Transitional annealed adaptive slice sampling for Gaussian process hyper-parameter estimation

A. Garbuno-Inigo[a,*], F. A. DiazDelaO[a], K. M. Zuev[b]

[a]*Institute for Risk and Uncertainty, School of Engineering, University of Liverpool*
Brownlow Hill, Liverpool, L69 3GH, United Kingdom
[b]*Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA.*

Abstract

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 optimisation 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, through combining Slice Sampling with some recently developed Sequential Monte Carlo samplers. The resulting algorithm improves the mixing in the sampling through delayed-rejection, the inclusion of an annealing scheme akin to Asymptotically Independent Markov Sampling and parallelisation via Transitional Markov Chain Monte Carlo. Examples related to the estimation of hyper-parameters, as well as examples applied in other contexts of Bayesian inference, 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.

Keywords:
Gaussian process, hyper-parameter, marginalisation, optimisation, slice sampling, simulated annealing.

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 optimisation, uncertainty analysis and sensitivity analysis [Forrester et al., 2008, Kennedy and O'Hagan, 2001a]. 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 non-parametric hierarchical structure. Common applications of Gaussian processes are found, amongst many other areas, in Machine Learning [Rasmussen and Williams, 2006], Spatial Statistics [Cressie, 1993] (with the name of Kriging), likelihood-free Bayesian Inference [Wilkinson, 2014], Genetics [Kalaitzis and Lawrence, 2011] and Engineering [DiazDelaO and Adhikari, 2011].

---

*Corresponding author
Email address:* agarbuno@liv.ac.uk (A. Garbuno-Inigo)

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 in an inadequate explanation of the uncertainty of the model parameters by uni-modal distributions. In these cases, one is able to acknowledge all uncertainties related to the modelling assumptions by resorting to Model Uncertainty Analysis [Draper, 1995]. More specifically, *hierarchical modelling* should be considered in the modelling assumptions. By doing so, the analyst is capable of accounting for structural uncertainty, and which be considered as a continuous or discrete construct [see Draper, 1995, §4]. In Gaussian processes models, continuous structural uncertainty can be incorporated through the Bayesian paradigm. In this work, the approach for dealing with different models which are originated from multi-modal distributions of the model's parameters is to employ samplers specifically built for such scenarios.

The implementation of a Gaussian process emulator requires a training phase. This involves the estimation of the parameters of the Gaussian process, referred to as *hyper-parameters*. The selection of the hyper-parameters is usually done by Maximum Likelihood estimates (MLE) [Forrester et al., 2008], Maximum a Posteriori estimates (MAP) [Oakley, 1999, Rasmussen and Williams, 2006], or by sampling from the posterior distribution [Williams and Rasmussen, 1996] in a fully Bayesian manner. It is frequently the case that estimating the hyper-parameters depends on maximising a complex, multimodal function. In this scenario, traditional optimisation routines [Nocedal and Wright, 2004] are not able to guarantee global optima when looking for the MLE or MAP, and a Bayesian treatment becomes a suitable option to account for all the uncertainties in the modelling. In the literature however, it is common that either the MLE or MAP alternatives are preferred [Kennedy and O'Hagan, 2001a, Gibbs, 1998] due to the numerical burden of maximising the likelihood function or because it is assumed that Bayesian integration (through for example, Markov chain Monte Carlo methods) is prohibitive. Although these are strong arguments in favour 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, measured with prediction-oriented metrics such as the root-mean-square error (RMSE) [Kennedy and O'Hagan, 2001b]. This 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 [Kennedy and O'Hagan, 2001a]. Therefore, a suitable approach is to perform Monte Carlo integration [MacKay, 1998]. This allows to approximate any integral by means of a weighted sum, given a sample from the *correct* distribution.

The Gaussian process model specification yields no direct simulation of its 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 no be appropriate for multi-modal distributions [Neal, 2001, Hankin, 2005]. 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 mass of the probability is concentrated. One of the most recognised concerns in simulation is to avoid *Random Walk* behaviour, since it delays the stationarity state achieved by the chain and limits its exploration capabilities [Neal, 1993]. An optimal tuning of the proposal distribution in high-dimensional spaces with intricate correlation among sets of variables can turn into a demanding task yielding MCMC samplers expensive [Ching and Chen, 2007]. The alternative in Gaussian processes, among other probabilistic applications, has been to resort to the Hybrid Monte Carlo (HMC) sampler, as it can avoid Random Walk behaviour at expense of additional computations needed for the gradient of the posterior [Neal, 1998, Williams and Rasmussen, 1996]. However, there is no guarantee that multi-modal distributions can be sampled thoroughly by HMC [Neal, 2011] and in Gaussian process models the gradients are not available in $\mathcal{O}(n)$ operations.

This paper proposes a sampling scheme for global optimisation based on two principles. Firstly, the concept of *crumb* introduced by Neal [2003] for a multivariate adaptive slice sampler. Secondly, the ideas by Zuev and Beck [2013] on how to simulate local approximations to the solution based on a sequence of nested

subsets as in Stochastic Subset Optimisation [Taflanidis and Beck, 2008a,b]. The use of delayed rejection in the Asymptotically independent Markov sampler [Garbuno-Inigo et al., 2015] has proven to enhance the mixing capabilities of the algorithm 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 the annealing schedule, as opposed to being tuned by trial and error [Birge and Polson, 2012]. 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 [Ching and Chen, 2007] 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 estimate the hyper-parameters of a Gaussian process, the resulting emulator is built taking into account both a probabilistic and an optimisation perspective. The probabilistic strategy is to treat the problem in a Bayesian manner, which accounts for the uncertainty that stems from the unknown parameters. This adds a layer of structural uncertainty to the model. The optimisation strategy is to treat the problem in a framework which enables to embed such uncertainty in a set of suboptimal approximations. Additionally, model uncertainty is accounted for by adding numerical stabilisation measures in the Gaussian process model as in Ranjan et al. [2011], Andrianakis and Challenor [2012] in a fully Bayesian framework.

The paper is organised as follows. In Section 2, a brief introduction to the Bayesian treatment of Gaussian processes, as well as related numerical stabilisation 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 simulator. 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 Gaussian process prior is assumed for the outputs of the simulator, then the set of design points has a joint Gaussian distribution, and the output satisfies the 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

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

where σ^2 is the signal noise and $\boldsymbol{\phi} \in \mathbb{R}_+^p$ denotes the *length-scale* parameters of the correlation function $k(\cdot, \cdot)$. 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,

3

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 [Neal, 1998, Rasmussen and Williams, 2006, Murray et al., 2009] prefer the ϕ_i^2 parameterisation 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 parameters $\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi}$ and the design points X, can be written as

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

where H is the *design matrix* whose rows are the inputs $h(\mathbf{x}_i)^\top$ and K is the correlation matrix with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\phi})$ for all $i, j = 1, \dots, n$.

2.1. Hyper-parameter estimation

The 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 unknown and random variables, robust estimators can be built through numerical integration. By marginalising 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. After this process, predictions for y^*, given a non-observed 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

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

where $\boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi})$ denotes the vector of hyper-parameters. Note that the model used for predictions y^*, given the data and $\boldsymbol{\theta}$, is a Gaussian random variable, which makes the model closed under the Gaussian family [see Oakley, 1999]. Additionally, note that each possible specification of $\boldsymbol{\theta}$ is realisation of a Gaussian random variable, by which referring to $\boldsymbol{\theta}$ as the Gaussian process' hyper-parameters is appropriate.

