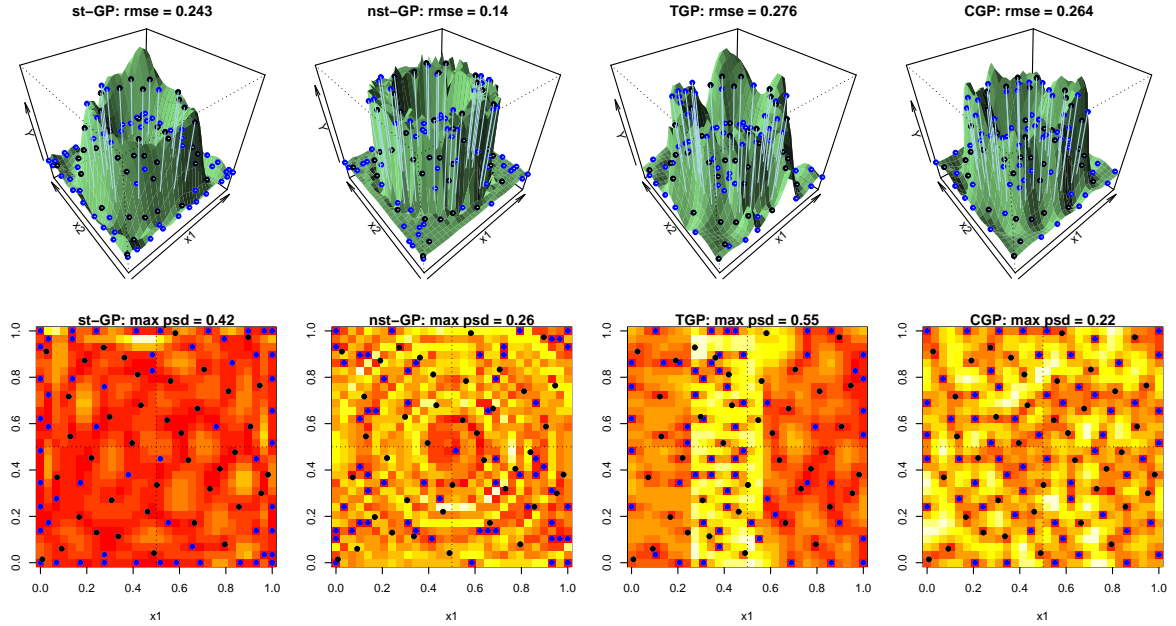
Figure S2. Comparison between stationary GP (st-GP), non-stationary GP via latent input augmentation (nst-GP), TGP, and CGP on the 2D well function at $T = 100$. Black points: initial 40 LHD (common to all emulators); blue points: 60 additional points selected via ALM. Top row: posterior mean predictive surface, $\hat{f}$, and root mean squared error (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 in $\Omega = [0, 1]^2$, i.e. an expanded grid of 30 equally spaced points along each coordinate axes.

Web Appendix B

Six-dimensional numerical examples

Our methodology relies on two stationary GPs, one for the function of interest and one for the latent input:

$$f|\boldsymbol{\theta} \sim \text{GP}(\mu_{\boldsymbol{\theta}}, C_{\boldsymbol{\theta}}), \qquad \text{and} \qquad g|\boldsymbol{\theta} \sim \text{GP}(0, \tilde{K}_{\boldsymbol{\theta}}), \tag{1}$$

where $\boldsymbol{\theta}$ is a vector of model parameters. To simplify the notations, we shall drop the $\boldsymbol{\theta}$ subscript to the mean, covariance, and correlation functions hereafter. For our formulation, we choose

$$\mu(\boldsymbol{x}) = h(\boldsymbol{x})^\top \boldsymbol{\beta}, \quad \text{with} \quad h(\boldsymbol{x}) = [1, \boldsymbol{x}]^\top \tag{2}$$

