Fast and robust Bayesian inference using Gaussian processes with GPry

Journal of Cosmology and Astroparticle Physics
An IOP and SISSA journal

Fast and robust Bayesian inference using Gaussian processes with GPry

Jonas El Gammal,[a] **Nils Schöneberg,**[b] **Jesús Torrado**[c,d] **and Christian Fidler**[e]

[a]Department of Mathematics and Physics, University of Stavanger, Kristine Bonnevies vei 22, Stavanger 4021, Norway
[b]Institut de Ciències del Cosmos, Universitat de Barcelona, Martí i Franquès 1, Barcelona E08028, Spain
[c]Dipartimento di Fisica e Astronomia "G. Galilei", Università degli Studi di Padova, Via Marzolo 8, Padova I-35131, Italy
[d]INFN, Sezione di Padova, Via Marzolo 8, Padova I-35131, Italy
[e]Institute for Theoretical Particle Physics and Cosmology (TTK), RWTH Aachen University, Sommerfeldstraße 16, Aachen 52074, Germany
E-mail: jonas.e.elgammal@uis.no, nils.science@gmail.com, jesus.torrado@pd.infn.it, fidler@physik.rwth-aachen.de

Received December 20, 2022
Revised June 3, 2023
Accepted July 6, 2023
Published October 6, 2023

Abstract. We present the GPry algorithm for fast Bayesian inference of general (non-Gaussian) posteriors with a moderate number of parameters. GPry does not need any pre-training, special hardware such as GPUs, and is intended as a drop-in replacement for traditional Monte Carlo methods for Bayesian inference. Our algorithm is based on generating a Gaussian Process surrogate model of the log-posterior, aided by a Support Vector Machine classifier that excludes extreme or non-finite values. An active learning scheme allows us to reduce the number of required posterior evaluations by two orders of magnitude compared to traditional Monte Carlo inference. Our algorithm allows for parallel evaluations of the posterior at optimal locations, further reducing wall-clock times. We significantly improve performance using properties of the posterior in our active learning scheme and for the definition of the GP prior. In particular we account for the expected dynamical range of the posterior in different dimensionalities. We test our model against a number of synthetic and cosmological examples. GPry outperforms traditional Monte Carlo methods when the evaluation time of the likelihood (or the calculation of theoretical observables) is of the order of seconds; for evaluation times of over a minute it can perform inference in days that would take months using traditional methods. GPry is distributed as an open source Python package (`pip install gpry`) and can also be found at https://github.com/jonaselgammal/GPry.

Keywords: Machine learning, Statistical sampling techniques, Bayesian reasoning
ArXiv ePrint: 2211.02045

https://doi.org/10.1088/1475-7516/2023/10/021

JCAP10(2023)021

Contents

1 Introduction

One of the fundamental tools of science is the comparison of observations with theory. In many situations, this involves inference on the parameters of a model (or on models themselves) given some observed data. This is often realised using Bayesian statistics, where

JCAP10(2023)021

one synthesises the probability of some data having been acquired into a *likelihood function*, assumes some *a priori* distribution for the model parameters, and samples from the product of both (proportional to the so-called *posterior*) using Monte Carlo methods, the most common ones in Cosmology being based on Markov Chain Monte Carlo [1–4] or Nested Sampling [5–12].

The new era of cosmological surveys will produce data in rapidly increasing amount and quality [13, 14]. This will in turn raise the computational costs of traditional Monte Carlo pipelines: data quality will call for an increase in the precision of theoretical computations of the observables that are compared against the data (e.g. including physical effects that could have been so far neglected), and likelihood computations will involve operations on ever larger data vectors. This can and will eventually result in traditional Bayesian inference becoming prohibitively slow, further increasing the already damaging carbon footprint of scientific computations in computer clusters [15, 16]. In order to keep being able to exploit cosmological data for parameter inference, we need to develop more advanced algorithms that significantly reduce the computational costs of performing inference, and machine-learning based methods are one of the most promising tools for that.

So far, a number of different solutions have been proposed. A family of them focus on substituting the theoretical computation of observables (or intermediate quantities to arrive at them) by appropriately-trained, usually Neural Network-based, *emulators* that cheaply map the theoretical parameters onto the space of vectors of observables. For applications to Cosmology and Astrophysics, see e.g. [17–40]. These methods are robust in the sense that they are guaranteed to reproduce the true posterior distribution, as long as the emulator is properly trained, which is easy to check a posteriori. Unfortunately their utility is limited by the need to retrain them whenever the theoretical model under investigation is varied. Additionally, as experiments become ever more precise, in order to achieve sufficient accuracy a larger number of systematic effects needs to be accounted for, which requires ever more costly experimental likelihoods, which cannot be easily accelerated by emulators.

Another proposed solution are simulation-based *likelihood-free* approaches, inspired by Approximate Bayesian Computation, but accelerated by Neural Networks [41, 42]. There, Neural Networks are used to learn a mapping between sets of model parameters and their corresponding simulated data, so that they can automatically extract features, marginalise over nuisance parameters, learn a likelihood function, or ultimately produce a posterior distribution of the model parameters when fed real experimental data. Recent development and applications in Cosmology and Astrophysics can be found in [43–54]. The claimed advantages are that they may discover or take into account features in the data that are not captured by summary statistics or observables, and the lack of need to formulate a likelihood, which can be complex or prohibitively expensive in some cases. On the other hand, they tend to require expensive training and the reusability of the trained networks is limited when considering model extensions. The need to accurately account for modelling uncertainty and possible biases has also been highlighted recently [55, 56].

The solution presented in this work differs from the previous ones in that it retains the full computation of the observable and data likelihood, but minimises the number of points in the parameter space where this full pipeline needs to be computed; it uses these points to create a model of the posterior, and to iteratively predict the next optimal evaluation locations. For the emulation of the posterior we use Gaussian Processes (GP) [57], which have a small number of hyperparameters that are easily interpretable in terms of properties of the posterior, and thus make it easier to incorporate prior information about its functional

JCAP10(2023)021

form. Furthermore, due to their simplicity, Gaussian Processes generally require smaller training sets than for example Neural Networks. We combine the GP model of the posterior with a support vector machine (SVM) [58, 59] to restrict the parameter space to a region of reasonable posterior values.

Our approach expands on previous work applying Bayesian quadrature and active sampling to statistical inference [60–64], which we improve upon by incorporating the expected scaling of the log-posterior with dimensionality, the definition of a cheap and consistent convergence criterion and the treatment of extreme log-posterior values with an SVM classifier. A previous attempt at a similar approach to inference in Cosmology with a GP surrogate of the posterior can be found in [65], and in the context of emulator-training in [66]. Alternative emulator-based approaches, relying on Variational Inference, have also been proposed, e.g. combined with a GP surrogate model to reduce the number of posterior evaluations [67–70], or targeted towards high dimensionalities but allowing for numbers of evaluations similar to MCMC [71, 72].

