# TRANSITIONAL ANNEALED ADAPTIVE SLICE SAMPLING FOR GAUSSIAN PROCESS HYPER-PARAMETER ESTIMATION

*A. Garbuno-Inigo,*[1,*] *F. A. DiazDelaO,*[1] *& K. M. Zuev*[2]

[1]*Institute for Risk and Uncertainty, School of Engineering, University of Liverpool, Brownlow Hill, Liverpool, L69 3GH, United Kingdom*

[2]*Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, California 91125, USA*

[*]Address all correspondence to: A. Garbuno-Inigo, E-mail: agarbuno@liv.ac.uk

*Original Manuscript Submitted: 8/16/2016; Final Draft Received: 9/28/2016*

*Surrogate models have become ubiquitous in science and engineering for their capability of emulating expensive computer codes, necessary to model and investigate complex phenomena. Bayesian emulators based on Gaussian processes adequately quantify the uncertainty that results from the cost of the original simulator, and thus the inability to evaluate it on the whole input space. However, it is common in the literature that only a partial Bayesian analysis is carried out, whereby the underlying hyper-parameters are estimated via gradient-free optimization or genetic algorithms, to name a few methods. On the other hand, maximum a posteriori (MAP) estimation could discard important regions of the hyper-parameter space. In this paper, we carry out a more complete Bayesian inference, that combines Slice Sampling with some recently developed sequential Monte Carlo samplers. The resulting algorithm improves the mixing in the sampling through the delayed-rejection nature of Slice Sampling, the inclusion of an annealing scheme akin to Asymptotically Independent Markov Sampling and parallelization via transitional Markov chain Monte Carlo. Examples related to the estimation of Gaussian process hyper-parameters are presented. For the purpose of reproducibility, further development, and use in other applications, the code to generate the examples in this paper is freely available for download at http://github.com/agarbuno/ta2s2_codes.*

**KEY WORDS:** *Gaussian process, slice sampling, simulated annealing, Markov chain Monte Carlo, sequential Monte Carlo*

## 1. INTRODUCTION

The use of computationally expensive computer codes is widespread in science and engineering in order to simulate and investigate complex phenomena. Such codes, also referred to as *simulators*, often require intensive use of computational resources that allows their use in contexts such as optimization, uncertainty analysis, and sensitivity analysis [1, 2]. For this reason, surrogate models are needed to efficiently approximate the output of demanding simulators and enable efficient exploration and exploitation of the input space. In this context, Gaussian processes are a common choice to build statistical surrogates—also known as *emulators*—which account for the uncertainty that stems from the inability to evaluate the original model in the whole input space. Gaussian processes are able to fit complex input/output mappings through a nonparametric hierarchical structure. Common applications of Gaussian processes are found in machine learning [3], spatial statistics [4] (with the name of Kriging), likelihood-free Bayesian inference [5], genetics [6] and engineering [7], among many other areas.

Building an emulator requires the simulator to be run a repeated number of times, but due to its computational complexity only a limited amount of evaluations is available. This cost often translates to an inadequate explanation of the uncertainty of the model parameters by unimodal probability distributions. This phenomenon is often encountered in the form of low signal-to-noise ratio which translates into multimodal posterior distributions of the parameters [8]. Although it is not pathological as it does not substantially affect the Gaussian process predictive performance, this might lead to parameters of the surrogate with poor inference significance, e.g., if a sensitivity analysis is performed by studying the length-scales. The Bayesian treatment of the inference problem often alleviates this concern by means of the prior; however multimodality cannot be fully discarded [9]. In this setting, where the information available to train a Gaussian process is limited, one is able to acknowledge all uncertainties related to the modeling assumptions by resorting to model uncertainty analysis [10]. More specifically, *hierarchical modeling* should be considered in the model formulation. By doing so, the analyst is capable of accounting for structural uncertainty, which can be considered either as a continuous or discrete construct. In Gaussian process models, continuous structural uncertainty can be incorporated through the Bayesian paradigm as each configuration of the hyper-parameters corresponds to a different predictive posterior distribution.

The implementation of a Gaussian process emulator requires a training phase. This involves the estimation of the parameters characterizing the Gaussian process, referred to as *hyper-parameters*. The selection of the hyper-parameters is usually done by maximum likelihood estimates (MLE) [1], maximum a posteriori estimates (MAP) [3, 11], or by sampling from the posterior distribution [12] in a fully Bayesian manner. It is frequently the case that estimating the hyper-parameters depends on maximizing a complex, multimodal function. In this scenario, traditional optimization routines [13] are not able to attain global optima when searching for the MLE or MAP, and a Bayesian treatment becomes a suitable option to account for all the uncertainties in the modeling. In the literature, however, it is common that either the MLE or MAP alternatives are preferred [2, 14] due to the low numerical burden of maximizing the likelihood function or because it is assumed that Bayesian integration (for example, through Markov chain Monte Carlo methods) is prohibitive. Although these are strong arguments in favor of point estimates, in high-dimensional applications it is difficult to assess if the number of runs of the simulator is sufficient to produce robust hyper-parameters. This is usually measured with prediction-oriented metrics such as the root-mean-square error (RMSE) [2]. The use of this metric ignores the uncertainty and risk assessment of choosing a single candidate for the hyper-parameters through an inference process with limited data. In order to account for such uncertainty, numerical integration should be performed. However, methods such as quadrature approximation become quickly infeasible as the number of dimensions increases [2]. Therefore, a suitable approach is to perform Monte Carlo integration [15]. This allows to approximate any integral by means of a weighted sum, given a sample from the *correct* distribution.

The Bayesian formulation of the Gaussian process model is unlikely to allow posterior inference of the hyper-parameters by means of standard distributions (Gaussian, uniform, or exponential, to name a few). In order to be able to approximate the related integrals, Markov chain Monte Carlo (MCMC) provides the proper statistical tool to generate a desired sample. Nonetheless, the canonical sampling schemes, such as Metropolis-Hastings or Gibbs sampling, might not be appropriate for multimodal distributions [16, 17]. This limitation is originated by the tuning of the proposal distribution, the function used to generate samples. If it is not correctly specified, the sampling space might not be properly explored. The efficiency of the sampler should balance the ability to move freely through the sampling space as well as to generate candidates according to the regions where the probability mass is concentrated. One of the most recognized concerns in simulation is to avoid *random walk* behavior, since it delays the stationarity state achieved by the chain and limits its exploration capabilities [18]. An optimal tuning of the proposal distribution in high-dimensional spaces can turn into a demanding task if intricate correlation among sets of variables is present. As a consequence, MCMC samplers become expensive to use [19]. One alternative in Gaussian processes, among other probabilistic applications, is to resort to the hybrid Monte Carlo (HMC) sampler, as it can avoid random walk behavior at the expense of additional computations needed for the gradient of the posterior [12, 20]. However, there is no guarantee that multimodal distributions can be sampled thoroughly by HMC [21].

This paper proposes a sampling scheme for multimodal distributions based on two principles: firstly, the concept of *crumb* introduced by Neal [22] for a multivariate adaptive slice sampler; secondly, the ideas by Zuev and Beck [23] on how to simulate through a sequence of nested subsets as in stochastic subset optimization [24, 25], with the Asymptotically Independent Markov Sampler. The use of delayed rejection in the Asymptotically Independent

Markov Sampler [26] has proven to enhance the mixing capabilities in highly correlated probability models. To our knowledge, coupling the adaptive Slice Sampling algorithm with a sequential sampler has not been explored previously. This presents an opportunity to develop efficient sampling algorithms for multi-modal distributions. The main advantage of the proposed scheme is that it requires little tuning of parameters as it automatically learns the sequence of temperatures for an annealing schedule, as opposed to being tuned by trial and error [27]. The approximation set simulated in the previous level can be exploited further as it provides the crumbs needed for the sampling in the next annealing level, leading the simulation to appropriate regions of the sampling space. Additionally, embedding the sampler with the transitional Markov chain Monte Carlo method [19] results in an algorithm that can be run in a cluster of cores, if available. By using the proposed transitional annealed adaptive slice sampling (TA$^2$S$^2$) algorithm to sample the hyper-parameters of a Gaussian process, the resulting emulator is built taking into account both a probabilistic and computationally efficient perspective. Efficiency is gained as the sampler adapts the proposal distribution, which for other MCMC schemes is a highly sensitive parameter to be tuned. The probabilistic strategy to treat the problem in a Bayesian manner accounts for the uncertainty that stems from the unknown parameters. This adds a layer of structural uncertainty to the model. Additionally, model uncertainty is accounted for by adding numerical stabilization measures in the Gaussian process model as in [9, 28] in a fully Bayesian framework.

The paper is organized as follows. In Section 2, a brief introduction to the Bayesian treatment of Gaussian processes, as well as related numerical stabilization procedures, are presented. Section 3 briefly reviews the concepts of Slice Sampling and adaptive Slice Sampling. Section 4 presents the proposed algorithm with the concepts discussed in the previous sections, as well as the extensions needed for a parallel implementation. In Section 5, some illustrative examples are used to discuss the efficiency and robustness of the proposed algorithm. Concluding remarks are presented in Section 6.