Due to its computational complexity, numerical integration of (2.5) is often avoided. Instead, it is commonly assumed that the MLE of the likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{y}|X, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi}) \tag{2.6}$$

or the MAP estimate from the posterior distribution

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathbf{y}|X, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi})\, p(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi}) \tag{2.7}$$

are capable of taking into account all the uncertainty in the model. However, when either the likelihood (2.6) is a non-convex function or the posterior (2.7) is a multi-modal distribution, conventional optimisation routines may only find local optima. Additionally, there are degenerate cases when it is crucial to estimate the integral in (2.5) by means of Monte Carlo simulation instead of proposing a single candidate [Andrianakis and Challenor, 2012]. It is possible to approximate the integrated predictive distribution in (2.5) through

MCMC as

$$p(y^*|\mathbf{x}^*, \mathcal{D}) \approx \sum_{i=1}^{N} w_i \ p(y^*|\mathbf{x}^*, \mathcal{D}, \boldsymbol{\theta}_i), \tag{2.8}$$

where $\boldsymbol{\theta}_i$ is obtained by a sampling scheme which is capable of managing multi-modal distributions. The coefficients w_i denote the weights of each sample generated. Since each term $p(y^*|\mathbf{x}^*, \mathcal{D}, \boldsymbol{\theta}_i)$ corresponds to a Gaussian density function, the predictions are made by a mixture of Gaussians. It can be shown [Garbuno-Inigo et al., 2015] that, if an emulator with input $\mathbf{x}^*$ and output y^* has a posterior density as in (2.8), then its mean and covariance functions are

$$\mu(\mathbf{x}^*) = \sum_{i=1}^{N} w_i \ \mu_i(\mathbf{x}^*), \tag{2.9}$$

$$\mathrm{cov}(\mathbf{x}^*, \mathbf{x}') = \sum_{i=1}^{N} w_i \ \left[(\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*))(\mu_i(\mathbf{x}') - \mu(\mathbf{x}')) + \mathrm{cov}(\mathbf{x}^*, \mathbf{x}'|\boldsymbol{\theta}_i)\right], \tag{2.10}$$

where $\mu_i(\mathbf{x}^*)$ is the expected value of the likelihood distribution of y^* conditional on the hyper-parameters $\boldsymbol{\theta}_i$, the training runs $\mathcal{D}$ and the input configuration $\mathbf{x}^*$. From equation (2.10) it follows that the variance (also known as the prediction error) of an untested configuration $\mathbf{x}^*$ is

$$\sigma^2(\mathbf{x}^*) = \sum_{i=1}^{N} w_i \ ((\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*))^2 + \sigma_i^2(\mathbf{x}^*)). \tag{2.11}$$

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

A fully Bayesian treatment of the predictive posterior, equation (2.5), requires the specification of the prior distribution $p(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi})$ in equation (2.7). Weak prior distributions [Oakley, 1999] have been used for $\boldsymbol{\beta}$ and σ^2 namely

$$p(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi}) \propto \frac{p(\boldsymbol{\phi})}{\sigma^2} \tag{2.12}$$

where it is assumed that both the covariance and the mean hyper-parameters are independent. Even more, $\boldsymbol{\beta}$ and σ^2 are assumed to have non-informative distributions. For the length-scale hyper-parameter $\boldsymbol{\phi}$, the reference prior [studied by Berger and Bernardo, 1992, Berger et al., 2009] allows for an objective framework 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 [Andrianakis and Challenor, 2011], thus reducing regions of possible candidates in the mixture model proposed as in equation (2.8). Since this allows for prior distributions without concerns in expert knowledge of feasible regions for the hyper-parameters, the reference prior for Gaussian processes developed by Paulo [2005] is used in this work, unless otherwise stated. Note however that there are no known analytical expressions for its derivatives 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 regularisation in the norm of the parameters. Other alternatives suggest a decaying prior Andrianakis and Challenor [2011] or a weakly informative distribution such as a gamma with appropriate parameters. If needed, elicited priors from experts as in Oakley [2002] can be used.

2.3. Marginalising the nuisance hyper-parameters

The set of hyper-parameters of the Gaussian process may be potentially different in terms of scales and dynamics, as explained in Garbuno-Inigo et al. [2015]. By using Gibbs sampling one can possibly cope with this limitation, although it is well-known that such sampling scheme can be inefficient when used for multi-modal distributions in high-dimensional spaces. The Metropolis-Hastings sampler exhibits the same problem, which leads us to focus on ϕ and perform the inference in the correlation function. To achieve this, β and σ^2 are considered as nuisance parameters and both are integrated from the posterior (2.7). The joint distribution of the training runs and the prior distribution of the hyper-parameters, equations (2.4) and (2.12) respectively, allow us to identify a Gaussian-inverse-gamma model of β and σ^2, which after integration, can be shown to yield the integrated posterior distribution

$$p(\phi|\mathcal{D}) \propto p(\phi)\,(\hat{\sigma}^2)^{-\frac{n-p}{2}}\,|K|^{-\frac{1}{2}}\,|H^\top K^{-1}H|^{-\frac{1}{2}}, \tag{2.13}$$

where

$$\hat{\sigma}^2 = \frac{\mathbf{y}^\top \left(K^{-1} - K^{-1}H(H^\top K^{-1}H)^{-1}H^\top K^{-1}\right)\mathbf{y}}{n-p-2} \tag{2.14}$$

and

$$\hat{\beta} = (H^\top K^{-1}H)^{-1}H^\top K^{-1}\mathbf{y} \tag{2.15}$$

can be considered as estimators of the signal noise σ^2 and regression coefficients β [see Oakley, 1999, for further details]. Finally, the predictive distribution conditioned on the remaining set of hyper-parameters, ϕ, follows a t-student distribution with mean and correlation functions

$$\mu(\mathbf{x}^*|\phi) = h(\mathbf{x}^*)^\top \hat{\beta} + t(\mathbf{x}^*)^\top K^{-1}(\mathbf{y} - H\hat{\beta}), \tag{2.16}$$

$$\mathrm{corr}(\mathbf{x}^*, \mathbf{w}^*|\phi) = k(\mathbf{x}^*, \mathbf{w}^*|\phi) - t(\mathbf{x}^*)^\top K^{-1}\,t(\mathbf{w}^*) +$$
$$\left(h(\mathbf{x}^*)^\top - t(\mathbf{x}^*)^\top K^{-1}H\right)(H^\top K^{-1}H)^{-1}\left(h(\mathbf{w}^*)^\top - t(\mathbf{w}^*)^\top K^{-1}H\right)^\top, \tag{2.17}$$

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 $t(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1|\phi), \ldots, k(\mathbf{x}, \mathbf{x}_n|\phi))^\top$. Note that both estimators depend solely on the correlation function hyper-parameters ϕ. MacKay [1996] has previously discussed the treatment of nuisance parameters and integrated posteriors. In the context of Gaussian process emulators, this discussion allows to reduce the dimensionality of the problem and overcome the limitations of samplers when faced to different dynamics posed by each subset of hyper-parameters.

In light of the above, this paper focuses on the correlation function $k(\cdot, \cdot)$ in equation (2.2), since the structural dependencies of the training runs to allow prediction in the outputs of the simulator is recovered by it. The key aspect is that the trend function, as its hyper-parameter, contains minor structural information of the data, when compared to the length-scale hyper-parameters. If this were not the case it would prevent the use of integrated likelihoods [see Berger et al., 1999, for further discussion]. Nonetheless, if global trend information is available, then an additional effort can be made on eliciting an appropriate mean function for the emulator. Such expert knowledge of the simulator would allow the analyst to model better the mean function by adding significant terms [see Vernon et al., 2010, for a detailed discussion].

2.4. Numerical stability

Numerical stability is commonly difficult to guarantee when implementing Gaussian processes. As explored previously by Andrianakis and Challenor [2012] and Ranjan et al. [2011] a term in the covariance matrix can

be added in order to preserve diagonal dominance, that is, to add a *nugget* hyper-parameter $\boldsymbol{\phi}_\delta$ such that

$$K_\delta = K + \boldsymbol{\phi}_\delta\, I, \tag{2.18}$$

is positive definite. The result is a stochastic simulator

$$y_i = \eta(\mathbf{x}_i) + \sigma^2\, \boldsymbol{\phi}_\delta, \tag{2.19}$$

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. In such scenario, the use of multi-modal oriented samplers is crucial for performing the Monte Carlo approximation in equation (2.8). In case that the resulting model is assessed as not appropriate, a regularisation term (an elicited prior) can be added to penalise such regions [Andrianakis and Challenor, 2012].

The nugget term $\boldsymbol{\phi}_\delta$ is incorporated as a hyper-parameter of the correlation function. As previously discussed by Ranjan et al. [2011], a uniform prior distribution $U(10^{-12}, 1)$ is used for our framework. 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 equation (2.18), the previous assumptions regarding the original noise parameter σ^2 remain unchanged, as one can still marginalise it as a nuisance parameter with the non-informative prior used [De Oliveira, 2007].