The result of our work is the development of the `GPry` algorithm. An open source implementation is available as a Python package (`pip install gpry`) and at https://github.com/jonaselgammal/GPry. `GPry` does not need any pre-training or parameter tuning, so it can be used as a *drop-in* replacement for traditional Monte Carlo algorithms for dimensionalities $N_d \lesssim 20$ (since the computational cost of the algorithm makes it impractical for larger problems in its current implementation). Unlike neural networks it also does not require any specialised hardware such as GPUs. As we will show, it allows for accurate and fast emulation of posteriors for moderate dimensionalities, including non-Gaussian distributions, by using just a few hundred or thousand evaluations of the posterior distribution. Especially when individual likelihood evaluations are computationally expensive, this can result in large speedups of typically two orders of magnitude.

This paper is structured as follows: in section 2 we review the basic concepts and useful notation. We continue in section 3 presenting the modelling choices involved in the construction of the GP surrogate model. The learning strategy for acquiring new sampling locations as well as a criterion for deciding on convergence are discussed in section 4. In section 5 we put together all the pieces and present the full algorithm, and comment on its general performance. We discuss the performance of `Gpry` on different synthetic and cosmological problems in section 6, and we present our conclusions and discuss possible future development in section 7. Appendix A is dedicated to discussing the inclusion of prior information on the dynamical range of the posterior into the surrogate model at different stages of the algorithm.

2 Basic concepts

In order to establish a consistent notation and a deeper understanding of the underlying concepts, we quickly summarize some of the theory, which we are going to use in the detailed description of section 3.

2.1 Bayesian inference of model parameters

A usual Bayesian inference problem is that of estimating the probability distribution $p(\mathcal{M}(\boldsymbol{x})|\mathcal{D})$ of the parameters $\boldsymbol{x}$ of a model $\mathcal{M}$ given some experimental data $\mathcal{D}$, also known as *posterior*. Following Bayes' theorem, this is proportional to the product of the *likelihood*

JCAP10(2023)021

$p(\mathcal{D}|\mathcal{M}(\boldsymbol{x}))$ (the probability of $\mathcal{D}$ having being measured given the model with these parameter values), and the *prior* probability of the parameter values $\boldsymbol{x}$ given the model, $p(\boldsymbol{x}|\mathcal{M})$, assigned before (or idependently of) the experiment that measured $\mathcal{D}$.[1] Fixing the model $\mathcal{M}$ and the data $\mathcal{D}$, we can drop their explicit dependence to simplify notation. With that Bayes' theorem reads

$$p(\boldsymbol{x}) \propto \mathcal{L}(\boldsymbol{x})\pi(\boldsymbol{x})\,, \tag{2.1}$$

where $p(\boldsymbol{x})$ is the posterior, $\mathcal{L}(\boldsymbol{x})$ the likelihood, and $\pi(\boldsymbol{x})$ the prior.

In Cosmology, likelihoods are typically provided by experimental collaborations, are generally non-analytic, or analytic but non-differentiable, and usually also costly to evaluate. Even when they are well-behaved, they sometimes depend on cosmological quantities whose computation in terms of the parameters to be inferred has the same undesirable properties. In these cases, the targeted solution to the inference problem is obtaining a Monte Carlo sample of the posterior, often using MCMC- or nested-sampling-based methods.

This work focuses on reducing the number of evaluations of the posterior (and thus the likelihood) needed to solve the inference problem. We do that by creating a surrogate model of the posterior using a Gaussian Process, and developing an active learning algorithm that decides sequentially on a small optimal set of parameter values where to evaluate the true likelihood, so that the surrogate model is accurate enough. One can then e.g. extract the usual Monte Carlo sample from the resulting surrogate model of the posterior (which, as a bonus, is differentiable) at a very low computational cost.

If the goal is to obtain 1D/2D posteriors (and their corresponding CLs) from the GP, one could wonder if there would be alternative efficient methods of computing the required marginalization integrals. However, generally the integrals involved are not solvable analytically and due to the high dimensionality of these integrals in most applications, the most efficient ways of computing them numerically are usually Monte-Carlo methods. We discuss the computational costs of this choice in section 5.3.

2.2 Gaussian processes

We briefly present the relevant GP notation and formulae that we will need for this work. For a more thorough review, see [57].

Gaussian Processes are useful to emulate a sufficiently smooth[2] function $f(\boldsymbol{x})$ at an arbitrary point $\boldsymbol{x}$ (within a certain domain) given a set of sampling locations $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}\}$ and their corresponding function values $y_i = f(\boldsymbol{x}_i)$ for $i = 1 \ldots N_s$. This last equation can be abbreviated to $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{X})$ (notice the bold symbol for $\boldsymbol{y}$ and $\boldsymbol{f}$ and the dependence on $\boldsymbol{X}$) following the usual notation in GP literature, where the number of samples is treated as an additional vector space of dimension N_s, with components denoted by a subscript. This means that $\boldsymbol{X}$ becomes a $N_s \times N_d$ matrix, where N_d is the dimensionality of the parameter space. This way, we write for a scalar function $s(\boldsymbol{x})$ evaluated at the N_s different sampling locations the vector $\boldsymbol{s}(\boldsymbol{X})$ with components $[\boldsymbol{s}(\boldsymbol{X})]_i = s(\boldsymbol{X}_i)$, and similarly for scalar functions of two arguments the tensor $\boldsymbol{s}(\boldsymbol{X}, \boldsymbol{X})$ with components $[\boldsymbol{s}(\boldsymbol{X}, \boldsymbol{X})]_{ij} = s(\boldsymbol{X}_i, \boldsymbol{X}_j)$.

A Gaussian Process posits that the function $f(\boldsymbol{x})$ in question is a random draw from a family of functions, informed by the sampling locations. For a given position $\boldsymbol{x}$ such random

[1] The missing proportionality constant is the inverse of the *evidence* $p(\mathcal{D}|\mathcal{M})$, which can usually be ignored in parameter estimation and will hence be omitted in all subsequent calculations. Note though, that the evidence is important when performing model selection.

[2] Here, "sufficiently smooth" refers to an underlying function which n-times continuously differentiable where $n \geq 1$. The function may still have some statistical or numerical noise added on top of it.