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma^2 K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\sum_{l=1}^{p} \phi_l (x_{il} - x_{jl})^2 - \phi_{p+1}(Z_i - Z_j)^2\right\} \tag{4}$$

$$Z_i = g(\boldsymbol{x}_i) \tag{5}$$

$$\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\sum_{l=1}^{p} \tilde{\phi}_l (x_{il} - x_{jl})^2\right\}. \tag{6}$$

Parameter $\phi_l \geq 0$ (or $\tilde{\phi}_l \geq 0$ for the latent correlation function $\tilde{K}$) controls the sensitivity of f (g) to x_l. For example, $\phi_l = 0$ ($\tilde{\phi}_l = 0$) removes x_l (dimension reduction), whereas larger ϕ_l ($\tilde{\phi}_l$) denotes smaller correlation, i.e. $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ ($g(\boldsymbol{x})$ and $g(\boldsymbol{x}')$) are less related in the x_l

direction and the function is more complex.

We implement our emulator via particle learning (PL - [2]) as described in Section 3 of the paper. We train our emulator on a 120 LHD and then let it select 80 additional points via ALM, for a total of 200 points in the final design. Each particle contains an estimate of model parameters in (1)-(6). Figure S3 shows the distribution across particles of the estimated $\{\tilde{\phi}_i\}_{i=1}^6$ for the 6D well example. We recall that true surface is

$$f(x_1, \ldots, x_6) = \begin{cases} 1, & \text{if } \sum_{i=1}^4 (x_i - 0.5)^2 > 0.025 \text{ and } \sum_{i=1}^4 (x_i - 0.5)^2 < 0.25 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