3. Slice sampling

The slice sampling algorithm [Neal, 2003] is a method to simulate a Markov chain of a random variable $\boldsymbol{\theta} \in \Theta$. 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 $\boldsymbol{\theta}$ is recovered by disregarding the values of u in the Markov chain, a consequence of defining an appropriate conditional distribution for u, given $\boldsymbol{\theta}$. The samples are generated by an iterative Gibbs sampling schedule to recover pairs $\{(\boldsymbol{\theta}_i, u_i)\}_{i=1}^N$, which follow a joint density probability distribution

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

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

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

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

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

where Z is the normalising constant for the target, that is $Z = \int_\Theta f(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ and the previous considerations for the marginal of $\boldsymbol{\theta}$ follow. It should be noted that floating-point underflows are common in the implementation of Gaussian processes due to ill-conditioning of the matrix K in equation (2.13). Thus, in order to compute stable evaluations of the target distribution, it is preferable to evaluate the negative logarithm of the target density. In such case, equation(3.2) can be computed as stated in the following proposition.

Proposition 1. *Given the state of the Markov chain $\boldsymbol{\theta}_0$, the uniform distribution for the next candidate has support in the slice given by*

$$S_{\boldsymbol{\theta}_0} = \{\boldsymbol{\theta} : z > \mathcal{H}(\boldsymbol{\theta})\}, \tag{3.4}$$

where $\mathcal{H}(\cdot)$ denotes the negative log-target density and $z = \mathcal{H}(\boldsymbol{\theta}_0) + e$, 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 $\boldsymbol{\theta}_0$ of the Markov chain, the auxiliary uniform random variable defining the slice can be written as the product $u\,f(\boldsymbol{\theta}_0)$ with u uniformly distributed in the interval $(0,1)$. Thus, the slice is defined as

$$\begin{aligned}
S_{\boldsymbol{\theta}_0} &= \{\boldsymbol{\theta} : u\,f(\boldsymbol{\theta}_0) < f(\boldsymbol{\theta})\} \\
&= \{\boldsymbol{\theta} : -\log(f(\boldsymbol{\theta}_0)) - \log(u) > -\log(f(\boldsymbol{\theta}))\} \\
&= \{\boldsymbol{\theta} : \mathcal{H}(\boldsymbol{\theta}_0) + e > \mathcal{H}(\boldsymbol{\theta})\}, \tag{3.5}
\end{aligned}$$

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 on the slice. In one-dimensional applications, the slice can be defined by several methods. The canonical example is a stepping-out and shrinkage procedure which aims to extend an initial interval until the edges are outside the slice and then shrinking it as samples generated from the interval are rejected [see Neal, 2003, 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, slow dramatically as the dimension of the problem increases. This is due to the generalisation 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 the case of Gaussian process emulators, the task of evaluating the target density becomes expensive if multiple evaluations are needed for each state in the Markov chain. Therefore, other alternatives are preferable. In this paper, we employ a framework proposed by [Neal, 2003] 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. The algorithm starts with a given candidate $\boldsymbol{\theta}_0$ and a slice defined by a uniform random variable u in the interval $(0, f(\boldsymbol{\theta}_0))$. The first crumb, denoted by ς_1, is drawn from a density distribution conditioned on the slice and the current point, that is $g_1(c|u, \boldsymbol{\theta}_0)$. Let $\boldsymbol{\theta}_1^*$ be the first candidate drawn from the proposal density $h_1(\boldsymbol{\theta}^*|u, \varsigma_1) \propto g_1(\varsigma_1|u, \boldsymbol{\theta}^*)$. If the candidate is on the slice then it becomes the next state of the Markov chain. If it is rejected, a second crumb is generated from a density $g_2(c|\boldsymbol{\theta}_0, u, \varsigma_1, \boldsymbol{\theta}_1^*)$, which may depend on the previous rejected sample and crumb, as well as on the current state of the chain and slice level. The next candidate is then drawn from a density $h_2(\boldsymbol{\theta}^*|u, \varsigma_1, \boldsymbol{\theta}_1^*, \varsigma_2) \propto g_1(\varsigma_1|u, \boldsymbol{\theta}^*)\,g_2(\varsigma_2|\boldsymbol{\theta}^*, u, \varsigma_1, \boldsymbol{\theta}_1^*)$. If the second candidate is inside the slice then a Gibbs update is performed to the slice level and the procedure repeats with the new state of the Markov chain. However, if the candidate is rejected then a third crumb is generated following the procedure described above, conditioned on the previous crumbs generated, the current state of the chain and the rejected samples. It can be proved [Neal, 2003] that this procedure generates a Markov chain leaving the target distribution $f(\boldsymbol{\theta})$ invariant, leading to proposal distributions more concentrated as the number of crumbs increases. This is due to the fact that the procedure can be interpreted as generating samples from pseudo-posterior distributions based on pseudo-data (the crumbs).

4. Transitional annealed adaptive slice sampling

In order to marginalise the posterior predictive distribution in equation (2.5), Monte Carlo integration is usually performed when aiming at a fully Bayesian treatment. This is usually done by Hybrid Monte Carlo [Neal, 1998, Williams and Rasmussen, 1996] which is capable of suppressing the Random Walk behaviour 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 [Murray et al., 2009] provides a framework for the simulation of the hyper-parameters of a Gaussian process with little tuning required from the analyst [Murray and Adams, 2010]. However, this is only applicable when the posterior predictive distribution for the hyper-parameters is of the form

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

where $p(\boldsymbol{\theta})$ denotes the prior distribution, $\mathcal{N}(\cdot|\cdot,\cdot)$ is a Gaussian distribution and $h(\cdot)$ is a latent variable that depends on the hyper-parameters. As it can be seen from the integrated posterior in equation (2.13), this form is not valid in the formulation followed. The difference is a full regression model assumed in the mean function of the Gaussian process. Note that by marginalising the regression coefficients and signal noise of the Gaussian process, one aims at reducing the dimensionality of the problem as well as coping with different scales for different sets of hyper-parameters.

In this setting, we propose Transitional Annealed Adaptive Slice Sampling (TA^{2}S^2), which can be used in other applications of Bayesian inference or Stochastic optimisation. Based on Asymptotically Independent Markov Sampling [Beck and Zuev, 2013] we formulate the problem to be solved as reminiscent of simulated annealing. We want to find the solution to the optimisation problem

$$\min_{\boldsymbol{\phi}\in\Phi} \mathcal{H}(\boldsymbol{\phi}|\mathcal{D}), \tag{4.2}$$

where $\mathcal{H}(\boldsymbol{\phi}|\mathcal{D})$ denotes the negative integrated log-posterior distribution of the length-scale hyper-parameters, given the set of training runs $\mathcal{D}$. Let the set of optimal solutions to the problem above be denoted by

$$\Phi^* = \left\{ \boldsymbol{\phi}\in\Phi \,:\, \boldsymbol{\phi} = \arg\min_{\boldsymbol{\phi}\in\Phi} \mathcal{H}(\boldsymbol{\phi}|\mathcal{D}) \right\}, \tag{4.3}$$

with $|\Phi^*| \geq 1$. Note that this formulation of the problem allows for the existence of multiple global optimisers. The algorithm provides a sequence of nested subsets $\Phi_{k+1} \subseteq \Phi_k$ converging to the set of optimal solutions Φ^*. That way, if the algorithm is terminated in an intermediate annealing level, a sample of local approximations to the global minimiser can be recovered. The annealing is done by tempering the target distribution using a sequence of density functions as in simulated annealing. The temperature is learned through an automatic mechanism to determine the sequence of distributions. Let $\{p_k(\boldsymbol{\phi}|\mathcal{D})\}_{k=1}^{\infty}$ be the sequence of density distributions in the annealing schedule such that

$$p_k(\boldsymbol{\phi}|\mathcal{D}) \propto p(\boldsymbol{\phi}|\mathcal{D})^{1/\tau_k} = \exp\left\{-\mathcal{H}(\boldsymbol{\phi}|\mathcal{D})/\tau_k\right\}, \tag{4.4}$$

for a sequence of monotonically decreasing temperatures τ_k converging to zero. By construction, the sample in the first level of annealing is distributed uniformly on a *practical support* of the sampling space [see Katafygiotis and Zuev, 2007, for more a detailed discussion]. For the limiting case, the samples are uniformly distributed in the set of optimal solutions. Both these observations can be summarised by

$$\lim_{\tau\to\infty} p_\tau(\boldsymbol{\phi}|\mathcal{D}) = U_\Phi(\boldsymbol{\phi}), \tag{4.5}$$

$$\lim_{\tau\to 0} p_\tau(\boldsymbol{\phi}|\mathcal{D}) = U_{\Phi^*}(\boldsymbol{\phi}), \tag{4.6}$$

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

4.1. Annealing at level k

This subsection focuses on the sampling carried out by TA^2S^2 at the k-th level of the annealing sequence. It is therefore 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.