JCAP10(2023)021

draw of a function $f(\boldsymbol{x})$ is assumed to be Gaussian-distributed (hence the name) around a mean function $m(\boldsymbol{x})$ with a covariance between the functional value at two different points given by some function $k(\boldsymbol{x}, \boldsymbol{x}')$, often called the *kernel* of the GP.

$$\widehat{f} \sim \mathcal{GP}(m, k) \qquad \Leftrightarrow \qquad \widehat{f}(\boldsymbol{x}) \sim \mathcal{N}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))\,, \tag{2.2}$$

where $\widehat{f}$ denotes a random function draw from the GP and $\sim$ means "is distributed according to". As a multivariate Gaussian distribution, the GP is completely defined by its mean and kernel functions. Their precise choice only aids in faster and more predictive emulation, but they do not in general restrict the shape of the functions being modeled, which can be complete arbitrary as long as the kernel function fulfills a number of weak conditions [73] (that all kernels considered in this work do). Importantly, while the *correlation* of the function value at two points is assumed to be Gaussian, this neither means that the function is itself assumed to be Gaussian, nor that the mean of the family of functions is presumed to be Gaussian.

We usually restrict the GP so that it agrees with the given set of sampling locations for all draws, $\widehat{\boldsymbol{f}}(\boldsymbol{X}) \stackrel{!}{=} \boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{y}$, sometimes up to some uncorrelated Gaussian noise. This information modifies the value of the drawn function's *predictions* $\widehat{\boldsymbol{f}}(\boldsymbol{X}_*)$ away from the sampled values $\boldsymbol{X}$. The joint distribution for sampled and non-sampled locations is

$$\begin{bmatrix} \widehat{\boldsymbol{f}}(\boldsymbol{X}) \\ \widehat{\boldsymbol{f}}(\boldsymbol{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\boldsymbol{X}) \\ \boldsymbol{m}(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}) & \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{k}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{k}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix} \right). \tag{2.3}$$

This defines the *conditional* probability for the predictions $\widehat{\boldsymbol{f}}(\boldsymbol{X}_*)$ given the observations $(\boldsymbol{X}, \boldsymbol{y})$ as

$$\widehat{\boldsymbol{f}}|_{\boldsymbol{f}(\boldsymbol{X})=\boldsymbol{y}} \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad \Leftrightarrow \qquad \widehat{\boldsymbol{f}}(\boldsymbol{X}_*)|_{\boldsymbol{f}(\boldsymbol{X})=\boldsymbol{y}} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{X}_*), \boldsymbol{\Sigma}(\boldsymbol{X}_*))\,. \tag{2.4}$$

with mean vector and covariance matrix

$$\boldsymbol{\mu}(\boldsymbol{X}_*) = \boldsymbol{m}(\boldsymbol{X}_*) + \boldsymbol{k}(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X})^{-1}\left[\boldsymbol{y} - \boldsymbol{m}(\boldsymbol{X})\right]\,, \tag{2.5}$$

$$\boldsymbol{\Sigma}(\boldsymbol{X}_*) = \boldsymbol{k}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{k}(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}_*)\,. \tag{2.6}$$

This conditioned GP for new sample predictions is then called the *posterior GP*. Comparing equations (2.2) and (2.4) we notice that the drawn samples $\widehat{\boldsymbol{f}}$ differ between the unconditioned and the conditioned GP, because the latter includes the additional information from the sampling locations. The algorithm described in section 5 will sequentially add new samples to the GP. These will be incorporated by *updating* the mean and covariance of this conditioned GP (μ, Σ) using sequentially enlarged sample sets $(\boldsymbol{X}, \boldsymbol{y})$.

From here on, we will use the scalar versions of equations (2.5) and (2.6) evaluated at an arbitrary single location $\boldsymbol{x}$ as $\mu(\boldsymbol{x})$ and $\Sigma(\boldsymbol{x})$, as well as $\sigma(\boldsymbol{x}) = \sqrt{\Sigma(\boldsymbol{x})}$ as the uncertainty of the GP at a location $\boldsymbol{x}$ to simplify notation, implicitly assuming it has been conditioned on the samples $\boldsymbol{X}$. As is standard in the literature (and as discussed without loss of modeling power for the GP), we will assume a zero-mean function $m(\boldsymbol{x}) = 0$ in all cases.

Kernel functions are usually chosen from a particular family of functions (such as squared exponentials, *Matérn* kernels, ...),[3] parameterized by some *hyperparameters* θ. Their value

[3]The kernel function is typically chosen according to the differentiability and smoothness of the given target function, see also section 3.1 for more details.

JCAP10(2023)021

is commonly chosen so that they maximize the likelihood that the GP would have produced the given sampled values $\boldsymbol{y}$ at the sampled locations $\boldsymbol{X}$. In practice, one marginalizes the evidence of the training data given the Gaussian Process [57]:

$$-\log p(\boldsymbol{y}|\boldsymbol{X},\theta) = \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{k}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{y} + \frac{1}{2}\log|\boldsymbol{k}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I}| - \frac{N_s}{2}\log 2\pi\,, \quad (2.7)$$

where $\boldsymbol{I}$ is the identity matrix, and σ_n is an arbitrary small level of uncorrelated *noise* included to make the algorithm more numerically stable (possibly in addition to a noise term added into the kernel function to model stochasticity of the original function). Using Bayes' theorem, the product of this likelihood and some prior can then be sampled or (more commonly) simply maximized with respect to the hyperparameters θ.

3 Surrogate model of the posterior

Our goal is to interpolate an unknown, possibly multi-dimensional log-posterior distribution with a GP, using the mean prediction $\mu(\boldsymbol{x})$ of the GP as a best estimate for the distribution's value. Furthermore we want to achieve an accurate estimate for the standard deviation $\sigma(\boldsymbol{x})$ in order to compute where to sample next. The nature of the posterior distribution being an (un-normalized) probability distribution implies certain properties/restrictions, that can be incorporated into the GP surrogate model in order to increase the performance of the algorithm and reduce the risk of numerical issues. These will be discussed in the following.

3.1 Choice of kernel function

As discussed in section 2.2, a kernel function with a minimal set of properties will ensure that the GP converges towards the target function (the log-posterior) given a large enough set of samples. However, in order the keep the computational costs low, we aim to use as few samples as possible, and this can be achieved by choosing a kernel function that encapsulates our prior information on the posterior distribution.

The prior information that we aim to encode is that the log-posterior distribution is deterministic,[4] and smooth over a characteristic correlation length-scale, that possibly differs between dimensions and is a fraction of the prior size (as we cannot resolve length-scales much larger than the prior). Our default choice in GPry is an anisotropic quadratic RBF kernel multiplied by a constant:

$$k(\boldsymbol{x},\boldsymbol{x}') = c^2 \cdot \exp\left(-\sum_{i=1}^{d}\frac{|x_i - x_i'|^2}{2L_i^2}\right)\,, \quad (3.1)$$

where c is usually called the output-scale, and $L_{i=1,\dots,N_d}$ are the length-scales.[5] On top of the choice of the kernel function itself, prior knowledge on the target function is also incorporated

[4] It would be easy to extend this to stochastic functions by adding a noise component to equation (3.1), but posterior density functions of physical data are most commonly deterministic.