on the hypercube $\boldsymbol{X} = [0,1]^6$. Therefore, f in (7) is constant in x_5 and x_6. 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 ϕ_7, which is associated to the latent input Z (Figure S4). This indicates that the latent input is needed to learn the surface well. At every input configuration $\boldsymbol{x}$ corresponds an estimate of the latent input $Z = g(\boldsymbol{x})$. The last two panels show the estimated latent input at each design point, $g(\boldsymbol{x}_1), \ldots, g(\boldsymbol{x}_{200})$, plotted versus the first and fourth dimension of the corresponding input configuration. Specifically, we plot $g(\boldsymbol{x}_1)$ (with $\boldsymbol{x}_1 = (x_{1,1}, \ldots, x_{1,6})$) at $x_{1,1}$ (or $x_{1,4}$ in the last panel), $g(\boldsymbol{x}_2)$ at $x_{2,1}$ (or $x_{2,4}$), and so forth. Despite being slightly elongated, the circular pattern of the well function emerges in the estimated latent input.
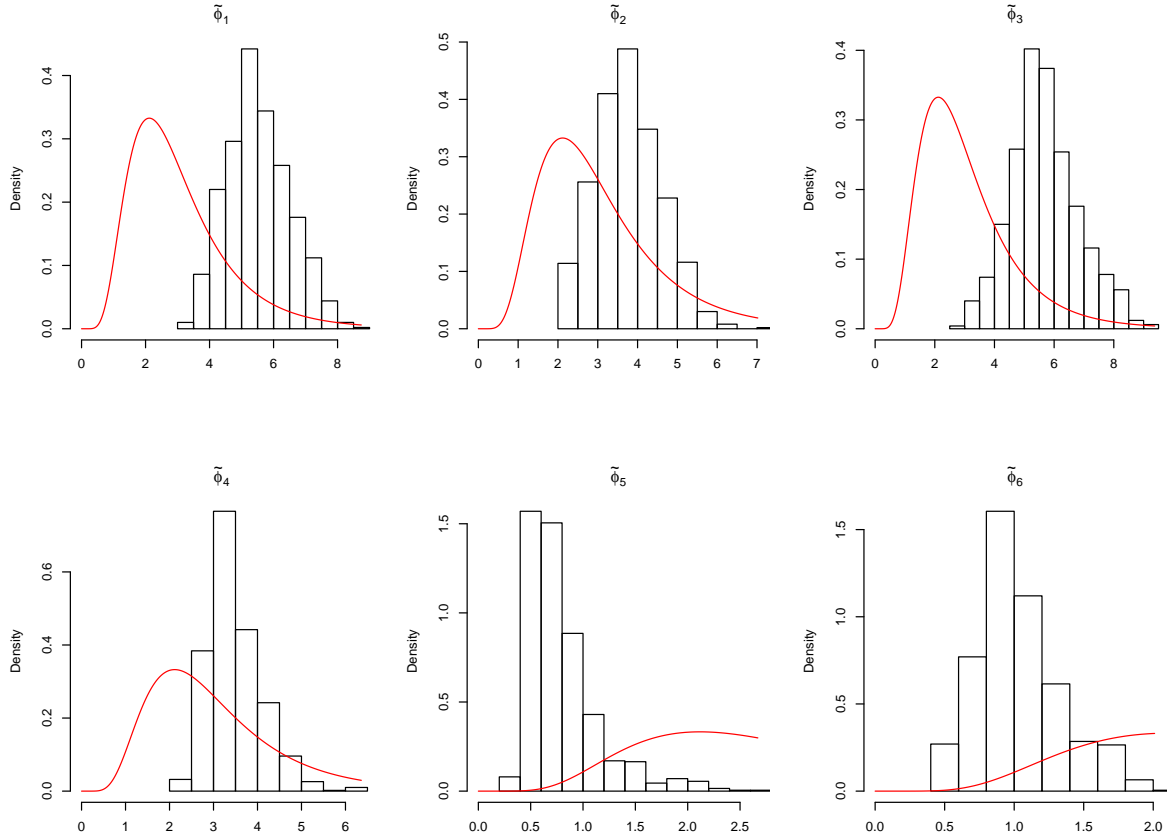
Figure S3. Distribution across particles of the correlation length parameters of $\tilde{K}$, $\{\tilde{\phi}_i\}_{i=1}^6$, at $T = 200$ (120 fixed LHD + 80 additional points selected via ALM) in the 6-dimensional well example (7). The red curve denotes the prior distribution on the latent correlation parameters. Specifically, $\log \tilde{\phi}_i \sim$ N$(0.5, 0.25)$, $i = 1, \ldots, 6$.