Proposition 2. *The slice defined in the k-th annealing level, given the current state of the Markov chain ϕ_0, is given by*

$$S_{\phi_0}^k = \{\phi : z_k > \mathcal{H}(\phi|\mathcal{D})\}, \tag{4.7}$$

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

Proof. The result follows directly from the definition of the target distribution at the annealing level k in equation (4.4) and as direct consequence from Proposition 1. ∎

As in other Sequential Monte Carlo algorithms [Del Moral et al., 2006, Fearnhead and Taylor, 2013], let us define the importance weights of the samples $\phi_1^{(k-1)}, \ldots, \phi_N^{(k-1)}$ as

$$\omega_j^{(k-1)} = \frac{p_k\left(\phi_j^{(k-1)}\right)}{p_{k-1}\left(\phi_j^{(k-1)}\right)} \propto \exp\left\{-\mathcal{H}\left(\phi_j^{(k-1)}|\mathcal{D}\right)\left(\frac{1}{\tau_k} - \frac{1}{\tau_{k-1}}\right)\right\}, \tag{4.8}$$

$$\overline{\omega}_j^{(k-1)} = \frac{\omega_j^{(k-1)}}{\sum_{j=1}^N \omega_j^{(k-1)}}, \tag{4.9}$$

where $\omega_j^{(k-1)}$ denotes the importance weights and $\overline{\omega}_j^{(k-1)}$ the normalised importance weights. The weights allow us 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 ϕ_0, is generated as follows. A slice is obtained as in equation (4.7) by generating an exponential random variable with mean τ_k, thus defining the slice $S_{\phi_0}^k$ 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

$$\mathcal{J} = \left\{j \in \{1, \ldots, N\} : \phi_j^{(k-1)} \in S_{\phi_0}^k\right\}. \tag{4.10}$$

The points $\phi_1^{(k-1)}, \ldots, \phi_N^{(k-1)}$ are uniformly distributed in the approximation set Φ_{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 increased too rapidly. A fail-safe can be used by generating a crumb from a wide Gaussian distribution centred at the current state of the Markov chain. Additionally, as it is done in other Sequential Monte Carlo methods [Del Moral et al., 2012], 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_{renew}, the crumb ς_1 will be distributed as

$$\varsigma_1 \sim \mathcal{N}(\phi_0, c_0^2\, \Sigma_k), \tag{4.11}$$

where c_0 is a spread parameter associated with the annealing sequence, and Σ_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 $\text{diag}\{d_1, \ldots, d_p\}$ 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, we define Σ_k as the weighted covariance matrix from the weighted samples $\{(\overline{\omega}_j^{(k-1)}, \boldsymbol{\phi}_j^{(k-1)})\}_{j=1}^N$. As discussed by [Gelman et al., 1996], 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

$$\boldsymbol{\xi}_1 \sim \mathcal{N}(\varsigma_1, c_0^2\, \Sigma_k), \tag{4.12}$$

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

$$\boldsymbol{\xi}_i \sim \mathcal{N}\left(\overline{\varsigma}_i, \left(\frac{c_0}{i}\right)^2 \Sigma_k\right), \tag{4.13}$$

where $\overline{\varsigma}_i$ is the average of the crumbs generated so far, as proposed by [Neal, 2003]. 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 multi-modal distribution. To cope with this limitation, we propose to use a weighted average of the current state and the crumb centre to enhance the mixing of the sampler. Namely, by sampling the i-th candidate from a Gaussian distribution with mean

$$\overline{\varsigma}_i^* = \alpha_i\, \boldsymbol{\phi}_0 + (1 - \alpha_i)\, \overline{\varsigma}_i \tag{4.14}$$

and covariance $(c_0/i)\, \Sigma_k$. The weight parameter α_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, α_i is linearly-dependent on the crumb iteration, since the exponential behaviour exhibits pronounced decay towards the current state, causing Random Walk behaviour. The sampling in each annealing level is summarised in Algorithm 1. To avoid cluttered notation,

the conditioning on the design points $\mathcal{D}$ will be dropped in the remainder.

Algorithm 1: TA^{2}S^2 at annealing level k

Input :
$\diamond$ $\boldsymbol{\phi}_1^{(k-1)}, \ldots, \boldsymbol{\phi}_N^{(k-1)} \sim p_{k-1}(\boldsymbol{\phi})$, generated at previous level;

$\diamond$ $\boldsymbol{\phi}_1^{(k)} \in \Phi$, initial state of the chain;

Output :
$\diamond$ $\boldsymbol{\phi}_1^{(k)}, \ldots, \boldsymbol{\phi}_N^{(k)} \sim p_k(\boldsymbol{\phi})$;

1 begin
2   Compute covariance matrix Σ_k from the weighted samples;
3   **for** $i \leftarrow 2$ **to** $n-1$ **do**
4   Define slice $S_{\boldsymbol{\phi}_i}^k$ as in (4.7) ;
5 $l \leftarrow 0$;
6   **do**
7   Increase l and generate $u \sim U(0,1)$;
8   **if** $\lfloor \mathcal{J} \rfloor \neq \emptyset$ **or** $u < p_{\mathrm{renew}}$ **then**
9   Choose random j from index set $\mathcal{J}$;
10 $\varsigma_l = \boldsymbol{\phi}_j^{(k-1)}$;
11   **else**
12   Generate $\varsigma_l \sim \mathcal{N}(\boldsymbol{\phi}_i^{(k)}, c_0^2 \Sigma_k)$;
13   **end**
14   Define crumb as $\overline{\varsigma}_l^* = \alpha_l \boldsymbol{\phi}_i^{(k)} + (1 - \alpha_l) \overline{\varsigma}_l$;
15   Generate candidate $\boldsymbol{\xi}_i \sim \mathcal{N}(\boldsymbol{\phi}_i^{(k)}, (c_0/l)^2 \Sigma_k)$;
16   **while** $\boldsymbol{\xi}_i \notin S_{\boldsymbol{\phi}_i}^k$;
17   Define new state of the chain $\boldsymbol{\phi}_{i+1}^{(k)} = \boldsymbol{\xi}_i$;
18   **end**
19 end

4.2. Overview of the full sampler

The algorithm starts with a uniform sample in an admissible space Φ, as implied by the *meta*-prior distribution in equation (4.5). 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 is the first approximation to the solution of the optimisation problem. As the sequence of temperatures converges to zero, we expect to recover better approximations until a desired sample is simulated. Thus, until a sample $\boldsymbol{\phi}_1^{(k^*)}, \ldots, \boldsymbol{\phi}_N^{(k^*)}$ has been drawn and is uniformly distributed in the set Φ^*. We also need to determine a stopping criterion for the sampler, as well as defining 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 subsequence of the annealing levels is redundant for the approximation of the solution. Setting the temperature sequence beforehand requires prior knowledge of the overall behaviour of the objective function $\mathcal{H}(\cdot)$ and the topology around the solution set Φ^*, both of which are generally not available.

Following the suggestion by Zuev and Beck [2013], the *Effective Sampling Size* can be used as a measure of degeneracy of the chain in each annealing level [see Zuev and Beck, 2013, for further discussion]. This allows to measure how similar the $(k-1)$-th and the k-th densities are. The effective sample size can be approximated by

$$\hat{n}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\overline{\omega}_j^{(k-1)} \right)^2}, \tag{4.15}$$

where $\overline{\omega}_j^{(k-1)}$ is the normalised weight of sample $\phi_j^{(k-1)}$. Given 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 unnormalised sample weights we obtain

$$\frac{\sum_{j=1}^{N} \exp \left\{ -2\mathcal{H} \left(\phi_j^{(k-1)} \right) \left(\frac{1}{\tau_k} - \frac{1}{\tau_{k-1}} \right) \right\}}{\left(\sum_{j=1}^{N} \exp \left\{ -\mathcal{H} \left(\phi_j^{(k-1)} \right) \left(\frac{1}{\tau_k} - \frac{1}{\tau_{k-1}} \right) \right\} \right)^2} = \frac{1}{\gamma N}, \tag{4.16}$$

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

The value of the threshold γ 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 γ is close to one, then there will be excessive tempered distributions and redundant annealing levels. As suggested by Beck and Zuev [2013], and as confirmed by our experiments, a value of $\gamma = 0.5$ delivers acceptable efficiency.