[5] If the covariance matrix of the posterior mode that is modelled is approximately known, and that mode is Gaussian enough, one could transform the parameter space using that covariance matrix so as to normalise the Gaussian, in which case the target function is isotropic and we can use a single common length scale, significantly reducing the computational cost of fitting the hyperparameters. In practice, this approach has its own difficulties: even at late stages of learning, the set of training points is too small to compute the covariance matrix via simple Monte Carlo (weighting by their posterior value), and one needs to resort to other approaches, such as fitting a Gaussian to the training, or MC-sampling from the GP (see e.g. [65]), the cost of which would likely compensate for the time saved by fitting a single isotropic correlation length-scale.

JCAP10(2023)021

in the priors for the hyperparameters. The fundamental assumption is that the length scales should be of an order of magnitude close to that of the posterior modes, and that the latter would be of an order of magnitude not much smaller than that of the prior ranges for the parameters of the posterior. We express this belief as the length-scales being between 0.01 and 1 in units of the prior length in each direction. The lower bound ensures that the GP does not overfit during early stages of the learning by fitting each sample individually as a peak on top of the mean of the GP,[6] while the upper bound represents the fact that the size of the prior box should prevent drawing any conclusions on the characteristic length-scale far beyond the region that can be sampled. The prior of the output scale c is chosen to be very broad and allows for values between 0.001 and 10000. The $N_d + 1$ free hyperparameters $\{c, L_i\}$ are then chosen such that they maximize equation (2.7).[7]

3.2 Parameter space transformations

As a un-normalized probability density, the posterior is a positive function ($p(\boldsymbol{x}) \geq 0$ everywhere), and even for a simple one-dimensional Gaussian, it varies over multiple orders of magnitude. Both enforcing positivity and reducing the dynamic range of function values can be achieved by modeling the result of a power-reduction operation $P\left(p(\boldsymbol{x})\right)$ on the posterior (e.g. a logarithm [61] or a square root [60]). We use a log-transformation, since in physics it is very common for likelihoods to belong to the *exponential family* of probability distributions [74] and in practice many likelihood codes usually return log-probabilities.

Another advantage of modelling in log-space, that was pointed out in [61], is that the characteristic length scale of isotropic kernels (e.g. Radial Basis Function (RBF) or Matérn) tends to be larger, which implies that the GP surrogate better generalizes to distant parts of the function, making the GP more predictive.

In practice, we construct a surrogate model for $\log p(\boldsymbol{x})$ given some training samples $\boldsymbol{y} = \log p(\boldsymbol{X})$. In addition, at every iteration of the algorithm, we *internally* re-scale the modeled function using the mean and standard deviation of the current samples set as

$$\log \widetilde{p}(\boldsymbol{X}) = \frac{\log p(\boldsymbol{X}) - \overline{\boldsymbol{y}}}{s_{\boldsymbol{y}}}\,, \tag{3.2}$$

where $\overline{\boldsymbol{y}}$ and $s_{\boldsymbol{y}}$ are the sample mean and standard deviation respectively. This re-scaling acts like a non-zero mean function, causing the GP to return to the mean value far away from sampling locations. This in turn encourages exploration when most samples are close to the mode and exploitation when most samples have low posterior values. This effect can be seen in figure 3 where the mean of the GP is pushed to higher values close to the edge of the prior. The variance reduction through division by $s_{\boldsymbol{y}}$ aids in ensuring numerical stability by restricting the range of values in the training set.

[6] This condition assumes that the size of the mode is larger than about 1/100th of the prior width in each dimension, which we find reasonably permissive. If this is not the case, either the prior dimensions or the allowed range for the length scales can be re-adjusted.

[7] In a full hierarchical Bayesian treatment, instead of maximising we would have to generate a family of GPs with hyperparameters following the likelihood of equation (2.7), each of them giving different predictions according to equations (2.5) and (2.6). Unfortunately, generating a MCMC sample in order to marginalize over equation (2.7) as function of θ is intractable. There have been some attempts at approximate methods [61] however even those introduce some computational overhead which we want to avoid. Luckily, as the number of training points of the GP increases we expect equation (2.7) to get narrower (for sufficiently tame distributions) so that the difference becomes negligible.

JCAP10(2023)021

As for the space of parameters $\boldsymbol{x}$, we transform the samples such that the prior boundary becomes a unit-length hypercube. For unbounded priors, such as Gaussian or half-Gaussian, we choose the prior boundary such that it contains a large fraction of the prior probability mass (99.95% by default, which is usually sufficient for the usual few-σ CL contours).

This parameter transformation aims at forcing posterior modes to have similar sizes in all dimensions. This usually leads to comparable correlation length scales of the GP across dimensions, which increases the effectiveness of the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B) constrained optimizer [75], used to optimize both the GP hyperparameters and the acquisition function.

Henceforth, if not specified otherwise, in the context of the training set we will refer to $\boldsymbol{X}$ (or $\boldsymbol{x}$) as the un-transformed values of the sampling locations while $\boldsymbol{y}$ refers to the un-transformed values of the log-posterior distribution at $\boldsymbol{X}$.

3.3 Treatment of infinities and extreme values

In realistic inference scenarios the prior is often chosen to be much larger than the posterior mode, since very little initial information is usually known. In these scenarios the log-posterior function is bound to return minus infinity for parameter values far away from the region of interest (the posterior modes): the negative log-posterior can be too large to be represented as a floating point number, or the physics code used to compute the likelihood could fail and report a zero-valued likelihood.

This is valuable information but unfortunately we cannot simply add those infinite values to the GP as equations (2.5) and (2.7) become ill-defined. Hence we are forced to find some numerically stable way of incorporating this information. Naively one could simply swap out the infinities with some large negative value. This approach turns out to be rather problematic as it introduces a discontinuity in the posterior shape or at least one of it derivatives thus modifying the hyperparameters of the GP's kernel. If we instead ignore these points, the learning algorithm will repeatedly try to acquire points in their vicinity, hence getting stuck.

Our solution to this problem is to simultaneously exclude these *infinities* from the GP, and to use them to divide the parameter space into a *finite* and an *infinite* region using a support vector machine (SVM) classifier [58, 59].[8] A SVM defines a hyperplane which maximizes the separation between samples with locations $\boldsymbol{x}_i$ belonging to one of two classes $y \in \{-1, 1\}$. By defining the distance between points through a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ the separating hyperplane is drawn in a higher-dimensional space which is connected to the sample space by a non-linear transformation. This effectively transforms the separating hyperplane into more complex hypersurfaces which are better suited to the classification problem at hand.