## 2. THE GAUSSIAN PROCESS MODEL

Let the real-valued function $\eta : \mathbb{R}^p \to \mathbb{R}$ represent the underlying input/output mapping of a computer model. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the set of *design points*, that is, the set of selected points in the input space, where $\mathbf{x}_i \in \mathbb{R}^p$ denotes a given input configuration. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ be the corresponding set of outputs $y_i = \eta(\mathbf{x}_i)$, such that each pair $(\mathbf{x}_i, y_i)$ denotes a *training run*. The emulator is assumed to be an interpolator for the training runs, i.e., $y_i = \tilde{\eta}(\mathbf{x}_i)$ for all $i = 1, \ldots, n$, where the tilde denotes approximation. This omits any random error in the output of the computer code in the observed simulations, for which the simulator is said to be deterministic. If a fully parametrized Gaussian process prior is assumed for the outputs of the simulator, then the set of design points has a joint Gaussian distribution. The general assumption is that the simulator satisfies the statistical model for the output with the following structure:

$$\eta(\mathbf{x}) = h(\mathbf{x})^\top \boldsymbol{\beta} + Z(\mathbf{x}|\sigma^2, \boldsymbol{\phi}), \tag{2.1}$$

where $h(\cdot)$ is a vector of known basis (location) functions of the input, $\boldsymbol{\beta}$ is a vector of regression coefficients, and $Z(\cdot|\sigma^2, \boldsymbol{\phi})$ is a Gaussian process with zero mean and covariance function,

$$\text{cov}(\mathbf{x}, \mathbf{x}'|\sigma^2, \boldsymbol{\phi}) = \sigma^2\, k(\mathbf{x}, \mathbf{x}'|\boldsymbol{\phi}), \tag{2.2}$$

where $\sigma^2$ is the signal noise and $\boldsymbol{\phi} \in \mathbb{R}^p_+$ denotes the *length-scale* parameters of the correlation function $k(\cdot, \cdot)$. The hyper-parameters of the Gaussian process are therefore $\theta = (\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi})$. For practical simplicity it is commonly assumed that $h(\mathbf{x}) = 0$. This allows to perform predictions and quantify the underlying uncertainty by relying completely on the covariance function to capture the dependencies among training runs. Thus, the discussion of the inference on $\boldsymbol{\beta}$ is left out in the remainder. Note that for a pair of design points $(\mathbf{x}, \mathbf{x}')$, the function $k(\cdot, \cdot|\boldsymbol{\phi})$ measures the correlation between $\eta(\mathbf{x})$ and $\eta(\mathbf{x}')$ based on their respective input configurations. The correlation function is capable of measuring how close different input configurations are, such that related inputs produce related outputs in the simulator. The base of such measure is related to the Euclidean distance in such a way that it weights differently each input variable. In this work, the squared-exponential correlation function has been chosen due to its tractability, namely,

$$k(\mathbf{x},\mathbf{x}'|\boldsymbol{\phi}) = \exp\left\{-\frac{1}{2}\sum_{i=1}^{p}\frac{(x_i - x_i')^2}{\phi_i}\right\}. \tag{2.3}$$

It is important to note that other authors [3, 20, 29] prefer the $\phi_i^2$ parametrization in the denominator. In our case, it is more natural to use a linear term, given that the length-scale parameters are restricted to lie in the positive orthant. The linear terms can be interpreted as weights in the norm used to measure closeness and sensitivity to changes in each dimension.

As a consequence of the Gaussian process prior, the joint probabilistic model for the vector of outputs $\mathbf{y}$, given the hyper-parameters $\sigma^2$, $\boldsymbol{\phi}$, and the design points $X$, can be written as

$$\mathbf{y}|X,\sigma^2,\boldsymbol{\phi} \sim \mathcal{N}(0,\sigma^2\,K), \tag{2.4}$$

where $K$ is the correlation matrix with elements $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j|\boldsymbol{\phi})$ for all $i,j = 1,\ldots,n$.

## 2.1 Hyper-Parameter Marginalization

The hyper-parameters of the Gaussian process emulator are commonly unknown before the training phase, which adds uncertainty to the surrogate. A common practice in the literature is to fix them to their maximum likelihood value. Though it has been widely accepted, this approach does not entirely treat the emulator as a probabilistic model and uncertainty quantification through it might become limited. On the other hand, if one acknowledges the parameters as random variables, robust estimators can be built through numerical integration by marginalizing them via samples generated from their posterior distribution. This allows to incorporate all possible configurations of the surrogate in light of the evidence shed by the training runs. Predictions for $y^*$, given a nonobserved configuration $\mathbf{x}^*$, can be performed using all the evidence provided by the available data $\mathcal{D} = (\mathbf{y},X)$, exploiting the posterior distribution of the hyper-parameters, namely,

$$p(y^*|\mathbf{x}^*,\mathcal{D}) = \int_\Theta p(y^*|\mathbf{x}^*,\mathcal{D},\theta)\,p(\theta|\mathcal{D})\,d\theta, \tag{2.5}$$

where $\theta = (\sigma^2,\boldsymbol{\phi})$ denotes the vector of hyper-parameters. Note that the model used for predictions $y^*$, given the data and $\theta$, is a Gaussian random variable, which makes the model closed under the Gaussian family (see [11]).

By means of a sample from the posterior distribution, the mean and covariance functions of the predictive posterior can be written through a mixture of Gaussians [26] with individual weights $w_i$ as

$$\mu(\mathbf{x}^*) = \sum_{i=1}^{N} w_i\;\mu_i(\mathbf{x}^*), \tag{2.6}$$

$$\text{cov}(\mathbf{x}^*,\mathbf{x}') = \sum_{i=1}^{N} w_i\;\left[(\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*))(\mu_i(\mathbf{x}') - \mu(\mathbf{x}')) + \text{cov}(\mathbf{x}^*,\mathbf{x}'|\theta_i)\right], \tag{2.7}$$

where $\mu_i(\mathbf{x}^*)$ is the expected value of the probability model of $y^*$ conditional on the hyper-parameters $\theta_i$, the training runs $\mathcal{D}$ and the input configuration $\mathbf{x}^*$. From Eq. (2.7) it follows that the variance (also known as the prediction error) of an untested configuration $\mathbf{x}^*$ is

$$s^2(\mathbf{x}^*) = \sum_{i=1}^{N} w_i\;((\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*))^2 + s_i^2(\mathbf{x}^*)). \tag{2.8}$$

This results in a more robust estimation of the prediction error, since it balances the predicted error in one sample with how far the prediction of such sample is from the overall estimation of the mixture.

## 2.2 Prior Distributions

The predictive posterior distribution in Eq. (2.5) requires the specification of a prior distribution $p(\sigma^2, \boldsymbol{\phi})$. Weak prior distributions have been used for $\boldsymbol{\phi}$ and $\sigma^2$ [11], namely,

$$p(\sigma^2, \boldsymbol{\phi}) \propto \frac{p(\boldsymbol{\phi})}{\sigma^2}, \tag{2.9}$$

where $\sigma^2$ is assumed to have a noninformative distribution. For the length-scale hyper-parameters $\boldsymbol{\phi}$, the reference prior [30, 31] allows for an objective framework in which the uncertainty of $\boldsymbol{\phi}$ can be accounted for. It requires no previous knowledge, such that the training runs are the only source of information for the inference process. Additionally, the reference prior is capable of ruling out subspaces of the sample space of the hyper-parameters [32], thus reducing regions of possible candidates in the mixture model expressed proposed as in Eqs. (2.6)–(2.8). This allows for prior distributions without concerns about expert knowledge of feasible regions for the hyper-parameters. Unless otherwise stated, the reference prior for Gaussian processes developed by [33] is used in this work. Note, however, that there are no known analytical expressions for the derivatives of this prior, which limits its application to samplers that require first-order information like HMC. Additionally, it is important to note that there are other possibilities available for the prior distribution of $\boldsymbol{\phi}$. Examples of these are the log-normal or log-Laplacian distributions, which can be interpreted as a regularization in the norm of the parameters. Other alternatives suggest a decaying prior [32] or a weakly informative distribution such as a gamma with appropriate parameters. If needed, elicited priors from experts can also be used [34].

## 2.3 Marginalizing the Nuisance Hyper-Parameters

The hyper-parameters $\boldsymbol{\phi}$ and $\sigma^2$ may be potentially different in terms of scales and dynamics, as explained previously in [26]. Gibbs sampling might help with this limitation when used for multimodal distributions in high-dimensional spaces. The Metropolis-Hastings sampler exhibits the same problem. For this reason, this work focuses on $\boldsymbol{\phi}$ to perform the inference based in the correlation function. To achieve this, $\sigma^2$ is considered as a nuisance parameter and is integrated out from the posterior. This way, all inference is driven by the length-scale hyper-parameters of the correlation function. Note that other authors follow different approaches in the inference problem of Gaussian processes as emulators. For example, the authors of [35] focus on the global trend function $h(\cdot)^\top \boldsymbol{\beta}$ since it allows to incorporate expert knowledge on the computer model being emulated. That is, a fully parametrized Gaussian process is considered. The probability model of the training runs and the prior distribution of the hyper-parameters (Eqs. (2.4) and (2.9)) allow to identify an inverse-gamma model for $\sigma^2$, which after integration, can be shown to yield the marginalized posterior distribution,

$$p(\boldsymbol{\phi}|\mathcal{D}) \propto p(\boldsymbol{\phi})\, (\hat{\sigma}^2)^{-(n-p)/2}\, |K|^{-1/2}, \tag{2.10}$$

where

$$\hat{\sigma}^2 = \frac{\mathbf{y}^\top K^{-1} \mathbf{y}}{n-1} \tag{2.11}$$

is an estimator of the signal noise $\sigma^2$ (see [11] for further details). Finally, the predictive distribution conditioned on the remaining set of hyperparameters, $\boldsymbol{\phi}$, follows a t-student distribution with mean and correlation functions

$$\mu(\mathbf{x}^*|\boldsymbol{\phi}) = t(\mathbf{x}^*)^\top K^{-1}\mathbf{y}, \tag{2.12}$$

$$\mathrm{corr}(\mathbf{x}^*, \mathbf{w}^*|\boldsymbol{\phi}) = k(\mathbf{x}^*, \mathbf{w}^*|\boldsymbol{\phi}) - t(\mathbf{x}^*)^\top K^{-1}\, t(\mathbf{w}^*), \tag{2.13}$$

where $\mathbf{x}^*$, $\mathbf{w}^*$ denote a pair of untested configurations and $t(\mathbf{x}^*)$ denotes the vector obtained by computing the covariance of a new input configuration with every design point available for training $t(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1|\boldsymbol{\phi}), \ldots, k(\mathbf{x}, \mathbf{x}_n|\boldsymbol{\phi}))^\top$. Note that both the mean and the correlation of unseen input configurations depend solely on the correlation function hyper-parameters $\boldsymbol{\phi}$. See [36] for further details in the inference of nuisance and integrated posteriors.