4.2.2. Stopping criterion

In theory, the algorithm should run with k increasing until the temperature $\tau_k \to 0$, resulting in uniformly distributed samples in the set Φ^*. However, in practical implementations, the absolute zero cannot be achieved and a stopping criterion for the annealing sequence is necessary. In this situation, the sample coefficient of variation (COV) of the objective function $\mathcal{H}(\cdot)$ can serve as a stopping criterion. Without loss of generality, it is assumed that the objective $\mathcal{H}(\cdot)$ is a non-negative function [see Zuev and Beck, 2013, for a detailed discussion]. Let δ_k denote the sample COV of the simulations $\mathcal{H}\left(\phi_1^{(k)}\right), \ldots, \mathcal{H}\left(\phi_N^{(k)}\right)$, that is

$$\delta_k = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{H}\left(\phi_i^{(k)}\right) - \frac{1}{N} \sum_{j=1}^{N} \mathcal{H}\left(\phi_j^{(k)}\right) \right)^2}}{\frac{1}{N} \sum_{j=1}^{N} \mathcal{H}\left(\phi_j^{(k)}\right)}. \tag{4.17}$$

As noted in [Garbuno-Inigo et al., 2015] and [Zuev and Beck, 2013] using δ_k yields a measure of the sensitivity of the objective (the integrated log-posterior distribution) to the hyper-parameters, in the domain induced by the annealing temperature τ_k. If the set of approximated solutions $\phi_1^{(k)}, \ldots, \phi_N^{(k)}$ lie in Φ^*, then the sample COV will equal 0 exactly, since for all j, $\mathcal{H}(\phi_j^{(k)}) = \min_{\phi \in \Phi^*} \mathcal{H}(\phi)$. Thus, the stopping criterion is defined when a threshold of the initial sample COV has been reached, that is

$$\delta_k < \alpha \, \delta_0, \tag{4.18}$$

where $\alpha \in (0,1)$ defines the target threshold attained by the sampling algorithm in the last annealing level. The value of α directly affects the accuracy of the approximation achieved by the sampling algorithm. In the context of Gaussian process emulators, our experiments suggest that $\alpha = 0.1$ yields good approximations recovering robust configurations of the surrogate model for error prediction and predictions made through

equations (2.9) and (2.11).

4.2.3. 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 (2.13) involved in the computation of the Gaussian process hyper-parameter posterior distribution. 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 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 Kennedy and O'Hagan, 2001a, for a discussion]. The idea of parallelisation comes from an adaptation of the Transitional Markov Chain Monte Carlo (TMCMC) method [Ching and Chen, 2007] 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 [Del Moral and Jasra, 2007] and Particle Filtering [Andrieu et al., 2010]. 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 equation (4.8). 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.

Summarising, the proposed TA^2S^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 criterion of Section 4.2.2 is met. The selection of the initial states of the Markov chains and their growth length is a direct implementation of the TMCMC method [Ching and Chen, 2007] for Bayesian model updating.

5. Numerical experiments

The following examples illustrate the effectiveness and robustness of TA^2S^2 when sampling the hyper-parameters of Gaussian process surrogates. The first example is Franke's function [Haaland and Qian, 2012], wich can have challenging features when emulated. The second example is a five-dimensional model [Nilson and Kuusk, 1989] which has been previously used to test Gaussian processes surrogates [Bastos and O'Hagan, 2009]. The third example is a ten-dimensional model for the weight of a wing of a light aircraft [Forrester et al., 2008]. Additionally, we present two examples that illustrate the application of TA^2S^2 in other contexts. These are related to Bayesian inference, assuming a scenario where the MAP estimate is not appropriate and a more robust marginalisation is needed: first, an application to Bayesian Neural Networks [Beck and Zuev, 2013] and second, an application to Variational inference where the solution is known to have multiple local optima [Bishop, 2006]. Unless otherwise stated, it is assumed that the global behaviour of the emulator can be fitted by a simple regression term with $h(\mathbf{x})^\top = (1, x_1, \ldots, x_p)^\top$. 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 z_δ component and extend the vector of hyper-parameters ϕ to $\mathbb{R}^{p+1}$. 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 Ranjan et al. [2011]. To incorporate the algorithm to the length-scale hyper-parameters, the sampling has been performed in logarithmic space to avoid additional handling concerns for the non-negative restrictions imposed to the

aforementioned variables as in other sampling schedules [Neal, 1997]. The initial values of the algorithm, equation (4.5), 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 non-informative 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`.

5.1. Franke's function

Franke's function has been used to test Gaussian process emulators [Haaland and Qian, 2012]. Its complexity stems from presence of two peaks and one dip in its landscape. Let $f : [0, 1]^2 \to \mathbb{R}$ be such that

$$f(\mathbf{x}) = 0.75 \exp\left(-\frac{(9x_1 - 2)^2}{4} - \frac{(9x_2 - 2)^2}{4}\right) + 0.75 \exp\left(-\frac{(9x_1 + 1)^2}{49} - \frac{9x_2 + 1}{10}\right)$$
$$+ 0.5 \exp\left(-\frac{(9x_1 - 7)^2}{4} - \frac{(9x_2 - 3)^2}{4}\right) - 0.2 \exp\left(-(9x_1 - 4)^2 - (9x_2 - 7)^2\right). \tag{5.2}$$

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 that the log posterior distribution exhibits different modes. Region A contains a mode with no preference for any dimension. Regions B and C depict different asymptotic behaviours of the emulator. In region B, the emulator behaves as linear regression model, as noted by [Andrianakis and Challenor, 2012]. Region C corresponds to a model which disregards the first dimension. By applying TA^{2}S^2, we obtain robust approximations to the MAP estimate, as shown in Figure 1(b). This has been done by running the algorithm with $N = 2000$, obtaining 5 intermediate distributions. A MAP estimate of such inference problem results in a Root-mean-squared error (RMSE) of 0.1557, in contrast with the mixture estimator in equation (2.9), which achieves a RMSE of 0.1069. This is due to the MAP estimate being located in Region (B), which is a linear model of the inputs. The mixture model helps alleviate this shortcoming by introducing emulators with local dependency among the training runs.
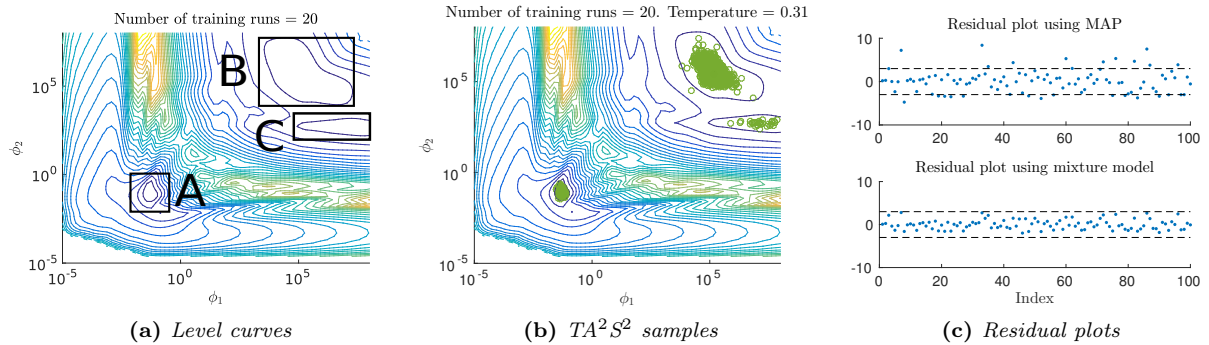

(a) *Level curves*   **(b)** *TA^{2}S^2 samples*   **(c)** *Residual plots*

Figure 1: *Projection of the negative log-posterior curves in the two dimensional length-scale space for Franke's simulator. The minimum possible value of 10^{-12} for the nugget ϕ_δ has been used for the projection.*

The estimation of the error prediction was improved as shown in the more concentrated standardised residuals in Figure 1(c). The standardised residuals are defined as

$$r(\mathbf{x}) = \frac{y - \mu(\mathbf{x})}{\sqrt{\sigma^2(\mathbf{x})}}, \tag{5.3}$$

15

where y is the simulator output of the configuration $\mathbf{x}$. Since equation $r(\mathbf{x})$ can be shown to be distributed as a Student's t Bastos and O'Hagan [2009], the residuals should lie within $[-3, 3]$. The more concentrated residuals provide evidence that by marginalising the hyper-parameters, uncertainty quantification, *e.g.* induced by structural uncertainty, is taken into account appropriately even when there is not much information to fit the surrogate model.