The categorical predictions $\hat{y}(\boldsymbol{x})$ of the SVM are then given by

$$\hat{y}(\boldsymbol{x}) = \mathrm{sgn}\left(b + \sum_{i=1}^{N_s} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})\right) \tag{3.3}$$

where the hyperparameters b and α_i are optimized in the training procedure.

We simply use the prediction of the SVM of whether a point is classified as being finite ($\hat{y} = +1$) or infinite ($\hat{y} = -1$) to "correct" the prediction of the GP. Compared to

[8] A similar "safe exploration space" approach, using different tools, has also been used e.g. [76, 77] in the context of Bayesian optimization.

JCAP10(2023)021

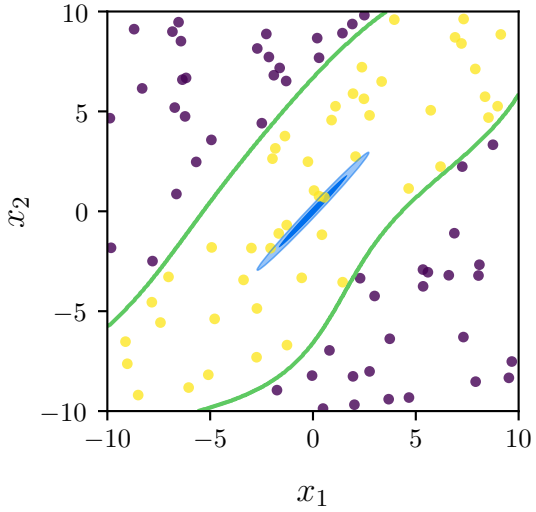


Figure 1. Illustration of the SVM classification. Yellow dots correspond to uniformly sampled locations where the log-posterior distribution is finite while purple dots correspond to infinite log-posterior samples. The green lines are the boundary found by the SVM separating the finite and infinite regions. The blue contours show the 1- and 2-σ contours of the posterior distribution (in this case a correlated 2-d Gaussian). In our construction this finite region is designed to roughly correspond to the 10σ volume of the Gaussian distribution.

equations (2.5) and (2.6) we can explicitly write

$$\mu_{\mathrm{GP+SVM}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) \cdot \begin{cases} 1 & \text{if } \hat{y}(\boldsymbol{x}) = +1 \\ -\infty & \text{if } \hat{y}(\boldsymbol{x}) = -1\,. \end{cases} \tag{3.4}$$

For now, we assert such classification from the SVM with absolute certainty, and set

$$\Sigma_{\mathrm{GP+SVM}} = \Sigma(\boldsymbol{x}) \cdot \begin{cases} 1 & \text{if } \hat{y}(\boldsymbol{x}) = +1 \\ 0 & \text{if } \hat{y}(\boldsymbol{x}) = -1\,. \end{cases} \tag{3.5}$$

The precise way of cutting the covariance is irrelevant in our case.[9] Figure 1 shows a two-dimensional toy example of such a classification for a Gaussian distribution in a comparatively much larger prior region.

Aside from making the acquisition procedure more efficient by ignoring unimportant regions, this approach also keeps the overhead cost of the algorithm lower than including a regularized version of the inifinities in the GP. This is because the computational expense of training a SVM scales as N_s^2, which is smaller than the N_s^3 scaling for the GP.

It is important to recognize that the same arguments can be made for very low posterior values which are far away from the top of the mode, even for well-behaved posterior distributions in high dimensionality. While the SVM is not strictly needed here, adding these values to the surrogate GP model is undesirable as they can dramatically change the scale of the emulation problem even though they do not provide a large amount of additional

[9]Still, one could imagine using the SVM output before sgn function (the classification step) to more smoothly suppress both mean and covariance, possibly combined with a sigmoid function.

JCAP10(2023)021

information. In this sense, the algorithm also benefits from a regularization of forwarding too small log-posterior values to the SVM.

We accomplish that by treating all values where $\log p(\boldsymbol{x})$ is smaller than some (sufficiently low) threshold as infinities. However, one has to be careful about the un-normalized nature of the posterior when applying the threshold. In practice, we compare against the maximum of the posterior in the training sample (corresponding to point $\boldsymbol{x}_{\max}$) and only treat values as infinite when $\log p(\boldsymbol{x}) < \log p(\boldsymbol{x}_{\max}) - T$. We provide a default value for T based on the prescription of appendix A, also giving the user the option to set it manually.

Lastly, we stress that the additional modelling presented in this section is only used in practice if the log-posterior distribution ever returns either negative infinities or values below the proposed threshold. Otherwise, only the bare GP model described in the previous sections is used.

4 Learning strategy

In section 3 we have described the process of constructing a Gaussian process to emulate the log-posterior distribution once a given set of samples are known. As discussed, a sufficiently large naive set of samples (e.g. prior samples) will in general lead to an accurate model. Unfortunately the computational cost of the algorithm scales with the number of samples N_s, both directly as the number of times a possibly-costly true posterior needs to be evaluated, and indirectly by increasing the computational cost of the Gaussian Process itself (as $\sim N_s^2$ at evaluation, and $\sim N_s^3$ when fitting). In practice, samples are chosen so that their location maximises an *acquisition function*, representing some measure of how valuable they would be for the emulation when added to the GP. We discuss this approach in section 4.1. A further reduction in computational costs can be achieved by taking advantage of the number of machines/CPUs in computing clusters (and of CPU cores in user-level CPUs). Thus, we discuss the parallelization of the algorithm in section 4.2. Finally, in section 4.3 we discuss the vital question of when to end the acquisition of further samples automatically. Together, this allows `GPry` to tackle the emulation of arbitrary distributions in a highly parallelized way without relying on the end-user to optimize the number or locations of the samples.

4.1 Acquisition function

As discussed above, in order to find a small, but informative set of sampling locations, we will look for locations that maximize an *acquisition function* $a(\boldsymbol{x})$. This function will be constructed using a combination of the mean and variance of the GP estimate, in such a way that it balances exploration of the full parameter space (typically where the uncertainty in the prediction is high) with exploitation of areas of high posterior values (which should be more precisely modeled).

4.1.1 Choice of the acquisition function

A simple ansatz for an acquisition function that balances exploration and exploitation could be the product of the estimated posterior $p(\boldsymbol{x})$ (which is always positive) and its uncertainty $\sigma_p(\boldsymbol{x})$:[10]

$$a_p(\boldsymbol{x}) = p(\boldsymbol{x}) \cdot \sigma_p(\boldsymbol{x})\,. \tag{4.1}$$

[10]This is by no means the only ansatz one could make to arrive at a suitable acquisition function. For a thorough investigation see [78].