In light of the above, this paper focuses on the correlation function $k(\cdot, \cdot)$ in Eq. (2.2), since the structural dependencies of the training runs are recovered by it.

## 2.4 Numerical Stability

Numerical stability is usually difficult to guarantee when implementing Gaussian processes. As explored previously by [9] and [28], a term can be added in the covariance matrix $K$ in order to preserve diagonal dominancy, that is, to add a *nugget* hyper-parameter $\boldsymbol{\phi}_\delta$ such that

$$K_\delta = K + \boldsymbol{\phi}_\delta \, I \tag{2.14}$$

is positive definite. This results in the stochastic simulator

$$y_i = \eta(\mathbf{x}_i) + \sigma^2 \, \boldsymbol{\phi}_\delta, \tag{2.15}$$

where the term $\sigma^2 \, \boldsymbol{\phi}_\delta$ is used to account for the variability of the simulator that cannot be explained by the correlation function. The inclusion of the nugget modifies the posterior distribution, and possibly adds new modes. This is a consequence of the noise process added by the $\boldsymbol{\phi}_\delta \, I$ term which has the potential to mutate the Gaussian process into a linear regression model or a pure noise process. In such scenario, the use of multimodal-oriented samplers is crucial for performing the Monte Carlo approximation of Eq. (2.5). In case the resulting model is assessed as not appropriate, a regularization term (an elicited prior) can be added to penalize regions of local modes [9].

As previously discussed in [28], a uniform prior distribution $U(10^{-12}, 1)$ is adopted for the nugget hyper-parameter $\phi_\delta$. The lower bound guarantees stability in computations associated with the covariance matrix. The upper bound forces the numerical noise of the simulator to be smaller than the signal noise of the emulator itself. By considering the modified correlation matrix in Eq. (2.14), the previous assumptions regarding the original noise parameter $\sigma^2$ remain unchanged, as one can still marginalize it as a nuisance parameter with the noninformative prior used [37].

## 3. SLICE SAMPLING

The Slice Sampling algorithm [22] is a method to simulate a Markov chain of a random variable $\theta \in \Theta$. This is done by introducing an auxiliary random variable $u \in \mathcal{U} \subseteq \mathbb{R}$ and sampling from the joint distribution on the extended space $\Theta \times \mathcal{U}$. The marginal of $\theta$ is recovered by disregarding the values of $u$ in the Markov chain, a consequence of defining an appropriate conditional distribution for $u$, given $\theta$. The samples are generated by an iterative Gibbs sampling schedule to recover pairs $\{(\theta_i, u_i)\}_{i=1}^N$, which follow the joint density probability distribution

$$\pi(\theta, u) \propto I_{\{u<\pi(\theta)\}}(\theta, u), \tag{3.1}$$

where $I_E(\cdot)$ is the indicator function for the set $E \subset \Theta \times \mathcal{U}$, and $\pi(\theta)$ is the target distribution of $\theta$. Slice Sampling first generates $u$ from the conditional distribution of $u \,|\, \theta$ specified as a uniform on the interval $(0, \pi(\theta))$. It then samples $\theta$, conditioned in $u$ from a uniform distribution in the *slice* defined by the set

$$S_u = \{\theta : u < \pi(\theta)\}. \tag{3.2}$$

Since the marginal satisfies $\int_0^{\pi(\theta)} \pi(\theta, u) \, du = \pi(\theta)$, samples from the target distribution can be recovered by disregarding the auxiliary component of the joint samples. If the target distribution is a non-normalized probability density $f(\theta)$ then the joint distribution can be written as

$$\pi(\theta, u) = \frac{1}{Z} \, I_{\{u<f(\theta)\}}(\theta, u), \tag{3.3}$$

where $Z = \int_\Theta f(\theta) \, d\theta$ and the previous considerations for the marginal of $\theta$ follow. In the context of Gaussian processes it should be noted that floating-point underflows are common due to ill-conditioning of the matrix $K$ in Eq. (2.10). Thus, in order to compute stable evaluations of the target distribution in Slice Sampling, it is preferable to evaluate the negative logarithm of the target density. In such case, Eq. (3.2) can be computed as stated in the following proposition.

**Proposition 1** (Slice characterization)**.** *Given the state of the Markov chain* $\theta_0$*, the uniform distribution for the next candidate has support in the slice given by*

$$S_{\theta_0} = \{\theta : z > \mathcal{H}(\theta)\}, \tag{3.4}$$

*where* $\mathcal{H}(\cdot)$ *denotes the negative logarithm of the target density and* $z = \mathcal{H}(\theta_0) + e$*, with* $e$ *distributed as an exponential random variable with mean equal to 1.*

*Proof.* The result follows from the fact that for a given state $\theta_0$ of the Markov chain, the auxiliary uniform random variable defining the slice can be written as the product $u \times f(\theta_0)$ with $u$ uniformly distributed in the interval $(0, 1)$. Thus, the slice is defined as

$$\begin{aligned} S_{\theta_0} &= \{\theta : u\, f(\theta_0) < f(\theta)\} \\ &= \{\theta : -\log(f(\theta_0)) - \log(u) > -\log(f(\theta))\} \\ &= \{\theta : \mathcal{H}(\theta_0) + e > \mathcal{H}(\theta)\}, \end{aligned} \tag{3.5}$$

where it is easy to prove that $e = -\log(u)$ is distributed as an exponential random variable with mean 1 and $\mathcal{H}(\cdot)$ denotes the negative logarithm of the target density. □

The main concern when implementing Slice Sampling is the ability to sample uniformly from the slice. In one-dimensional applications, the slice can be defined in many ways. The canonical example is a stepping-out and shrinkage procedure which aims to adapt an initial interval centered in the current state of the Markov chain (see [22] for further details).

### 3.1 Adaptive Slice Sampling

For multivariate distributions, the concept of the slice extends naturally. However, methods based on intervals (e.g., the stepping-out and shrinking procedure) become dramatically slow as the dimension of the problem increases. This is due to the generalization of intervals as hyper-rectangles in $\mathbb{R}^p$ and the need to compute the target function for each vertex a repeated number of times along the expansion and shrinkage of the boundaries. For Gaussian process emulators, the task of evaluating the target density becomes expensive, a consequence of the nonparametric nature of the model and the computational cost of evaluating Eq. (2.10). If multiple evaluations are needed for the construction of the Markov chain either because of a high rejection rate, difficult characterization of the slice, or if longer chains are required, simulation by MCMC with Slice Sampling becomes computationally expensive and inefficient. Therefore, other alternatives are preferable.

This work employs a framework proposed by [22] for adaptive slice sampling in multivariate applications. The key idea is the use of the information provided by the rejected samples in order to lead the future generation of a candidate inside the slice. In this framework, the evidence gathered by the rejected candidates is referred to as *crumbs*, as they will be "followed" towards the slice.

## 4. TRANSITIONAL ANNEALED ADAPTIVE SLICE SAMPLING

As previously stated, in order to marginalize the posterior predictive distribution in Eq. (2.10), Monte Carlo integration is usually performed when aiming at a fully Bayesian treatment of Gaussian process surrogates. This is usually done by hybrid Monte Carlo [12, 20] which is capable of suppressing the random walk behavior of traditional MCMC methods. Nonetheless, the tuning of this kind of algorithm is problem-dependent and expert knowledge is crucial for an optimal sampling schedule. The development of elliptical Slice Sampling [29] provides a framework for the simulation of the hyper-parameters of a Gaussian process with little tuning required from the analyst [38]. However, this is only applicable when the posterior predictive distribution for the hyper-parameters is of the form

$$p(\theta|\mathcal{D}) \propto \mathcal{N}(l(\theta)|\boldsymbol{\mu}, \Sigma)\, p(\theta), \tag{4.1}$$

where $p(\theta)$ denotes the prior distribution, $\mathcal{N}(\cdot|\cdot,\cdot)$ is a Gaussian distribution, and $l(\cdot)$ is a latent variable that depends on the hyper-parameters. As it can be seen from the integrated posterior in Eq. (2.10), this is not an expression for the assumed posterior distribution. The difference stems from $\sigma$ being considered a nuisance parameter and the prior considered for the length-scales of the Gaussian process.

In this setting, we propose transitional annealed adaptive Slice Sampling (TA$^2$S$^2$), which can also be used in other applications of Bayesian inference and stochastic optimization. Based on Asymptotically Independent Markov Sampling [39] we formulate the sampling problem to be solved as reminiscent of simulated annealing. The objective is to sample from intermediate posterior distributions $p_k(\phi|\mathcal{D})$ that eventually converge to the true posterior. This is done by tempering the posterior distribution by means of a monotonically decreasing sequence of temperatures $\tau_k$ converging to 1. Let $\{p_k(\phi|\mathcal{D})\}_{k=1}^{\infty}$ be the sequence of density distributions in the annealing schedule such that

$$p_k(\phi|\mathcal{D}) \propto p(\phi|\mathcal{D})^{1/\tau_k} = \exp\left\{-\mathcal{H}(\phi|\mathcal{D})/\tau_k\right\}, \tag{4.2}$$

where $\mathcal{H}(\phi|\mathcal{D})$ denotes the negative integrated log-posterior distribution of the length-scale hyper-parameters, given the set of training runs $\mathcal{D}$.

The algorithm provides a sequence of nested subsets $\Phi_{k+1} \subseteq \Phi_k$ converging to the set of posterior samples denoted by $\Phi^*$. The temperature is learned through an automatic mechanism to determine the sequence of distributions.