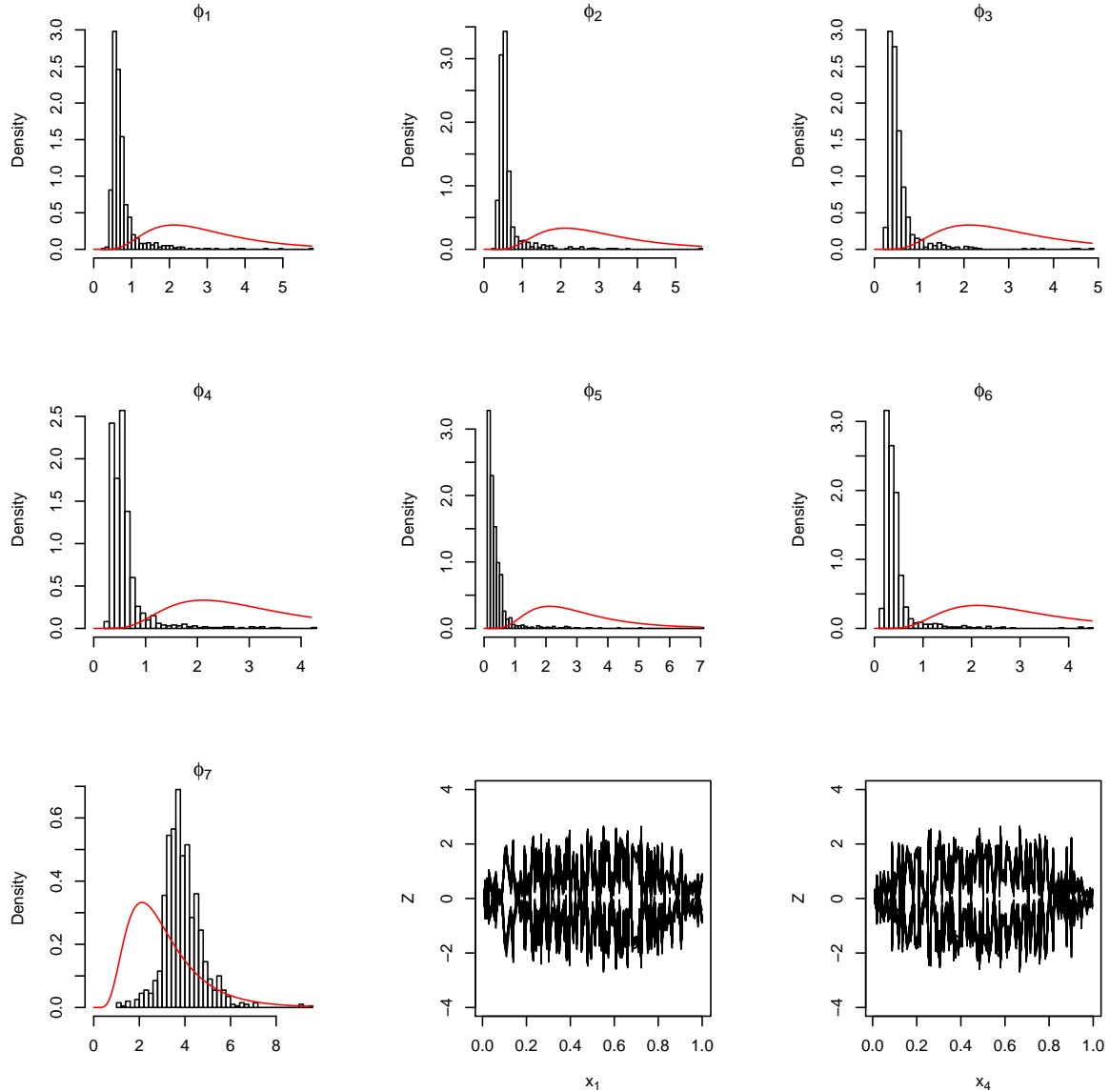
Figure S4. Panels $\{\phi_i\}_{i=1}^7$: distribution across particles of the correlation length parameters of K, $\{\phi_i\}_{i=1}^7$, at $T = 200$ (120 fixed LHD + 80 additional points selected via ALM) in the 6-dimensional well example (7). The red curve denotes the prior distribution on the correlation parameters. Specifically, $\log \phi_i \sim \mathrm{N}(1, 0.25), i = 1, \ldots, 7$. The last two panels show the latent input estimated at each design point, $g(\boldsymbol{x}_1), \ldots, g(\boldsymbol{x}_{200})$, plotted versus the first and fourth dimension of the corresponding input configuration. Specifically, plots show $g(\boldsymbol{x}_1)$ at $x_{1,1}$ (or $x_{1,4}$ in the last panel), $g(\boldsymbol{x}_2)$ at $x_{2,1}$ (or $x_{2,4}$), and so forth.
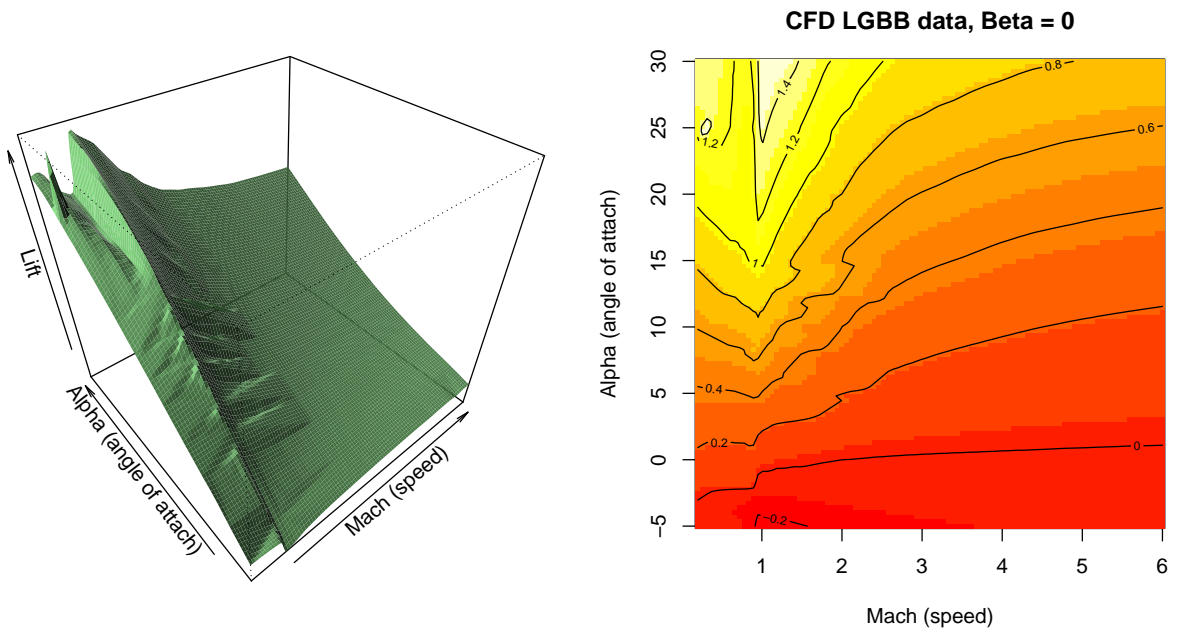
Figure S5. Interpolated lift surface plotted as a function of Mach (speed) and Alpha (angle of attack) with Beta (side-slip angle) fixed to zero. The ridge at Mach 1 denotes a distinction between subsonic flows and supersonic flows. The upper-left corner of the plot (high angle of attack, low speed) shows a spike which is a result of false convergence of the simulator [1].