JCAP10(2023)021

Given that the GP models $\log p$, we have to convert the GP's mean $\mu(\boldsymbol{x})$ and uncertainty $\sigma(\boldsymbol{x})$ into those of the linear $p(\boldsymbol{x})$. Since the transformation from $\log p$ to p is non-linear, the corresponding prediction for p from the GP is not a Gaussian distribution and the computation of its mean and standard deviation is non-trivial. However, in practice these details are irrelevant since the acquisition function only needs to approximate the most beneficial sampling location. Then, we can simply write for the mean $p(\boldsymbol{x}) \approx \exp[\mu(\boldsymbol{x})]$ and for the uncertainty $\sigma_p \approx \exp[\mu(\boldsymbol{x}) + \sigma(\boldsymbol{x})] - \exp[\mu(\boldsymbol{x})]$. With this, the acquisition function above becomes:

$$a_p(\boldsymbol{x}) \approx \exp[2\mu(\boldsymbol{x})] \left\{\exp[\sigma(\boldsymbol{x})] - 1\right\} , \tag{4.2}$$

which is similar to the acquisition functions used in [79, 80]. This approximation can be further linearized assuming $\sigma(\boldsymbol{x}) \ll 1$ to give $a_p^{\rm lin}(\boldsymbol{x}) = \exp[2\mu(\boldsymbol{x})]\sigma(\boldsymbol{x})$.

As discussed in the next section, we found it beneficial to boost the exploratory behaviour of the acquisition function, especially in high dimensions. To achieve that, we include a relaxation factor $\zeta \in (0, 1]$ multiplying the mean to discourage exploitation (similar to what was done in e.g. [65]). This yields the final acquisition function:

$$a_p(\boldsymbol{x}) \approx \exp[2\zeta\mu(\boldsymbol{x})] \left\{\exp[\sigma(\boldsymbol{x}) - \sigma_n] - 1\right\} \tag{4.3}$$

which can again be linearized as $a_p^{\rm lin}(\boldsymbol{x}) = \exp[2\zeta\mu(\boldsymbol{x})][\sigma(\boldsymbol{x}) - \sigma_n]$. Notice also that from the $\sigma(\boldsymbol{x})$ term we have subtracted a possible uncorrelated noise term proportional to σ_n^2 in the kernel function (equivalently, a constant term added to the diagonal of the kernel covariance matrix). This is because for acquisition purposes we only care about the uncertainty coming from decorrelation from the sampled locations.

The logarithm of this acquisition function is maximized at every acquisition step which yields a candidate for the next sampling location. The computational overhead of the acquisition procedure is dominated by the prediction of $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ by the GP which scales as $\sim N_s^2$.

4.1.2 Acquisition hyperparameter

The effect of ζ in equation (4.3) is that of balancing exploitation and exploration. Values of ζ that are too high make the algorithm focus too much on the top of a posterior mode, so that samples in the tails are unlikely to be proposed, and during large numbers of iterations the GP model is mostly stable (only adding high-posterior but low-information samples). This leads to unnecessarily high computational costs, and often to false positives in assessing convergence. These effects are more dramatic in higher dimensions. On the other hand, values of ζ that are too low would produce more regular but slower convergence, neglecting information about the expected value of the true function that could have been exploited to converge faster. In general, a sub-optimal choice of ζ will increase the amount of samples necessary for convergence, sometimes quite significantly.

To select appropriate values, we have conducted a series of experiments on degenerate Gaussian posterior distributions in 2, 4, 8 and 16 dimensions (for $N_d < 4$ the effect of ζ is small), generated as explained in section 6.1. In order to isolate the effects of ζ, in these tests we have not used the parallelization scheme described in section 4.2. The results of these experiments in terms of KL divergence (see appendix B) are shown in figure 2, and have led us to propose the empirical formula $\zeta = N_d^{-0.85}$ as a default value for ζ (users can override it if prior knowledge of the posterior shape suggests that exploration should be favored over exploration or vice versa). Preliminary tests in higher dimensions (up to $N_d = 27$) have

JCAP10(2023)021

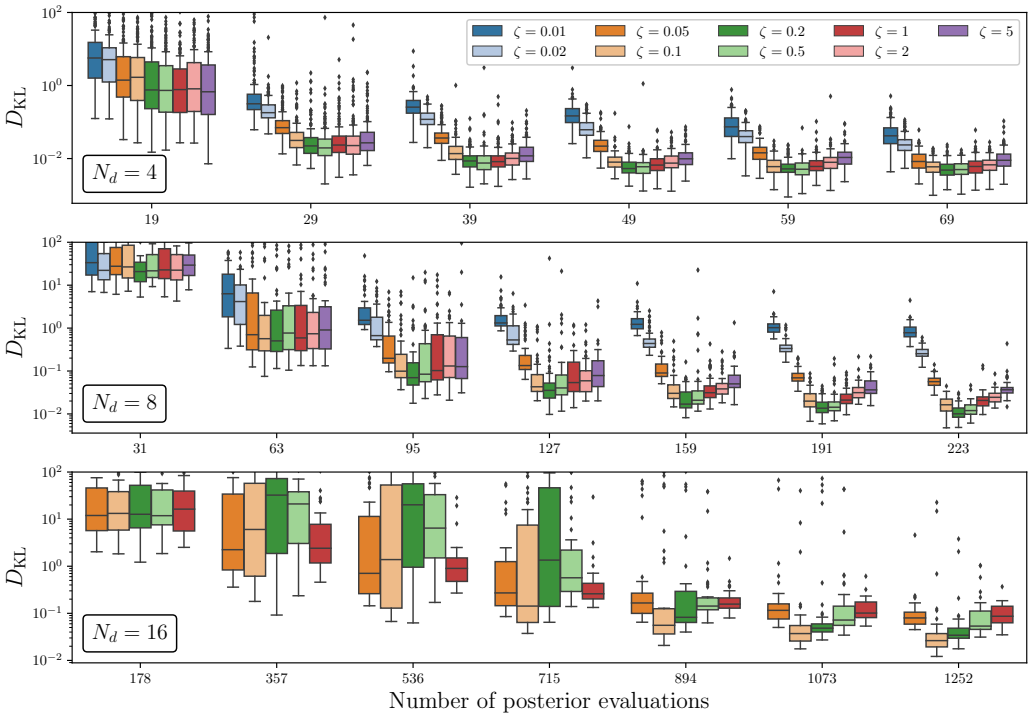


Figure 2. Distribution of Kullback-Leibler divergences between the GP prediction and the true distribution at various learning stages (i.e., N_s samples) for random correlated Gaussian posteriors with dimensionality $N_d = 4$, 8, and 16 (150, 50, and 50 realizations, respectively). The boxes represent inter-quartilic ranges, the black line inside them the median, and the whiskers and dots represent the tails of the distributions. For each dimensionality there is a visible trend towards an optimal trade-off between exploration and exploitation in terms of ζ.

shown this formula to produce good results. The fact that a fixed ζ becomes greedier as dimensionality goes up should not come as a surprise, as discussed in appendix A.