By construction, the sample in the first level of annealing is distributed uniformly on a *practical support* of the sampling space (see [40] for more a detailed discussion). For the limiting case, the samples are uniformly distributed in the support of the posterior density. Both these observations can be summarized by

$$\lim_{\tau\to\infty} p_\tau(\phi|\mathcal{D}) = U_\Phi(\phi), \tag{4.3}$$

$$\lim_{\tau\to 1} p_\tau(\phi|\mathcal{D}) = U_{\Phi^*}(\phi), \tag{4.4}$$

where $U_A(\phi)$ denotes a uniform distribution over the set $A$ for every $\phi \in A$.

## 4.1 Annealing at Level $k$

This subsection focuses on the sampling carried out by TA$^2$S$^2$ at the $k$-th level of the annealing sequence. It is assumed that a sample from level $k-1$, which is distributed according to $p_{k-1}(\phi|\mathcal{D})$, has already been generated. Let $\phi_1^{(k-1)},\ldots,\phi_N^{(k-1)}$ denote such sample and let $N$ be the sample size in each annealing level. Following the ideas discussed in Section 3.1 for adaptive Slice Sampling, the *crumb* formulation will be exploited. The samples from the previous level play the role as the crumbs to be followed to generate candidates from each slice. Thus, retaining information from the posterior landscape and limiting the amount of evaluations of the integrated posterior, which can be expensive for a reasonable number of training runs. Firstly, note that Proposition 1 implies the following:

**Corollary 1** (Slice set at the $k$th level)**.** *The slice defined in the $k$th annealing level, given the current state of the Markov chain $\phi_0$, is given by*

$$S^k_{\phi_0} = \{\phi : z_k > \mathcal{H}(\phi|\mathcal{D})\}, \tag{4.5}$$

*where $z_k = \mathcal{H}(\phi_0|\mathcal{D}) + e_k$, with $e_k$ an exponential random variable with mean $\tau_k$.*

As in other sequential Monte Carlo algorithms [41, 42], let us define the importance weights of the samples $\phi_1^{(k-1)},\ldots,\phi_N^{(k-1)}$ as

$$\omega_j^{(k-1)} = \frac{p_k\left(\phi_j^{(k-1)}\right)}{p_{k-1}\left(\phi_j^{(k-1)}\right)} \propto \exp\left\{-\mathcal{H}\left(\phi_j^{(k-1)}|\mathcal{D}\right)\left(\frac{1}{\tau_k}-\frac{1}{\tau_{k-1}}\right)\right\}, \tag{4.6}$$

$$\overline{\omega}_j^{(k-1)} = \frac{\omega_j^{(k-1)}}{\sum_{j=1}^N \omega_j^{(k-1)}}, \tag{4.7}$$

where $\omega_j^{(k-1)}$ denotes the importance weights and $\overline{\omega}_j^{(k-1)}$ the normalized importance weights. The weights allow to measure the importance of each sample as being drawn for the next annealing level.

The proposal for a new state of the Markov chain, given the current one, $\boldsymbol{\phi}_0$, is generated as follows. A slice is obtained as in Eq. (4.5) by generating an exponential random variable with mean $\tau_k$, thus defining the slice $S^k_{\boldsymbol{\phi}_0}$ for the current state. A first crumb is randomly selected from the set of past approximations that lie inside the slice. This means selecting a uniformly distributed index $j$ from the set

$$\mathcal{J} = \left\{ j \in \{1, \ldots, N\} \; : \; \boldsymbol{\phi}_j^{(k-1)} \in S^k_{\boldsymbol{\phi}_0} \right\}. \tag{4.8}$$

The points $\boldsymbol{\phi}_1^{(k-1)}, \ldots, \boldsymbol{\phi}_N^{(k-1)}$ are uniformly distributed in the approximation set $\Phi_{k-1}$ and will be used as markers for the annealing level $k$. If the above index set is empty, there is evidence of the annealing temperature being decreased too rapidly. A fail-safe can be used by generating a crumb from a wide Gaussian distribution centered at the current state of the Markov chain. Additionally, as it is done in other sequential Monte Carlo methods [43], a renewal component can be added. The renewal is performed as the crumbs are selected from the markers, due to the fact that relying on the sample from the previous level can lead to bias in the simulations. To this end, if the index set $\mathcal{J}$ is empty, or with probability $p_{\text{renew}}$, the crumb $\boldsymbol{\varsigma}_1$ will be distributed as

$$\boldsymbol{\varsigma}_1 \sim \mathcal{N}(\boldsymbol{\phi}_0, c_0^2\, \Sigma_k), \tag{4.9}$$

where $c_0$ is a spread parameter associated with the annealing sequence, and $\Sigma_k$ denotes a covariance matrix at level $k$. Typical choices for the covariance matrix are the identity matrix $I_{p\times p}$ or a diagonal matrix diag$\{d_1, \ldots, d_p\}$ which defines a different scale for each variable. In order to use a better proposal in terms of scales and correlations observed along the annealing sequence, $\Sigma_k$ is defined as the weighted covariance matrix from the weighted samples $\{(\overline{\omega}_j^{(k-1)}, \boldsymbol{\phi}_j^{(k-1)})\}_{j=1}^N$. As discussed in [44], the spread parameter is set as $c_0 = 2.38/\sqrt{p}$, since it allows for efficient transitions in Gaussian steps.

Once the first crumb is drawn, a first candidate $\boldsymbol{\xi}_1$ is generated from the appropriate Gaussian distribution

$$\boldsymbol{\xi}_1 \sim \mathcal{N}(\boldsymbol{\varsigma}_1, c_0^2\, \Sigma_k), \tag{4.10}$$

where $c_0$ is a spread parameter for the proposals and $\Sigma_k$ defined as above. In general, the $i$-th candidate for the next state of the Markov chain can be generated as

$$\boldsymbol{\xi}_i \sim \mathcal{N}\left(\overline{\varsigma}_i, \left(\frac{c_0}{i}\right)^2 \Sigma_k\right), \tag{4.11}$$

where $\overline{\varsigma}_i$ is the average of the crumbs generated so far, as proposed in [22]. Note how the generation of new candidates in the slice is narrower as the candidates are rejected by means of the parameter $c_0/i$. However, the mean for the Gaussian proposal might not converge to a point in the slice if the posterior is a multimodal distribution. To cope with this limitation, we propose to use a weighted average of the current state and the crumb center to enhance the mixing of the sampler, namely, by sampling the $i$-th candidate from a Gaussian distribution with mean

$$\overline{\varsigma}_i^* = \alpha_i\, \boldsymbol{\phi}_0 + (1 - \alpha_i)\, \overline{\varsigma}_i \tag{4.12}$$

and covariance $(c_0/i)\, \Sigma_k$. The weight parameter $\alpha_i$ can be defined in terms of the number of crumbs previously rejected. Since it is desirable that $\alpha_i \to 1$ as $i$ increases, we can define it either as $\alpha_i = (1 - 1/i)$ or $\alpha_i = (1 - \exp(-i))$. As confirmed by our experiments, $\alpha_i$ is linearly-dependent on the crumb iteration, since the exponential behavior exhibits pronounced decay towards the current state, causing random walk behavior. The sampling in each annealing level is summarized in Algorithm 1. To avoid cluttered notation, the conditioning on the design points $\mathcal{D}$ is dropped in the remainder.

**Algorithm 1:** TA$^2$S$^2$ at annealing level $k$

```
Input:
    ◇ $\boldsymbol{\phi}_1^{(k-1)}, \ldots, \boldsymbol{\phi}_N^{(k-1)} \sim p_{k-1}(\boldsymbol{\phi})$, generated at previous level;
    ◇ $\boldsymbol{\phi}_1^{(k)} \in \Phi$, initial state of the chain;
Output:
    ◇ $\boldsymbol{\phi}_1^{(k)}, \ldots, \boldsymbol{\phi}_N^{(k)} \sim p_k(\boldsymbol{\phi})$;
begin
    Compute covariance matrix $\Sigma_k$ from the weighted samples;
    for $i \leftarrow 1$ to $N-1$ do
        Define slice $S^k_{\boldsymbol{\phi}_i}$ as in (4.5);
        Define the crumb counter as: $l \leftarrow 0$;
        do
            $l \leftarrow l+1$, and generate $u \sim U(0,1)$;
            if $|\mathcal{J}| \neq \emptyset$ or $u < p_{renew}$ then
                Choose random $j$ from index set $\mathcal{J}$;
                $\varsigma_l = \boldsymbol{\phi}_j^{(k-1)}$ ;
            else
                Generate $\varsigma_l \sim \mathcal{N}(\boldsymbol{\phi}_i^{(k)}, c_0^2\,\Sigma_k)$ ;
            end
            Define crumb as $\overline{\varsigma}_l^* = \alpha_l\,\boldsymbol{\phi}_i^{(k)} + (1-\alpha_l)\,\overline{\varsigma}_l$;
            Generate candidate $\boldsymbol{\xi}_i \sim \mathcal{N}(\overline{\varsigma}_l^*, (c_0/l)^2\,\Sigma_k)$;
        while $\boldsymbol{\xi}_i \notin S^k_{\boldsymbol{\phi}_i}$;
        Define new state of the chain $\boldsymbol{\phi}_{i+1}^{(k)} = \boldsymbol{\xi}_i$;
    end
end
```

## 4.2 Overview of the Full Sampler

The algorithm starts with a uniform sample in an admissible space $\Phi$, as implied by the *meta*-prior distribution in Eq. (4.3). As a second step, the algorithm described in the previous section is used to generate the samples of the first annealing level, that is, $\boldsymbol{\phi}_1^{(1)}, \ldots, \boldsymbol{\phi}_N^{(1)} \sim p_1(\boldsymbol{\phi})$. As mentioned before, this set of points allows to approximate the slices in the next annealing level and the areas where the posterior mass is concentrated. As the sequence of temperatures converges to 1, we expect to recover better approximations until posterior samples are generated—that is, until a sample $\boldsymbol{\phi}_1^{(k^*)}, \ldots, \boldsymbol{\phi}_N^{(k^*)}$ has been drawn and is uniformly distributed in the set $\Phi^*$. In the next section we discuss how to learn the temperature sequence and the overall parallel implementation achieved by embedding it on a transitional Markov chain schedule.