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 λ [see Nilson and Kuusk, 1989, 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 have been rescaled to fit the hyper-rectangle $[0, 1]^5$ as in Bastos and O'Hagan [2009].

The TA^{2}S^2 algorithm was used for an emulator with 100 training runs. The test set consisted 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`). A total of 10 tempered distributions were used in the annealing schedule, while sampling $N = 5000$ in each level. The RMSE for both the MAP and mixture is 0.0214. This seems to indicate that given the training set, the posterior distribution of the length-scales is highly concentrated around one mode. The boxplots in figure 2(a) shows the distribution of the length-scales in the five-dimensional log-space. There is strong evidence that the fifth dimension is more important and there is clearly and ordered triplet (ϕ_5, ϕ_1, ϕ_2) for the sensitivity of the output to such variables. However, there is not enough information to make a more informed judgement regarding ϕ_3 and ϕ_4. The standardised residuals confirm the assumption of little difference when considering the sensitivity of the simulator to the dimensions associated with ϕ_3 or ϕ_4.
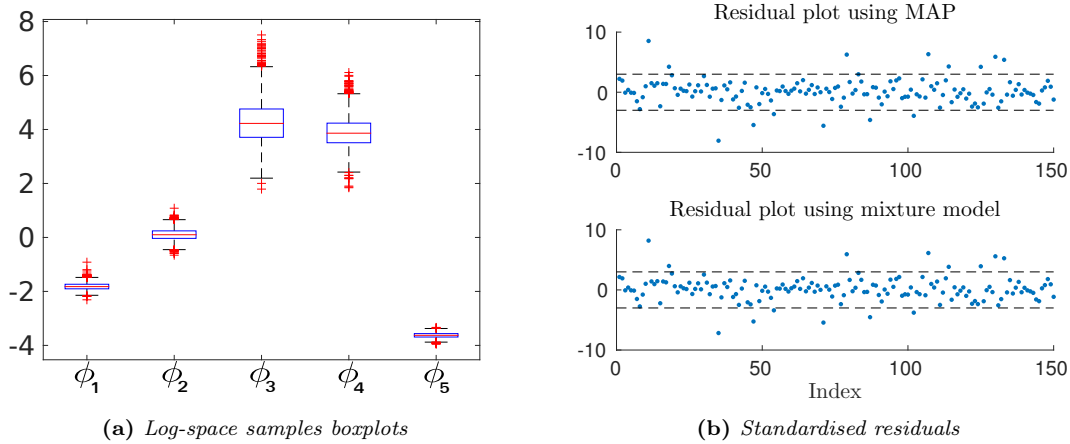

(a) *Log-space samples boxplots*

(b) *Standardised residuals*

Figure 2: *Nilson-Kuusk results obtained from the TA^{2}S^2 algorithm.*

5.3. Wing weight model

This simulator of the weight of the wing of a light aircraft [Forrester et al., 2008] has been used for input screening. Coupled with a Gaussian process emulator, a sensitivity analysis of the wing weight with respect to each input variable can be performed. The model is given by

$$f(\mathbf{x}) = 0.036\, S_w^{0.758}\, W_{fw}^{0.0035} \left(\frac{A}{\cos^2(\Lambda)} \right)^{0.6} q^{0.006}\, \lambda^{0.04} \left(\frac{100\, t_c}{\cos(\Lambda)} \right)^{-0.3} (N_z\, W_{dg})^{0.49} + S_w\, W_p, \qquad (5.4)$$

where the input variables and the range of their values are summarised in Table 1. In order to focus on determining the most relevant inputs on the correlation structure of the Gaussian process, it is assumed that $h(\mathbf{x}) = 1$. For this problem, the evaluation of the reference prior is prohibitive since, it scales with the number of dimensions [Paulo, 2005]. Thus, a uniform prior in the length-scale parameters' log-space has been used.

Input	Range	Description
S_w	$[150, 200]$	Wing area
W_{fw}	$[220, 300]$	Weight of fuel in the wing
A	$[6, 10]$	Aspect ratio
Λ	$[-10, 10]$	Quarter-chord sweep
q	$[16, 45]$	Dynamic pressure at cruise
λ	$[0.5, 1]$	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

Table 1: *Inputs of the wing weight model.*

The inputs have been rescaled to the 10-dimensional unit hypercube $[0, 1]^{10}$. Two LHS samples of size 500 were chosen as training and testing sets respectively. At each annealing level, 5000 samples were generated, achieving convergence after 15 levels. The RMSE for the MAP estimate was 3.1719, whilst 3.1918 for the mixture. This could suggest that there is little value in computing the mixture. However, the simple analysis presented in figure 3(a) not only shows that there is a subset of influential variables on the output of the model. It also reveals that the less influential of the inputs (2, 4, 5) are not concentrated around a single mode and thus the ranking of their influence on the output is not unique. This information would not be available had the mixture of Gaussian processes not been employed. The standardised residuals shown in Figure 3(b) are evidence that the posterior distribution is highly concentrated in the modes of the relevant inputs, and its application for error prediction is not affected by the variation of the remaining inputs, though the mixture model consistently narrows the spread of the residuals. By fully characterising the optimal approximations with the TA²S² algorithm, the structural uncertainty of the model is taken fully into account.
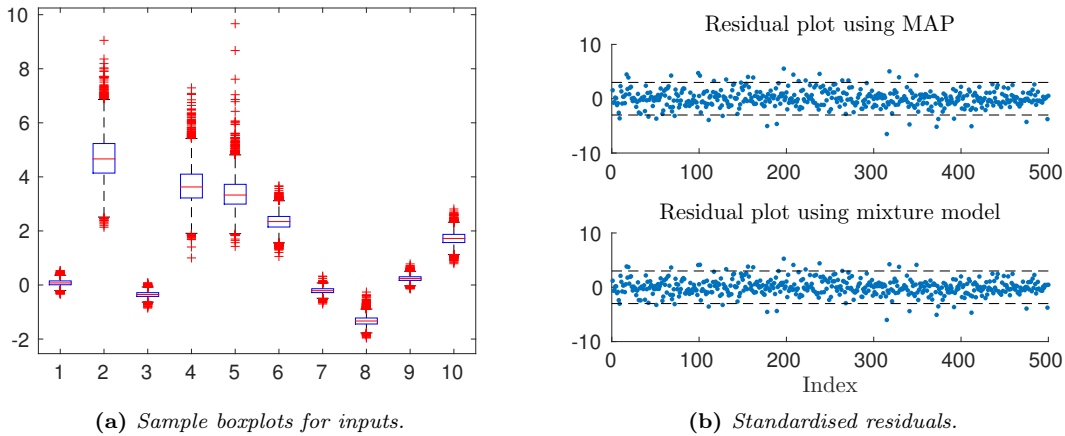

(a) *Sample boxplots for inputs.*

(b) *Standardised residuals.*

Figure 3: *Results for the wing weight model using the TA^2S^2 algorithm. The order of the inputs is consistent with the order in which they are presented in Table 1*

5.4. Other applications to Bayesian inference

In order to illustrate how the TA^{2}S^2 algorithm provides a framework that can be used for other instances of Bayesian inference we present the following two examples.