Web Appendix D

Two-stage empirical Bayes approximation of non-stationary GP emulator

The particle learning (PL) implementation of our emulator is formulated around an effort at joint modelling of both f and Z (Section 3). The latent input $Z = g(\boldsymbol{x})$ essentially becomes a vector of model parameters that can not be marginalised out within our construction, thus needs to be learnt. The learning of the latent input vector, which is of sequentially increasing dimension within our PL construction, can face challenges and a large amount of data may be needed to learn it well. In alternative to the PL "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 $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^t$. We consider $Z = g(\boldsymbol{x})$ and model g by a stationary GP indexed by the p-dimensional vector of known inputs, $\boldsymbol{x}$, and a vector of model parameters, $\boldsymbol{\theta}$:

$$g|\boldsymbol{\theta} \sim \mathrm{GP}(\tilde{\mu}_{\boldsymbol{\theta}}, \tilde{K}_{\boldsymbol{\theta}}). \tag{8}$$

Several choices are available for the GP mean $\tilde{\mu}_{\boldsymbol{\theta}}$ including the constant-zero mean. The correlation function $\tilde{K}_{\boldsymbol{\theta}}$ corresponds to Equation (2.8) in Section 2.2. If we consider a unit scale and fix $\tilde{\mu}_{\boldsymbol{\theta}} \equiv 0$, (8) reduces to the GP prior on g as of Expression (2.7) in Section 2.2. One can estimate g from a smooth GP (noisy) regression:

$$f(\boldsymbol{x}_i) = g(\boldsymbol{x}_i) + \epsilon_i, \quad \text{with} \quad \epsilon_i \sim \mathrm{N}(0, \tau^2), \quad \text{and} \quad i = 1, \dots, t. \tag{9}$$