### *4.2.1 Annealing Schedule*

The way the temperature sequence is determined is one of the most crucial aspects of any simulated-annealing-based method. It is clear that if the change of temperatures is abrupt the markers will degenerate quickly, as observed in sequential Monte Carlo samplers. On the contrary, if the sequence of temperatures decreases slowly the actual efficiency of the algorithm is hindered, since sampling in a sequence of annealing levels is redundant for the generation of a posterior sample. Setting the temperature sequence beforehand requires prior knowledge of the overall behavior of the function $\mathcal{H}(\cdot)$ and the topology around the set $\Phi^*$, both of which are generally not available.

Following the suggestion by [23], the *effective sampling size* can be used as a measure of degeneracy of the chain in each annealing level. This allows to measure how similar the $(k-1)$-th and the $k$-th densities are. The effective sample size can be approximated by

$$\hat{n}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\overline{\omega}_j^{(k-1)}\right)^2}, \tag{4.13}$$

where $\overline{\omega}_j^{(k-1)}$ is the normalized weight of sample $\boldsymbol{\phi}_j^{(k-1)}$. Given that the temperature of the previous level is known, the problem is to determine the temperature of the next one. This is done by determining a target threshold for $\hat{n}_{\text{eff}}$ in terms of the size of the simulated set. Thus, given $\gamma \in (0,1)$, the target threshold is defined by $\gamma N = \hat{n}_{\text{eff}}$. Rewriting this expression in terms of the un-normalized sample weights we obtain

$$\frac{\sum_{j=1}^{N} \exp\left\{-2\mathcal{H}\left(\boldsymbol{\phi}_j^{(k-1)}\right)\,(1/\tau_k - (1/\tau_{k-1}))\right\}}{\left(\sum_{j=1}^{N} \exp\left\{-\mathcal{H}\left(\boldsymbol{\phi}_j^{(k-1)}\right)\,(1/\tau_k - (1/\tau_{k-1}))\right\}\right)^2} = \frac{1}{\gamma N}, \tag{4.14}$$

which yields an equation for the unknown temperature $\tau_k$. Solving the equation for $\tau_k$ can be done efficiently by standard numerical techniques such as the bisection method.

The value of the threshold $\gamma$ affects the overall efficiency of the annealing schedule. If a value close to zero is chosen, the resulting algorithm will create few tempered distributions and this will result in poor approximations. If $\gamma$ is close to 1, then there will be excessive tempered distributions and redundant annealing levels. As suggested by [39], and as confirmed by our experiments, a value of $\gamma = 0.5$ delivers acceptable efficiency.

### 4.2.2 Parallel Markov Chains

As described so far, the proposed algorithm can be computationally expensive if the Markov chain of the samples is drawn sequentially. This is due to the inversion of a $n \times n$ matrix and related products in Eq. (2.10). Hence, it is desirable to speed up the process of generating samples in each annealing level. In our context, the inversion of such matrix is not prohibitive, since we assume that the set of training points is expensive to acquire; however, a fast sampling algorithm is desired for a complete Bayesian treatment of the problem. This way we can compensate the drawbacks associated with an appropriate error estimation by using the emulator in a Bayesian setting (see [2] for a discussion). The idea of parallelization comes from an adaptation of the transitional Markov chain Monte Carlo (TMCMC) method [19] in the context of the annealed adaptive Slice Sampling algorithm described previously.

The TMCMC algorithm builds a Markov chain from a target distribution in a sequential schedule as in sequential Monte Carlo [45] and particle filtering [46]. That means that $N$ Markov chains are started, each from the state of an initial Markov chain being drawn from the prior distribution of the Bayesian inference problem. The key difference is that the chains are allowed to communicate among each other by a *transition* mechanism that allows to grow each chain differently within the same annealing level, disregarding poor initial states for certain chains. The length of the chain is determined by a probability proportional to the importance sampling weight defined in Eq. (4.6). By doing so, the markers are automatically selected in the updating sequence and concentrated around the modes found during the annealing. This improves the mixing of the samples generated in each annealing level.

Summarizing, the proposed TA$^2$S$^2$ algorithm consists of Markov chains generated as established in Algorithm 1, the annealing temperature being determined empirically by the effective sampling size described in Section 4.2.1 and stopped whenever the temperature reaches 1. The selection of the initial states of the Markov chains and their growth length is a direct implementation of the TMCMC method [19] for Bayesian model updating.

## 5. NUMERICAL EXPERIMENTS

The following examples illustrate the effectiveness and robustness of TA$^2$S$^2$ when sampling the hyper-parameters of Gaussian process surrogates. The first example is Franke's function [47], which can have challenging features when emulated. The second example is a five-dimensional model [48] which has been previously used to test Gaussian process meta-models [49]. The third example is a ten-dimensional model for the weight of a wing of a light aircraft [1]. Concerning the nugget of the surrogate, we perform a sigmoid transformation in order to sample all covariance hyper-parameters with multivariate Gaussian distributions as discussed in Section 4.1. That is, we introduce an auxiliary component $z_\delta$ and extend the vector of hyper-parameters $\boldsymbol{\phi}$ to $\mathbb{R}^{p+1}$. That is, $z_\delta$ is the $(p+1)$-th component to be sampled in the algorithm. Finally, we compute the nugget as

$$\theta_\delta = \frac{1 - l_b}{1 + \exp(-z_\delta)} + l_b, \tag{5.1}$$

where $l_b$ is the lower bound, which is set equal to $10^{-12}$ following the discussions in [28]. To incorporate the algorithm to the length-scale hyper-parameters, the sampling has been performed in logarithmic space to avoid additional concerns for the non-negative restrictions imposed to the aforementioned variables as in other sampling schedules [50]. The initial values of the algorithm, Eq. (4.3), are set to a uniform distribution in a wide practical range, that is, the interval $[-7, 7]$ for the length-scales. For the nugget, a noninformative truncated beta distribution in the interval $[l_b, 1]$ has been considered.

The code was implemented in MATLAB and all examples were run in a GNU/Linux machine with an Intel i5 processor with 8 Gb of RAM. For the purpose of reproducibility, the code used to generate the examples in this paper is available for download at http://github.com/agarbuno/ta2s2_codes.

In order to contrast our proposed methodology with existing ones, we take the particle learning sampler PLGP [51] as a benchmark. This sampler has proven effective for sampling the posterior distribution of the hyper-parameters of a Gaussian process by means of tempering in a data-oriented manner, i.e., by feeding subsets of the training runs in each annealing level. Although the proposed sampler can be implemented in an online fashion akin to PLGP, we resort only at comparing them as strategies in batch applications. Note that the extension to online learning tasks can be done by regarding the posterior of a subset of data as the prior for the next set of training runs. This can be followed easily as the reweighting of the samples by a data-oriented alternative to Eq. (4.6) can help adjust the importance of the samples.

Following the discussion of [52] proper scoring rules should be used in order to compare the probability statements made by the Gaussian process model resulting from the samples used to marginalize the predictive posterior. In the context of Gaussian processes, both prediction and error estimation are used to assess the quality of the surrogate, i.e., the estimated mean and variance. If a local scoring rule such as the negative logarithm of predictive density (NLPD) is used to evaluate the generated samples, we risk penalizing heavily overconfident predictions and treat with less rigor underconfident far-off predictions. This is not desirable since it is known that the full Bayesian treatment in Gaussian processes is preferred for better error estimation in uncertainty analysis [2]. In contrast, by using distance-sensitive scoring rules such as the continuously ranked probability score (CRPS) we aim for better placement of probability mass near target values, although not exactly placed at the target. It is defined as

$$\text{CRPS}(F, x) = \int_{-\infty}^{\infty} (F(y) - \mathbf{1}\{y \geq x\})^2 dy, \tag{5.2}$$

where $F$ is the cumulative predictive distribution and $x$ is the point where it is verified. We assume a Gaussian approximation for the predictions made by the Gaussian process emulator and use the mixture model expressed in Eqs. (2.6) and (2.8) to be able to use the complete mixture expression developed in [53] which we include for completeness. That is, for a mixture of Gaussians the CRPS can be written as

$$\text{CRPS}\left(\sum_{m=1}^{N} \omega_m \mathcal{N}(\mu_m, s^2),\, x\right) = \sum_{m=1}^{N} \omega_m A(x - \mu_m, s_m^2) - \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N} \omega_m \omega_n A\left(\mu_m - \mu_n, s_m^2 + s_n^2\right), \tag{5.3}$$

where $\omega_i$ denotes the weight of the sample, $\mu_i$ is the mean of $x$ given by sample $i$, and $s_i^2$ is the corresponding estimated variance. The function $A(\cdot, \cdot)$ is defined as

$$A(\mu, \sigma^2) = 2\sigma f_{\mathcal{N}}\left(\frac{\mu}{\sigma}\right) + \mu\left(2\, F_{\mathcal{N}}\left(\frac{\mu}{\sigma}\right) - 1\right), \tag{5.4}$$

where $f_{\mathcal{N}}(\cdot)$ and $F_{\mathcal{N}}(\cdot)$ denote the density and cumulative functions of a standard Gaussian random variable. It should be noted that the CRPS does not possess an analytic expression for every probability function used for prediction [53], thus the choice of using a mixture of Gaussians for the predictive posterior distribution instead of $t$-distributions.