5.4.1. Bayesian Neural Network

In this example we consider a feed-forward neural network. Such models are commonly used for function approximation by providing a prediction-oriented surrogate. They have been used to approximate functions which are likely to be non-linear. The training is based on a set of observed measurements $y_i = f(\mathbf{x}_i)$ by using a linear combination of activation functions Ψ_j, that is

$$\hat{f}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j \, \Psi_j(\mathbf{x}^\top \boldsymbol{\beta} + \gamma_j), \tag{5.5}$$

where the model parameters $\boldsymbol{\alpha}, \boldsymbol{\gamma} \in \mathbb{R}^M$ and $\boldsymbol{\beta} \in \mathbb{R}^p$ are denoted by $\boldsymbol{\theta}$. Such model is known as a single hidden layer neural network with M hidden units and one output unit. The weight of the signals coming from the inputs towards the hidden units are denoted by $\boldsymbol{\beta}$. Similarly, the weight of the signals from the hidden units to the output are denoted by $\boldsymbol{\alpha}$, while the $\boldsymbol{\gamma}$ is used as an offset for each hidden unit. A more detailed discussion of the Bayesian treatment can be found on [Beck and Zuev, 2013]. We concentrate on the result that for such a surrogate model predictions are made using the expected predictive posterior. By taking expectations, that is, by marginalising the model parameters, we obtain that

$$\mathbb{E}_\pi[y, |\mathbf{x}, \mathcal{D}] = \int_\Theta \hat{f}(\mathbf{x}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}, \tag{5.6}$$

which can be approximated by Monte Carlo as

$$\mathbb{E}_\pi[y, |\mathbf{x}, \mathcal{D}] \approx \frac{1}{N} \sum_{i=1}^{N} \hat{f}(\mathbf{x}, \boldsymbol{\theta}_i), \tag{5.7}$$

by sampling from the posterior distribution $\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta}|\mathcal{D})$. For this example, we assume white noise in the output, a one dimensional input, two hidden units and $\tanh(\cdot)$ the activation function as in [Beck and Zuev, 2013]. Figure 4(a) depicts the layer structure of the Neural Network. The parameters are assumed to be distributed as

$$\alpha_j \sim \mathcal{N}(0, \sigma_\alpha^2), \; \beta_j \sim \mathcal{N}(0, \sigma_\beta^2), \; \gamma_j \sim \mathcal{N}(0, \sigma_\gamma^2), \; \theta_7 = \log \sigma^{-2} \sim \mathcal{N}(0, \sigma_{\theta_7}^2) \tag{5.8}$$

with $\sigma_\alpha = \sigma_\beta = \sigma_\gamma = \sigma_{\theta_7} = 5$. The data was generated from a distribution with $\alpha = 1$, $\alpha_2 = -5$, $\beta_1 = -1$, $\beta_2 = -3$, $\gamma_1 = 5$, $\gamma_2 = 2$, $\sigma = 0.1$ and inputs $x_i = i/10$ for $i = 1, \ldots, 100$. The TA^{2}S^2 algorithm was run with a threshold value for the effective sampling size, $\gamma = 0.3$ to speed up the annealing schedule. This resulted in 17 intermediate distributions. Each annealing level generating 2000 samples from the tempered target distributions in the 7-dimensional space. Figure 4(b) shows how, by marginalising the MAP estimate, a surrogate able to capture the variability of the output in the 95% confidence bands was obtained.
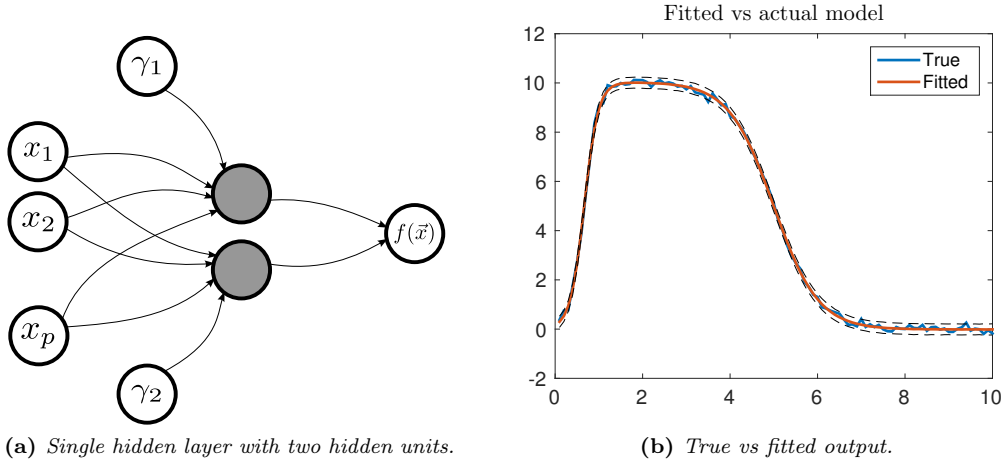
(a) *Single hidden layer with two hidden units.*

(b) *True vs fitted output.*

Figure 4: *Bayesian neural network estimation.*

5.4.2. Variational inference

For complex Bayesian inference problems, sampling from the posterior distribution of the model parameters is usually not a tractable alternative. Variational inference allows to reformulate the problem as an optimisation problem in the space of probability functions [Ormerod and Wand, 2010]. This is usually done be finding a variational posterior $q(\boldsymbol{\theta})$ which minimises the Kullback-Leibler divergence with respect to the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$. In other words, it is necessary to minimise the functional

$$KL(q||p) = \int_{\Theta} \log \left\{ \frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})} \right\} q(\boldsymbol{\theta}) \, d\boldsymbol{\theta}. \tag{5.9}$$

It is widely known that by constraining the solution to a family of parametric distributions the solution to the resulting optimisation problem is likely to have multiple optima [Bishop, 2006, Chapter 10]. For convenience, the family of distributions is selected to be a conjugate distribution or a family from which sampling can be done efficiently.

We present a well-known example in which the true posterior is a mixture of Gaussians and the variational posterior is chosen to be a single Gaussian distribution. The mean vector $\boldsymbol{\mu}$ and the Cholesky decomposition of the covariance matrix, $C^{\top}C = \Sigma$, are chosen to be the design variables. The objective function is

$$\mathcal{H}(\boldsymbol{\varphi}) = \int_{\Theta} \log \left\{ \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\varphi})} \right\} q(\boldsymbol{\theta}|\boldsymbol{\varphi}) \, d\boldsymbol{\theta}, \tag{5.10}$$

where $\boldsymbol{\varphi} = (\boldsymbol{\mu}, C_{11}, C_{12}, C_{22})$ is the vector of design variables. The corresponding Monte Carlo approximation is

$$\min_{\boldsymbol{\varphi}} \hat{\mathcal{H}}(\boldsymbol{\varphi}) = \frac{1}{n} \sum_{i=1}^{n} \log \left\{ \frac{p(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i|\boldsymbol{\varphi})} \right\}, \tag{5.11}$$

where the true density $p(\boldsymbol{\theta})$ is a mixture of two Gaussians with modes separated by a wide valley of zero probability, as depicted in Figure 5(a). Additionally, Figure 5(b) shows one local solution which completely ignores the second mode of the mixture model. By providing approximations generated by TA^{2}S^2, a more robust proposal for the Variational approximation can be achieved. The number of samples generated was $n = 1000$ for the Monte Carlo integration, $N = 5000$ samples in each annealing level. An effective sampling

size threshold of $\gamma = 0.15$ to speed up the annealing schedule resulted in 9 levels.
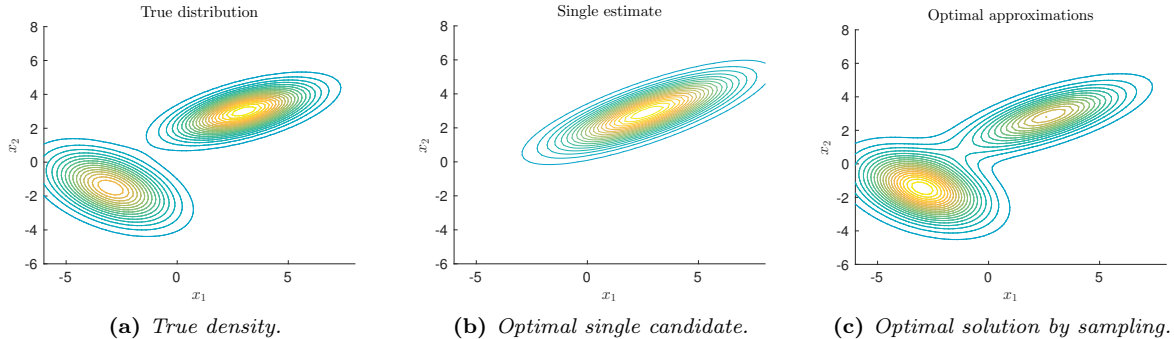