4.1.3 Optimization of the acquisition function

For the maximization of the acquisition function we use the L-BFGS-B optimizer [75] included in the `scipy` Python package. Since this optimization problem is highly non-convex, with the acquisition function often having many disconnected maxima, the numerical optimization is performed multiple times from different randomly-drawn starting locations. In high dimensions drawing an initial point with a non-vanishing value of the acquisition function becomes increasingly unlikely as the prior volume with vanishing posterior increases as a power of the dimension (curse of dimensionality).

Because of this problem, optimal proposals most likely fall in the vicinity of the current sampling locations. In order to generate such points we, by default, use a *centroid* algorithm: take the average location of $N_d + 1$ randomly selected samples, and perturb them in each dimension by the coordinate difference to one the samples multiplied by a draw from an exponential distribution with parameter $1/\lambda$. Here a lower λ increases the spread of the proposed locations. A fraction of the locations are drawn from a uniform distribution within the original prior boundaries, in case a region of high posterior has not yet been captured by the current samples.

JCAP10(2023)021

For highly non-Gaussian distributions this method of proposing points tends not to be exploratory enough. In these cases we resort to drawing proposals uniformly within the prior volume.

Lastly also provide a method to generate Gaussian-distributed proposals given an estimate of the mean and covariance matrix of the posterior, if such information happens to be known.

We notice that alternative approaches to maximizing the acquisition function exist. In [65], the sampling locations with high acquisition function value are picked out of an MCMC of the mean GP model.

4.2 Parallelization

The naive approach of using the acquisition function presented above is to acquire and evaluate sampling locations in sequence, with each acquisition step consisting of the evaluation of the true posterior distribution (and updating the GP model) in order to obtain the next candidate for a sampling location. However, as often multiple processing units (either on the same or across different machines) are available, we can make this algorithm more efficient by attempting to propose *batches* of sampling locations, so that the true posterior, which is expected to be the largest source of computational cost, can be evaluated in parallel.

There have been many different proposals for batch acquisition for GPs in the past which can broadly divided into two categories: algorithms like [81–84] construct an acquisition function which can be optimized for several points at once. However, for a d-dimensional posterior distribution acquiring q points at once involves global optimization in $d{\cdot}q$ dimensions which obviously becomes computationally prohibitive even if d and q are not extremely large.

The second category [83–85] works by sequentially acquiring multiple points without having to sample from the posterior distribution in between and afterwards evaluating the true posterior at the gathered locations in parallel. We will be using one of these methods called the *Kriging believer* method [84].

The Kriging believer method. The fundamental assumption of the Kriging believer method (similarly to our assumption when constructing the acquisition function) is that the value of the posterior distribution in any point roughly equals the predicted mean of the GP. We can therefore acquire a batch of points by sequentially (1) obtaining a maximum of the acquisition function at $\boldsymbol{x}_*$, (2) assuming for it a log-posterior evaluation equal to $\mu(\boldsymbol{x}_*)$, (3) adding it to an intermediate *augmented* GP (thereby producing a different new maximum of the *augmented* acquisition function), and repeating until the desired number of locations has been proposed. This method will be increasingly accurate as more samples are added to the GP so that $\mu(\boldsymbol{x}_*)$ approaches the true $\log p(\boldsymbol{x}_*)$. An illustration of the Kriging believer algorithm sampling on the log of a normal distribution is shown in figure 3.

The obvious advantage of this method, as discussed above, is that the true posterior can be evaluated in parallel for the acquired locations. This is beneficial as we expect the true posterior evaluations to dominate the computational cost in most scenarios. In addition, there is another source of speedup: since adding new mean-valued samples does not change the optimal hyperparameters of the GP according to equation (2.7), there is no point to re-fitting them (see section 5.3).[11]

[11] On top of that, the necessary step of inverting the kernel matrix after adding new points, in order to get predictions using equations (2.5) and (2.6), could be accelerated by taking advantage of the fact that the inverse of the previous kernel matrix is known, using a fast, blockwise matrix inversion formula. To our knowledge this has not been pointed out in the past. In our case, the amount of possible time savings is small.

JCAP10(2023)021

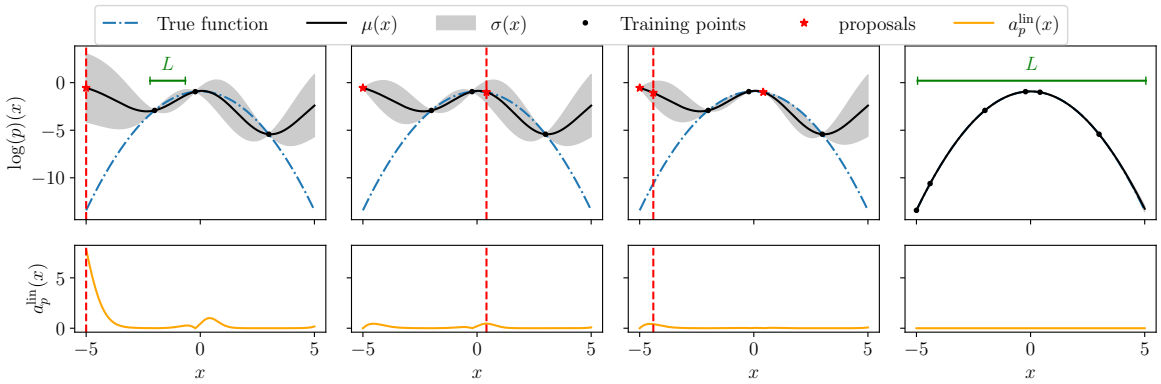


Figure 3. Illustration of the Kriging believer method. Three points are acquired sequentially (three left plots) by using the prediction from the GP instead of evaluating the posterior at each iteration. After the three samples have been acquired the posterior function can be evaluated at these points (right). The hyperparameters of the GP regressor only need to be refit at the last step. Obviously using this approach comes at the expense of requiring more points to converge (e.g. the third point did not add much information and is unlikely to have been selected after the second one if using sequential acquisition). This can however be compensated by the computation time that is saved by both acquiring points faster and evaluating the posterior in parallel. The characteristic length-scale L of the kernel increases as more samples are added, which aids the better fit in the right panel.