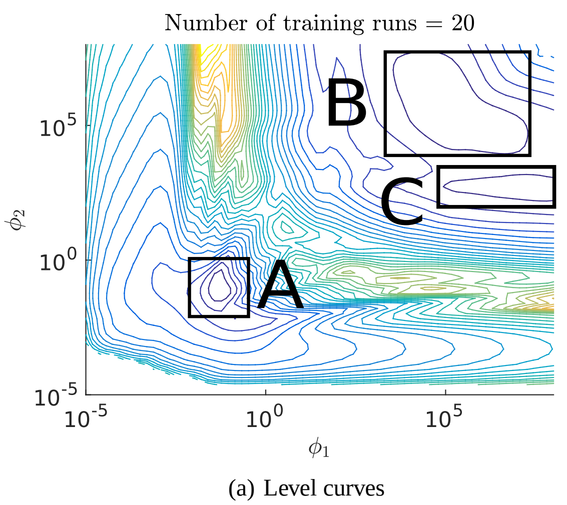
Other alternatives for scoring rules for Gaussian processes could be the bootstrapped variance predictor of [54]. This predictor aims to estimate the variance of the Gaussian process independent of the set of points used for training. However, such approach leads to prefer underconfident predictions not exactly around the target value and relies on the assumption of infinite repeatability of the simulator experiments. This is not satisfied by Bayesian analysis of computer code output (BACCO), since by assumption the generation of training runs is limited by computational cost.

## 5.1 Franke's Function

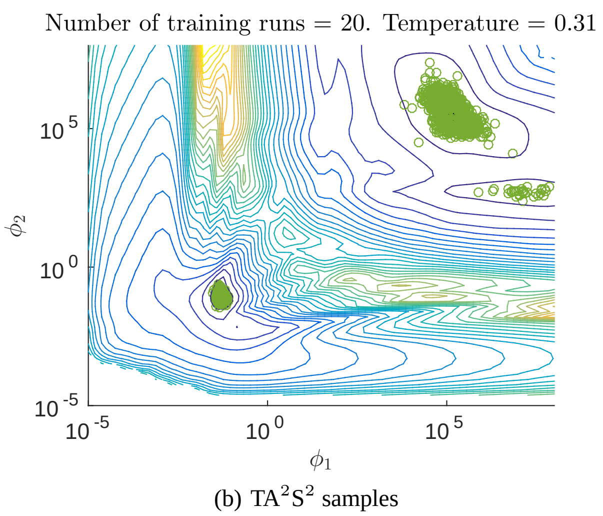
Franke's function has been used to test Gaussian process emulators [47]. Its complexity stems from the presence of two peaks and one dip in its landscape. Let $f : [0, 1]^2 \to \mathbb{R}$ be such that

$$\begin{aligned} f(\mathbf{x}) &= 0.75 \exp\left(-\frac{(9x_1 - 2)^2}{4} - \frac{(9x_2 - 2)^2}{4}\right) + 0.75 \exp\left(-\frac{(9x_1 + 1)^2}{49} - \frac{9x_2 + 1}{10}\right) \\ &+ 0.5 \exp\left(-\frac{(9x_1 - 7)^2}{4} - \frac{(9x_2 - 3)^2}{4}\right) - 0.2 \exp\left(-(9x_1 - 4)^2 - (9x_2 - 7)^2\right). \end{aligned} \tag{5.5}$$

To train the emulator, 20 design points were chosen using Latin hypercube sampling (LHS). For testing purposes, 100 independent design points were chosen by a second LHS. Figure 1(a) shows the multimodal integrated log-posterior for a fully-parametrized Gaussian process [26]. Region A contains a mode with no preference for any



(a) Level curves



(b) TA$^2$S$^2$ samples

**FIG. 1:** Projection of the negative log-posterior curves in the two-dimensional length-scale space for Franke's simulator using a fully-parametrized Gaussian process. The minimum possible value of $10^{-12}$ for the nugget $\boldsymbol{\phi}_\delta$ has been used for the projection. Depicted in Fig. 1(b), the temperature has been decreased beyond 1 to retrieve samples from the posterior modes.

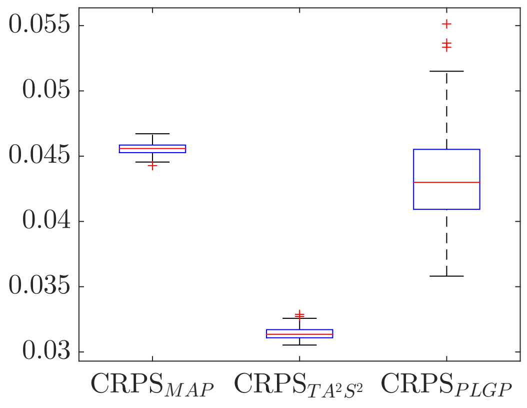
dimension. Regions B and C depict different asymptotic behaviors of the emulator. In region B, the emulator behaves as linear regression model, as noted by [9]. Region C corresponds to a model which disregards the first dimension. In Fig. 1(b) a set of samples obtained by applying TA$^2$S$^2$ is shown, illustrating the ability to overcome possible multimodal distributions that arise in Bayesian analysis of expensive computer codes.

To contrast the proposed sampler against the PLGP benchmark, a set of 100 experiments was run. In each experiment, a sample of size 100 length-scale hyper-parameters was obtained by each method. This sample size was achieved by thinning the TA$^2$S$^2$ results when a chain of length 2000 was constructed in every annealing level. The quality of the probability statements made from both results was compared by means of the CRPS, as depicted in Fig. 2. Note that the training set was the same for each experiment and variations in the results among experiments are mainly because of the stochastic nature of the sampling schemes. In Fig. 2(a) the box plots for the CRPS computed from the samples are shown. Figure 2(b) shows the same plots but with an exponential prior for all hyper-parameters with rate 0.2, i.e., $\lambda = 5$. This choice of a prior distribution was made since the PLGP software assumes an exponential prior for both the length-scales and nugget term [51]. In both settings, the proposed sampler outperforms both the PLGP alternative and the MAP estimate. The latter was calculated from the samples generated by TA$^2$S$^2$. Our experiments demonstrated that the MAP estimated this way usually corresponds to the one found by local optimization routines such as Nelder-Meade or BFGS. The variation in the scores of the MAP illustrates the multimodality properties of the integrated posterior. All the MAP estimates reported in the remainder are calculated based on this observation. Additionally, it can be seen that the PLGP results seem to contain those achieved by the most probable candidate. In this experiment either using a sample from PLGP or MAP translates to comparable results.

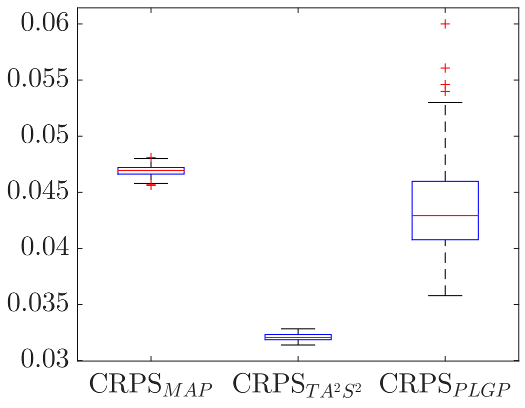
## 5.2 Nilson-Kuusk Model

This simulator models the reflectance of a homogeneous plant canopy. Its five-dimensional input space includes the solar zenith angle, the leaf area index, the relative leaf size, the Markov clumping parameter, and a model parameter $\lambda$ (see [48] for further details on the model itself and the meaning of the inputs and output). For the analysis presented in this paper, a single output emulator is assumed and the set of the inputs has been rescaled to fit the hyper-rectangle $[0, 1]^5$ as in [49].

In this experiment, both samplers were used to train a Gaussian process emulator with a dataset of 100 simulation runs. The test set consisted of a different set of 150 training runs. Both datasets, whose design points were generated through LHS, were obtained from the GEM-SA software web page (http://ctcd.group.shef.ac.uk/gem.html). On average, a total of 10 tempered distributions were used in the annealing schedule, while keeping the sampling as $N = 5000$ in each level. A thinned sample of 100 experiments was recovered by the end of each TA$^2$S$^2$ run to compare results.



(a) Reference prior



(b) Exponential prior, $\lambda = 5$

**FIG. 2:** Boxplots of CRPS comparing MAP, TA$^2$S$^2$, and PLGP. In both cases the proposed sampler outperforms PLGP.

The results shown in Fig. 3 demonstrate again the overall improved performance of using the proposed sampler in contrast with the benchmark. In this case, one set of experiments (50 iterations) consisted of making inference with the reference prior discussed previously, while the second set (50 iterations) used a common exponential prior. Both samplers outperform the MAP estimate which provides evidence that a more complete uncertainty analysis can be carried out if instead one turns to a full Bayesian inference scheme. Additionally, it is worth noting that Figures 3(a) and 3(b) show that the MAP changes with the prior used. This is not a matter of concern, as it is consistent with the notion that a small amount of data is being used for inference and the probabilistic model of the observables is not dominating the prior beliefs.