Overall, the model in (8)-(9) is equivalent to assuming a GP prior on f:

$$f|\boldsymbol{\theta}, \tau^2 \sim \mathrm{GP}(\tilde{\mu}_{\boldsymbol{\theta}}, \tilde{K}_{\boldsymbol{\theta}} + \tau^2 \delta_{i,j}), \tag{10}$$

where $\delta_{.,.}$ is the Kronecker delta function. Bayesian inference of (10) proceeds via MCMC by drawing realisations from the joint posterior distribution of the model parameters. For all $\boldsymbol{x}$ of interest and using the parameter values drawn from the joint posterior at iteration k, we can estimate Z at $\boldsymbol{x}$ as $\hat{g}_k(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})|\boldsymbol{x}, \{\boldsymbol{\theta}, \tau^2\}_k]$, the point predictor of f at $\boldsymbol{x}$ at iteration k. 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 $f|\hat{g}, \boldsymbol{\theta} \sim \mathrm{GP}(\mu_{\boldsymbol{\theta}}, K_{\boldsymbol{\theta}})$, where f is now a stationary GP indexed by a $p+1-$dimensional vector of inputs $\{\boldsymbol{x}, \hat{g}(\boldsymbol{x})\}$ as in (2.5)-(2.6) (Section 2.2) under the fiction that the latent input is known. Inference for $\boldsymbol{\theta}$ proceeds via MCMC and sequential design is embedded 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 in that 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 $[\boldsymbol{x}_{t+1}, f(\boldsymbol{x}_{t+1})]$. Fits from previous iterations can only guide the initialization of the new Markov Chain.

Simulation studies

We evaluate the performance of the two-stage approximation on the sequential experiments presented in Section 5.

Figure S6 shows the performance of the two-stage approximation on the 2D examples of Section 5.1 given the initial 40 LHD (black points). Figure S7 reports the predictive surface and standard deviation after 20 additional inputs (blue points) are selected based on uncertainty. The patterns observed in the predictive standard deviation (panels in the second row of Figure S6 and S7) show that the emulator is learning the geometry of the local features. The emulator creates a balance between an *exploration* of the input space via the LHD (given design) and the (*exploitation*) of the local features as shown by the higher concentration of new inputs at the edges of the features.

Figure S8 shows a comparison between two-stage approximation and PL "full Bayes" in terms of progression of the root mean square error (RMSE). Top panels refer to the 2D examples outlined in Section 5.1: 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 (top right), whereas two-stage is preferred on the Menhir function for small designs (top left). This is probably due to the ability of the two-stage approximation in learning better 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. Bottom panels in Figure S8 show 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.

References

[1] 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.