(a) *True density.*　　　**(b)** *Optimal single candidate.*　　　**(c)** *Optimal solution by sampling.*

Figure 5: *Variational approximation to a mixture of Gaussians*

6. Conclusions

The proposed Transitional Annealed Adaptive Slice Sampling (TA^{2}S^2) combines Slice Sampling, Transitional Markov Chain Monte Carlo (TMCMC) and Asymptotically Independent Markov Sampling (AIMS). 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 concerns about warm up periods for the chain, since each sample generated is distributed as desired. Efficient coverage of the sampling space achieved by annealing and an extension of Slice Sampling to Sequential Monte Carlo. The examples presented showed how the method is capable of efficiently exploring multi-modal distributions. This is particularly important when sampling from the posterior distribution of hyper-parameters of a Gaussian process emulator. Additionally, it has also been shown that TA^{2}S^2 can be used in other Bayesian inference contexts (such as variational inference) and for other surrogates (such as a Bayesian neural networks).

The computational cost of the proposed sampler is dominated by the inversion of the covariance matrix in the integrated posterior distribution. Further strategies to improve the speed of the sampler could be developed, such as only evaluating 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.

Acknowledgements

References

I. Andrianakis and P. G. Challenor. The effect of the nugget on Gaussian process emulators of computer models. Computational Statistics and Data Analysis, 56(12):4215–4228, 2012.

Y. Andrianakis and P. G. Challenor. Parameter estimation for Gaussian process emulators. Technical report, Managing Uncertainty in Complex Models, 2011.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. Journal of the Royal Statistical Society. Series B: Statistical Methodology, 72(3):269–342, 2010.

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

J. Beck and K. M. Zuev. Asymptotically Independent Markov Sampling: a new MCMC scheme for Bayesian Inference. International Journal for Uncertainty Quantification, 3(5), 2013.

J. O. Berger and J. M. Bernardo. On the development of reference priors. Bayesian Statistics, 4(4), 1992.

J. O. Berger, B. Liseo, and R. L. Wolpert. Integrated Likelihood Methods for Eliminating Nuisance Parameters. Statistical Science, 14(1):1–28, 1999.

J. O. Berger, J. M. Bernardo, and D. Sun. The formal definition of reference priors. Annals of Statistics, 37 (2):905–938, 2009.

J. R. Birge and N. G. Polson. Optimisation via Slice Sampling. pages 1–22, 2012.

C. Bishop. Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, 2006.

J. Ching and Y.-C. Chen. Transitional Markov Chain Monte Carlo method for Bayesian model updating, model class selection and model averaging. Journal of Engineering Mechanics, 133(7):816–832, 2007.

N. Cressie. Statistics for Spatial Data. Wiley Series in Probability and Statistics. Wiley, 1993.

V. De Oliveira. Objective Bayesian analysis of spatial data with measurement error. Canadian Journal of Statistics, 35(2):283–301, 2007.

P. Del Moral and A. Jasra. Sequential Monte Carlo for Bayesian Computation. Bayesian Statistics, 8:1–34, 2007.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. Journal of the Royal Statistical Society B, 68(3):411–436, 2006.

P. Del Moral, A. Doucet, and A. Jasra. On adaptive resampling strategies for sequential Monte Carlo methods. Bernoulli, 18(1):252–278, 2012.

F. A. DiazDelaO and S. Adhikari. Gaussian process emulators for the stochastic finite element method. International Journal for Numerical Methods in Engineering, 87(6):521–540, 2011.

D. Draper. Assessment and propagation of model uncertainty. Journal of the Royal Statistical Society, Series B, 57(1):45–97, 1995.

P. Fearnhead and B. M. Taylor. An adaptive sequential Monte Carlo sampler. Bayesian Analysis, 8(2): 411–438, 2013.

A. I. J. Forrester, A. Sóbester, and A. J. Keane. Engineering Design via Surrogate Modelling. 2008.

A. Garbuno-Inigo, F. A. DiazDelaO, and K. M. Zuev. Gaussian process hyper-parameter estimation using parallel asymptotically independent Markov sampling. Arxiv pre-print, 2015.

A. Gelman, G. Roberts, and W. Gilks. Efficient metropolis jumping rules, 1996.

M. N. Gibbs. Bayesian Gaussian processes for regression and classification. PhD thesis, Citeseer, 1998.

B. Haaland and P. Z. Qian. Accurate emulators for large-scale computer experiments. Ann. Stat., 39(arXiv: 1203.2433):2974–3002, 2012.

R. Hankin. Introducing BACCO, an R bundle for Bayesian analysis of computer code output. Journal of Statistical Software, 14(16), 2005.

A. A. Kalaitzis and N. D. Lawrence. A simple approach to ranking differentially expressed gene expression time courses through Gaussian process regression. BMC bioinformatics, 12:180, Jan. 2011.

L. Katafygiotis and K. Zuev. Estimation of small failure probabilities in high dimensions by adaptive linked importance sampling. ECCOMAS thematic conference on . . . , pages 1–12, 2007.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(3):425–464, Aug. 2001a.

M. C. Kennedy and A. O'Hagan. Supplementary details on Bayesian Calibration of Computer Models. Technical report, Internal Report. URL http://www. shef. ac. uk/~ st1ao/ps/calsup. ps, 2001b.

D. J. MacKay. Introduction to Monte Carlo methods. In Learning in graphical models, pages 175–204. Springer, 1998.

D. J. C. MacKay. Hyperparameters: Optimize, or integrate out? Maximum entropy and Bayesian methods, pages 43–59, 1996.

I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. 2(1):9, 2010.

I. Murray, R. P. Adams, and D. J. C. Mackay. Elliptical slice sampling. (2), 2009.

R. M. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report, pages 1–144, 1993.

R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, 1997.

R. M. Neal. Regression and classification using Gaussian process priors. Bayesian Statistics, 6, 1998.

R. M. Neal. Annealed importance sampling. Statistics and Computing, 11(2):125–139, 2001.

R. M. Neal. Slice sampling. Annals of Statistics, 31(3), 2003.

R. M. Neal. MCMC using Hamiltonian dynamics. In Handbook of Markov Chain Monte Carlo. 2011.

T. Nilson and A. Kuusk. A reflectance model for the homogeneous plant canopy and its inversion. Remote Sensing of Environment, 27(2):157–167, 1989.

J. Nocedal and S. J. Wright. Numerical Optimization. Springer, 2004.

J. Oakley. Bayesian Uncertainty Analysis for Complex Computer Codes. PhD thesis, 1999.

J. Oakley. Eliciting Gaussian process priors for complex computer codes. Journal of the Royal Statistical Society Series D: The Statistician, 51(1):81–97, 2002.

J. T. Ormerod and M. Wand. Explaining variational approximations. The American Statistician, 64(2): 140–153, 2010.

R. Paulo. Default priors for Gaussian processes. Annals of Statistics, 33(2):556–582, 2005.

P. Ranjan, R. Haynes, and R. Karsten. A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data. Technometrics, 53(4):366–378, 2011.

C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, 2006.

A. A. Taflanidis and J. L. Beck. Stochastic Subset Optimization for optimal reliability problems. Probabilistic Engineering Mechanics, 23(2-3):324–338, Apr. 2008a.

A. A. Taflanidis and J. L. Beck. An efficient framework for optimal robust stochastic system design using stochastic simulation. Computer Methods in Applied Mechanics and Engineering, 198(1):88–101, 2008b.

I. Vernon, M. Goldstein, and R. G. Bower. Galaxy formation: a Bayesian uncertainty analysis. Bayesian Analysis, 5(4):619–669, Dec. 2010.

R. D. Wilkinson. Accelerating ABC methods using Gaussian processes. arXiv preprint, 2014.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. Advances in Neural Information Processing Systems, pages 514–520, 1996.

K. M. Zuev and J. L. Beck. Global optimization using the Asymptotically Independent Markov Sampling method. Computers & Structures, 126:107–119, Sept. 2013.