## 5.3 Wing Weight Model

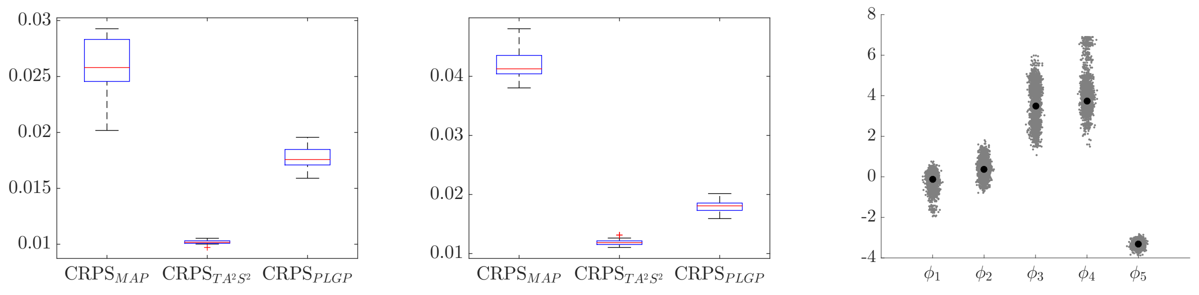
This simulator of the weight of the wing of a light aircraft [1] has been used for input screening. Coupled with a Gaussian process emulator with the squared exponential kernel in Eq. (2.3), a sensitivity analysis of the wing weight with respect to each input variable can be performed. The model is given by

$$f(\mathbf{x}) = 0.036\, S_w^{0.758}\, W_{fw}^{0.0035} \left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6} q^{0.006}\, \lambda^{0.04} \left(\frac{100\, t_c}{\cos(\Lambda)}\right)^{-0.3} (N_z\, W_{dg})^{0.49} + S_w\, W_p, \tag{5.6}$$

where the input variables and the range of their values are summarized in Table 1. For this problem, the evaluation of the reference prior is prohibitive since it scales with the number of dimensions [33]. Thus, a uniform prior in the hyper-parameters' log-space has instead been used for this experiment.

The inputs were rescaled to the 10-dimensional unit hypercube $[0, 1]^{10}$. Two LHS samples of size 100 and 300 were chosen as training and testing sets, respectively. At each annealing level, 5000 samples were generated, achieving convergence after 15 levels on average. As before, a thinned sample of 100 was kept to compare results with the benchmark in each experiment. A total of 50 experiments were run in this case.

In Figs. 4(a) and 4(b), it can be noted that by using sampling one can obtain an improved version of the probabilistic statements made by the surrogate. Figure 4(a) shows that the proposed sampler outperforms the benchmark, concentrating its samples around the mode of the posterior distribution. This is shown by the location and spread of the CRPS for both TA$^2$S$^2$ and MAP results, which are similar, although the CRPS with TA$^2$S$^2$ exhibits better performance as can been seen from the location and spread of the box plot. PLGP in the case of uniform priors seems to be sampling from other areas with less spread than that of the proposed algorithm. However, if an exponential prior distribution is used for both length-scales and nugget, as in Fig. 4(b), TA$^2$S$^2$ is outperformed by the benchmark. Note how the location of the mode changes dramatically, shown by both the means of the box plot of the CRPS of the MAP, and by the difference of the samples plotted in Figs. 4(c) and 4(d). Nonetheless, the proposed sampler is capable of offering improved probabilistic statements for the regression task compared to the MAP estimate.



(a) Reference prior

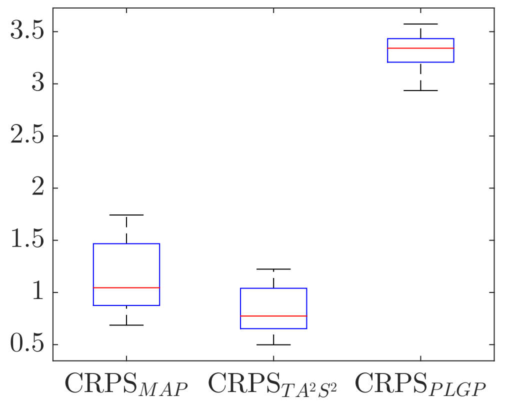
(b) Exponential prior, $\lambda = 5$

(c) Sample from TA$^2$S$^2$ for each length-scale

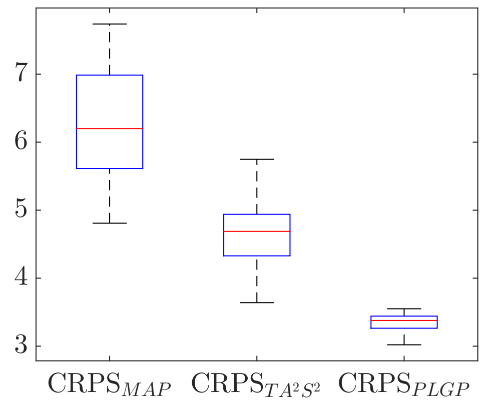
**FIG. 3:** Nilson-Kuusk results obtained after 50 experiments were run. The scatter plot [3(c)] is drawn from a single experiment; the MAP estimates are in black.

**TABLE 1:** Inputs of the wing weight model

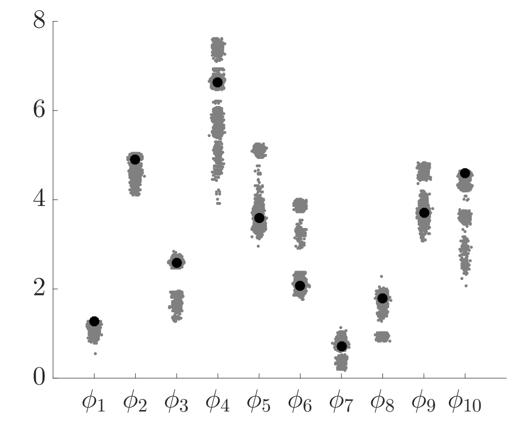
| **Input** | **Range** | **Description** |
|---|---|---|
| $S_w$ | $[150, 200]$ | Wing area |
| $W_{fw}$ | $[220, 300]$ | Weight of fuel in the wing |
| $A$ | $[6, 10]$ | Aspect ratio |
| $\Lambda$ | $[-10, 10]$ | Quarter-chord sweep |
| $q$ | $[16, 45]$ | Dynamic pressure at cruise |
| $\lambda$ | $[0.5, 1]$ | Taper ratio |
| $t_c$ | $[0.08, 0.18]$ | Aerofoil thickness to chord ratio |
| $N_z$ | $[2.5, 6]$ | Ultimate load factor |
| $W_{dg}$ | $[1700, 2500]$ | Flight design gross weight |
| $W_p$ | $[0.025, 0.08]$ | Paint weight |



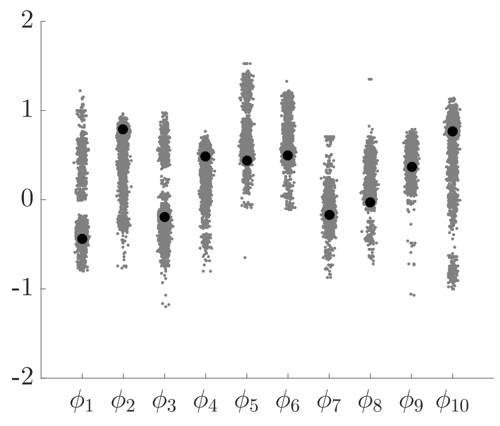
(a) Uniform prior



(b) Exponential prior, $\lambda = 5$



(c) Sample from $TA^2S^2$ for each dimension. Uniform prior



(d) Sample from $TA^2S^2$ for each dimension. Exponential prior

**FIG. 4:** Results for the wing-weight model using the $TA^2S^2$ and PLGP algorithms. The scatter plot [3(c)] is drawn from a single experiment; the MAP estimates are in black.

## 6. CONCLUSIONS

This paper proposes a method, transitional annealed adaptive Slice Sampling ($TA^2S^2$), to sample from the posterior distribution of the Gaussian process hyper-parameters. This is known to be a problem where multimodal distributions are encountered. $TA^2S^2$ combines Slice Sampling for delayed-rejection, transitional Markov chain Monte Carlo (TMCMC) for efficient parallelization and Markov chain growth, and Asymptotically Independent Markov Sampling (AIMS) for driving the annealing schedule. The delayed-rejection feature of Slice Sampling provides improved mixing which is desirable in highly correlated spaces. Additionally, the proposed algorithm provides a sampling scheme with no burn-in period for the Markov chain, since each sample generated follows the desired distribution. This is advantageous in Gaussian process applications where the computational burden of the sampling scheme is dominated by the inversion of the correlation matrix. Moreover, efficient coverage of the sampling space is achieved by annealing and an extension of Slice Sampling to sequential Monte Carlo.

The examples presented show how the method is capable of efficiently exploring multimodal distributions providing a better alternative to MAP estimates or traditional MCMC methods for Gaussian process applications. Efficiency is gained through the adaptive nature of the algorithm which allows to sample from complicated regions of the posterior distribution, namely, modes separated by large valleys of low probability. Furthermore, $TA^2S^2$ allows the generation of samples more efficiently than in traditional MCMC applications, where longer chains are needed to ensure good coverage of the sampling space. This is particularly relevant in the context of computer experiments and engineering, where it is usually considered that the cost of marginalizing the hyper-parameters does not justify the additional computational burden. The proposed method justifies the cost of sampling the hyper-parameters by providing robust estimates of the predicted error. This is reflected through the continuously ranked probability score (CRPS) which shows that the marginalized emulator outperforms MAP estimates. Moreover, the proposed sampling scheme performs as well as the PLGP benchmark, which justifies the annealing by means of functional tempering rather than by subsets of data, which are also sensitive to data permutations.

The proposed algorithm could also be employed for global optimization problems, by allowing the temperature to reach a practical zero. This has application in other areas of Bayesian inference problems and machine learning. As discussed previously, the algorithm can be used in active learning schedules by means of Bayes' theorem.

The computational cost of the proposed sampler is dominated by the inversion of the covariance matrix in the integrated posterior distribution. This limitation is inherent to any MCMC schedule used for Gaussian processes. Further strategies to improve the speed of the sampler could be developed, such as the exact evaluation of the log-posterior for candidates where the probability of acceptance is high. For such strategy, a first order Taylor expansion of the objective function could be a feasible enhancement. This is material for future research.

## ACKNOWLEDGMENT

The first author gratefully acknowledges the Consejo Nacional de Ciencia y Tecnología (CONACyT) for the award of a scholarship from the Mexican government for postgraduate studies.