[2] H. F. Lopes, C. M. Carvalho, M. S. Johannes, and N. G. Polson. *Particle learning for sequential Bayesian computation.* Oxford University Press, 2011.
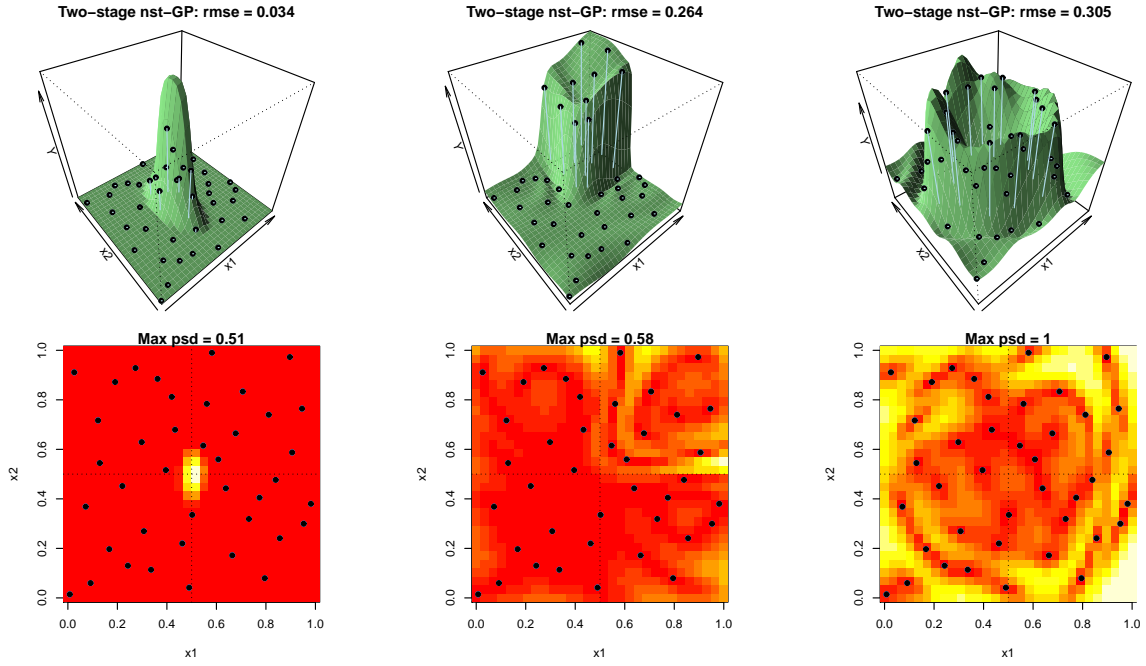
Figure S6. Predictive surface and standard deviation at a set of 900 predictive points $[0, 1]^2$ obtained with the two-stage MCMC-based implementation of the non-stationary GP emulator on the 2D Menhir, building, and well examples numerical examples. Quantitative summaries report the root mean squared 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.
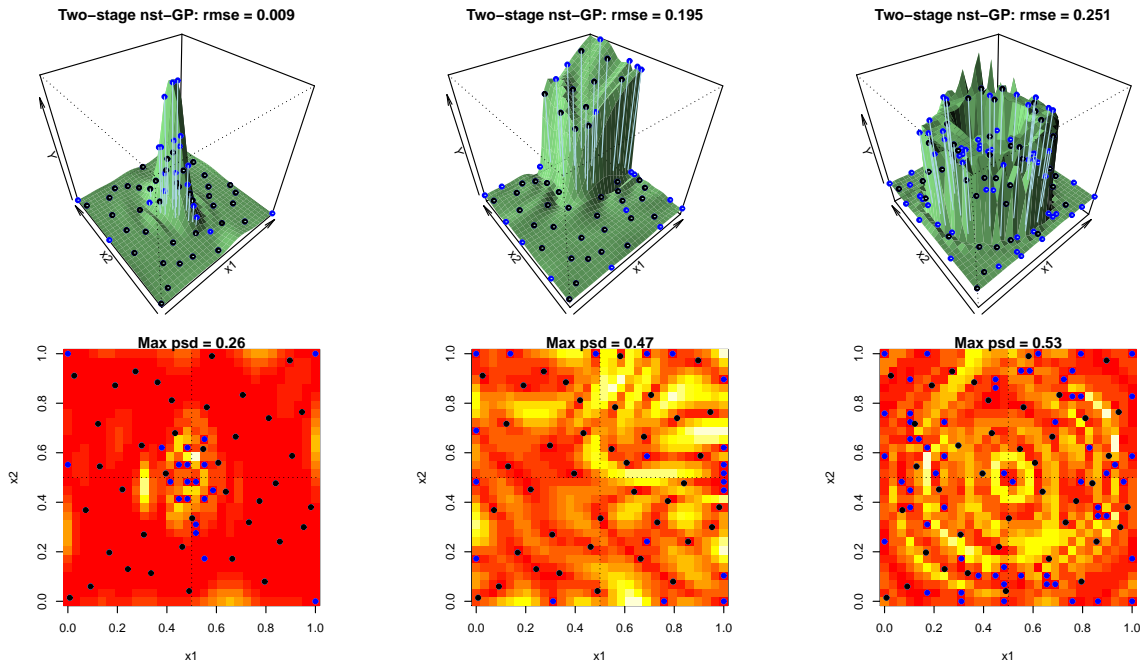