## REFERENCES

1. Forrester, A. I. J., Sóbester, A., and Keane, A. J., *Engineering Design via Surrogate Modelling: A Practical Guide*, Chichester, UK: John Wiley & Sons, pp. 1–138, 2008.
2. Kennedy, M. C. and O'Hagan, A., Bayesian calibration of computer models, *J. R. Stat. Soc., Ser. B* , vol. **63**, no. 3, 425–464, 2001.
3. Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*, Cambridge, MA: MIT Press, pp. 7–30, 2006.
4. Cressie, N. A., *Statistics for Spatial Data*, New York: John Wiley & Sons, pp. 105–200, 1993.
5. Wilkinson, R. D., Accelerating ABC methods using Gaussian processes, In *Proc of 17th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pp. 1015–1023, 2014.
6. Kalaitzis, A. A. and Lawrence, N. D., A simple approach to ranking differentially expressed gene expression time courses through Gaussian process regression, *BMC Bioinf.*, vol. **12**, no. 1, pp. 1–13, 2011.

7. DiazDelaO, F. A. and Adhikari, S., Gaussian process emulators for the stochastic finite element method, *Int. J. Numer. Methods Eng.*, vol. **87**, no. 6, pp. 521–540, 2011.
8. Warnes, J. J. and Ripley, B. D., Problems with likelihood estimation of covariance function of spatial Gaussian processes, *Biometrika*, vol. **74**, no. 3, pp. 640–642, 1987.
9. Andrianakis, I. and Challenor, P. G., The effect of the nugget on Gaussian process emulators of computer models, *Comp. Stat. Data Anal.*, vol. **56**, no. 12, pp. 4215–4228, 2012.
10. Draper, D., Assessment and propagation of model uncertainty, *J. R. Stat. Soc., Ser. B*, vol. **57**, no. 1, pp. 45–97, 1995.
11. Oakley, J., *Bayesian uncertainty analysis for complex computer codes*, Ph.D. Thesis, University of Sheffield, 1999.
12. Williams, C. K. I. and Rasmussen, C. E., Gaussian processes for regression, *Adv. Neural Inf. Process. Syst.*, vol. **8**, pp. 514–520, 1996.
13. Nocedal, J. and Wright, S. J., *Numerical Optimization*, New York: Springer, pp. 30–163, 2004.
14. Gibbs, M. N., *Bayesian Gaussian processes for regression and classification*, Ph.D. Thesis, University of Cambridge, 1998.
15. MacKay, D. J., Introduction to Monte Carlo methods, in *Learning in Graphical Models*, pp. 175–204. Springer, 1998.
16. Neal, R. M., Annealed importance sampling, *Stat. Comput.*, vol. **11**, no. 2, pp. 125–139, 2001.
17. Hankin, R., Introducing BACCO, an R bundle for Bayesian analysis of computer code output, *J. Stat. Software*, vol. **14**, no. 16, pp. 1–16, 2005.
18. Neal, R. M., Probabilistic inference using Markov chain Monte Carlo methods, *Technical Report*, University of Toronto, pp. 1–144, 1993.
19. Ching, J. and Chen, Y.-C., Transitional Markov chain Monte Carlo method for Bayesian model updating, model class selection and model averaging, *J. Eng. Mech.*, vol. **133**, no. 7, pp. 816–832, 2007.
20. Neal, R. M., Regression and classification using Gaussian process priors, *Bayesian Stat.*, vol. **6**, pp. 475–501, 1998.
21. Neal, R. M., MCMC using Hamiltonian dynamics, In *Handbook of Markov Chain Monte Carlo*, Boca Raton, FL: Chapman and Hall/CRC Press, pp. 113–162, 2011.
22. Neal, R. M., Slice sampling, *Ann. Stat.*, vol. **31**, pp. 705–741, 2003.
23. Zuev, K. M. and Beck, J. L., Global optimization using the asymptotically independent Markov sampling method, *Comput. Struct.*, vol. **126**, pp. 107–119, 2013.
24. Taflanidis, A. A. and Beck, J. L., Stochastic subset optimization for optimal reliability problems, *Probab. Eng. Mech.*, vol. **23**, no. 2, pp. 324–338, 2008.
25. Taflanidis, A. A. and Beck, J. L., An efficient framework for optimal robust stochastic system design using stochastic simulation, *Comput. Methods Appl. Mech. Eng.*, vol. **198**, no. 1, pp. 88–101, 2008.
26. Garbuno-Inigo, A., DiazDelaO, F. A., and Zuev, K. M., Gaussian process hyper-parameter estimation using Parallel Asymptotically Independent Markov Sampling, *Comp. Stat. Data Anal.*, vol. **103**, pp. 367–383, 2016.
27. Birge, J. R. and Polson, N. G., Optimisation via Slice Sampling, *arXiv preprint*, pp. 1–22, 2012.
28. Ranjan, P., Haynes, R., and Karsten, R., A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data, *Technometrics*, vol. **53**, no. 4, pp. 366–378, 2011.
29. Murray, I., Adams, R. P., and MacKay, D. J., Elliptical slice sampling., In *Proc of 13th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, vol. **13**, pp. 541–548, 2010.
30. Berger, J. O. and Bernardo, J. M., On the development of reference priors, *Bayesian Stat.*, vol. **4**, no. 4, pp. 35–60, 1992.
31. Berger, J. O., Bernardo, J. M., and Sun, D., The formal definition of reference priors, *Ann. Stat.*, vol. **37**, no. 2, pp. 905–938, 2009.
32. Andrianakis, Y. and Challenor, P., Parameter estimation for Gaussian process emulators, *Technical Report, Managing Uncertainty in Complex Models*, 2011.
33. Paulo, R., Default priors for Gaussian processes, *Ann. Stat.*, vol. **33**, no. 2, pp. 556–582, 2005.
34. Oakley, J., Eliciting Gaussian process priors for complex computer codes, *J. R. Stat. Soc., Ser. D*, vol. **51**, no. 1, pp. 81–97, 2002.
35. Vernon, I., Goldstein, M., and Bower, R. G., Galaxy formation: a Bayesian uncertainty analysis, *Bayesian Analysis*, vol. **5**, no. 4, pp. 619–669, 2010.

36. MacKay, D. J. C., Hyperparameters: Optimize, or integrate out?, In Heidbreder, G. R., Ed., *Maximum, Entropy and Bayesian Methods*, Dordrecht: Kluwer Academic, pp. 43–59, 1996.

37. De Oliveira, V., Objective Bayesian analysis of spatial data with measurement error, *Can. J. Stat.*, vol. **35**, no. 2, pp. 283–301, 2007.

38. Murray, I. and Adams, R. P., Slice sampling covariance hyperparameters of latent Gaussian models, *Adv. Neural Inf. Process. Syst.*, vol. **23**, pp. 1732–1740, 2010.

39. Beck, J. and Zuev, K. M., Asymptotically Independent Markov Sampling: a new MCMC scheme for Bayesian Inference, *Int. J. Uncertainty Quant.*, vol. **3**, no. 5, pp. 445474, 2013.

40. Katafygiotis, L. and Zuev, K., Estimation of small failure probabilities in high dimensions by adaptive linked importance sampling, In *Proc of ECCOMAS Thematic Conf.*, pp. 1–12, 2007.

41. Del Moral, P., Doucet, A., and Jasra, A., Sequential Monte Carlo samplers, *J. R. Stat. Soc. B*, vol. **68**, no. 3, pp. 411–436, 2006.

42. Fearnhead, P. and Taylor, B. M., An adaptive sequential Monte Carlo sampler, *Bayesian Analysis*, vol. **8**, no. 2, pp. 411–438, 2013.

43. Del Moral, P., Doucet, A., and Jasra, A., On adaptive resampling strategies for sequential Monte Carlo methods, *Bernoulli*, vol. **18**, no. 1, pp. 252–278, 2012.

44. Gelman, A., Roberts, G., and Gilks, W., Efficient metropolis jumping rules, *Bayesian Stat.*, vol. **5**, pp. 599–608, 1996.

45. Del Moral, P. and Jasra, A., Sequential Monte Carlo for Bayesian computation, *Bayesian Stat.*, vol. **8**, pp. 1–34, 2007.

46. Andrieu, C., Doucet, A., and Holenstein, R., Particle Markov chain Monte Carlo methods, *J. R. Stat. Soc., Ser. B*, vol. **72**, no. 3, pp. 269–342, 2010.

47. Haaland, B. and Qian, P. Z., Accurate emulators for large-scale computer experiments, *Ann. Stat.*, vol. **39**, no. 6, pp. 2974–3002, 2011.

48. Nilson, T. and Kuusk, A., A reflectance model for the homogeneous plant canopy and its inversion, *Remote Sens. Environ.*, vol. **27**, no. 2, pp. 157–167, 1989.

49. Bastos, L. S. and O'Hagan, A., Diagnostics for Gaussian process emulators, *Technometrics*, vol. **51**, no. 4, pp. 425–438, 2009.

50. Neal, R. M., Monte Carlo implementation of Gaussian process models for Bayesian regression and classification, *Technical Report, University of Toronto*, pp. 1–24, 1997.

51. Gramacy, R. B. and Polson, N. G., Particle learning of Gaussian process models for sequential design and optimization, *J. Comput. Graph. Stat.*, vol. **20**, no. 1, pp. 1–18, 2009.

52. Kohonen, J. and Suomela, J., Lessons learned in the challenge: Making predictions and scoring them, In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Berlin: Springer, vol. **3944**, pp. 95–116, 2006.

53. Grimit, E. P., Gneiting, T., Berrocal, V. J., and Johnson, N. A., The continuous ranked probability score for circular variables and its application to mesoscale forecast ensemble verification, *Q. J. R. Meteorol. Soc.*, vol. **132**, no. 621C, pp. 2925–2942, 2006.

54. Hertog, D. d., Kleijnen, J. P., and Siem, A., The correct Kriging variance estimated by bootstrapping, *J. Oper. Res. Soc.*, vol. **57**, no. 4, pp. 400–409, 2006.