Figure S7. Predictive surface and standard deviation at a set of 900 predictive points $[0, 1]^2$ obtained with the two-stage MCMC-based implementation of the non-stationary GP emulator on the 2D Menhir, building, and well examples numerical examples. Quantitative summaries report the root mean squared error (RMSE) and the maximum predictive standard deviation (pred sd) computed based on the test points. Black points: initial 40 LHD (given). Blue points: additional points selected via ALM criterion.
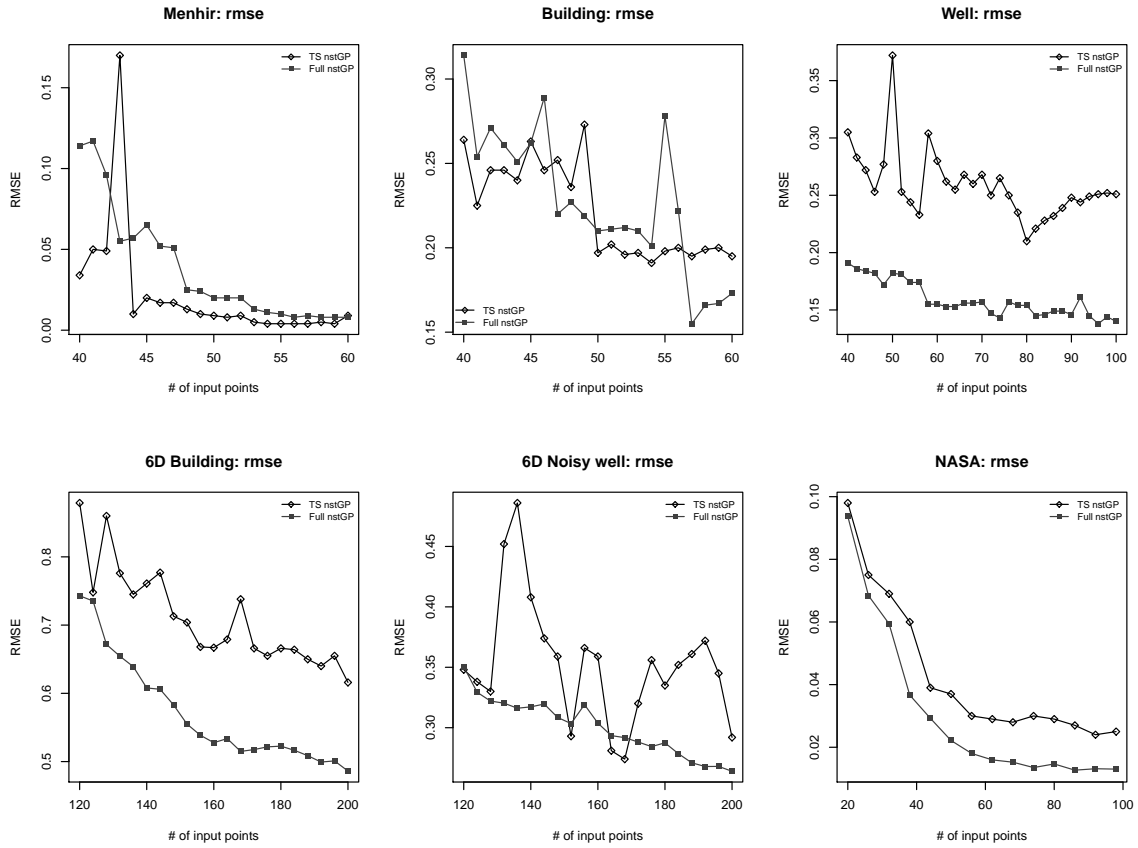
9

Figure S8. Comparison between two-stage approximation (TS nstGP) and "full Bayes" (Full nstGP) in terms of RMSE progression as additional input points are being selected. Top panels: 2D Menhir, building and well functions. Bottom panels: 6D building and well examples and NASA rocket booster experiment.