In terms of the precise size of the batch, there is evidently a trade-off between the speedup gained by not refitting the GP's hyperparameters at every iteration, and the inaccuracy of the GP's mean prediction making the active sampling less efficient as the number of Kriging believer steps grow. Due to the loss of accuracy in the predictions, more samples are required to converge to the true distribution, but this is compensated by the speedup achieved through parallel evaluations of the posterior. Overall this results in a smaller number of iterations (hence a smaller wall-clock run time) than sequential learning, as long as the size of the batches is kept reasonable.

We find that a batch size corresponding to at most the number of dimensions of the inference problem N_d works reasonably well. We therefore set the standard number of Kriging believer steps to the minimum between N_d and the number of parallel processes.

4.3 Convergence criterion

The last component of our algorithm is its *convergence criterion*, which should terminate it as soon as (or at least not much later than) the GP has reached sufficient precision at modelling the log-posterior. Precision could be assessed as the reduction in the variance of a GP-predicted global posterior quantity such as the evidence $\int p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$. Analytical computation of these quantities in terms of the GP are usually not possible, e.g. in our case because of the modelling of the log-posterior instead of the posterior itself, or because the product of a GP times arbitrary priors does not have a closed-form integral in general. Numerical approaches would involve MC samples of the GP-modelled posterior, which come at a reasonably-small computational cost, but whose use for the convergence criterion would involve obtaining them at (nearly) every iteration.

A much cheaper convergence criterion would involve computations using the much smaller set of current and/or proposed GP samples. We propose one such criterion, that we call `CorrectCounter`, based on observing the accuracy of the learning process and stop-

JCAP10(2023)021

ping when the model does not seem to learn any new information. We will show how that the speedup in this case does not necessarily come at the cost of precision.

The assumption here is that our algorithm stops learning if the GP's predictions at newly acquired sampling locations $\boldsymbol{x}$ repeatedly match the value of the true log-posterior $\log p(\boldsymbol{x})$ distribution to close approximation. We set a threshold using relative and absolute tolerances $\epsilon_{\rm abs}$, $\epsilon_{\rm rel}$ such that

$$|\mu_{\rm GP+SVM}(\boldsymbol{x}) - \log p(\boldsymbol{x})| \overset{!}{<} \epsilon_{\rm abs} + |y_{\rm max} - \mu_{\rm GP+SVM}(\boldsymbol{x})| \cdot \epsilon_{\rm rel}\,, \tag{4.4}$$

where $y_{\rm max}$ is the largest log-posterior from the current GP sample, and $\mu_{\rm GP+SVM}(\boldsymbol{x})$ is the GP's prediction at $\boldsymbol{x}$ before the GP has been fit to this point. This criterion can be computed at virtually no cost, since both $\mu_{\rm GP+SVM}(\boldsymbol{x})$ and $\log p(\boldsymbol{x})$ have been computed as part of the acquisition procedure. If this condition is satisfied a few times in a row we consider the model converged and stop the algorithm. Convergence in this case means a guarantee that (on average) new evaluations of the GP will at least approximately comply with the true posterior at the same location (as opposed to convergence meaning stability of some global quantity).

Similarly to the discussion in sections 3.3 and 4.1.2, the behaviour of this convergence criterion is sensitive to the dimensionality N_d of the problem. As explained in appendix A, since the dynamic range of a log-posterior enclosing a given probability mass grows with dimensionality, the effect of a constant $\epsilon_{\rm abs}$ will become more stringent as dimensionality increases, making the criterion fail to report as converged GP models that already very precisely characterise the posterior. In appendix A we propose a way to relax $\epsilon_{\rm abs}$ in a dimensionally-consistent way. The relative thresold $\epsilon_{\rm rel}$ should not be affected by dimensionality, and it is fixed to 0.01. In both cases, we also give the user the option to set their own values for the convergence criterion.

On the other hand, as the number of dimensions N_d increases, correctly mapping the tails of the distribution becomes increasingly more important (for a detailed discussion see appendix A), while the surrogate model tends to converge first around the maximum of true posterior distribution. The tails usually remain underrepresented at first and only get explored later in the acquisition procedure. A higher dimensionality therefore makes it likelier to acquire a batch of consecutive correctly-predicted points in a non-converged GP model around the top of the mode. We account for this by increasing the number of times points have to be predicted correctly to claim convergence to $n = N_d/2$ (with the exception of fixing $n = 4$ for low dimensionality, $N_d < 8$). This reduces the risk of neglecting convergence at the tails.

We tested the `CorrectCounter` criterion on a set of correlated Gaussians in 2, 4, 8, 12 and 16 dimensions, generated as explained in section 6.1. We target a KL divergence with respect to the true Gaussian distribution of less than 5%. As shown in figure 4, we achieve such threshold with the settings described above for the tolerances and the number of consecutive correct predictions, at least for the range of dimensionality targeted in this study. Towards higher dimensionality there is a trend to converge before the convergence curve flattens out entirely, which hints at the need for more sophistication in dealing with dimensional consistency. We leave this for future work.

We also note that we have written a criterion based on the costlier KL divergence (see appendix B), which we provide as an alternative option. This alternative criterion is based on the posterior emulation stabilizing over multiple subsequent steps (defined through the KL divergence being below some critical threshold). This criterion comes with its own sets of

JCAP10(2023)021

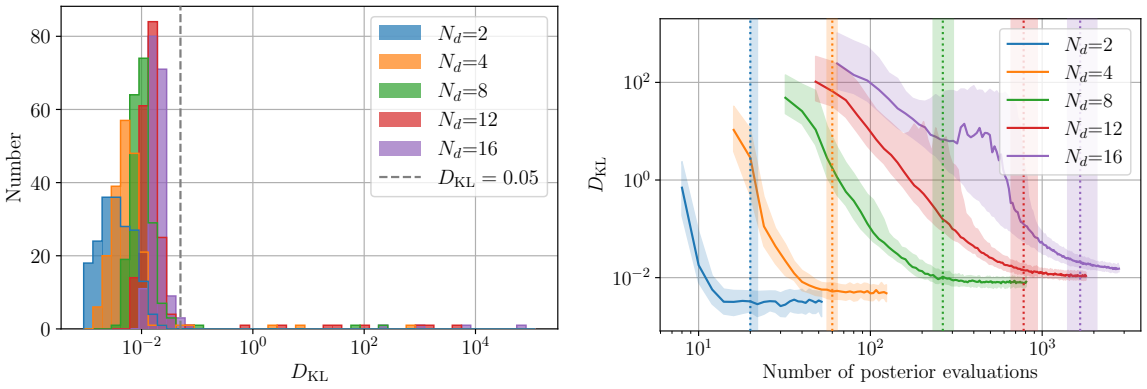


Figure 4. *Left*: distribution of KL divergences between GP models and their correspoding true posterior at `CorrectCounter`-reported convergence, for $N_d = 2, 4, 8, 12, 16$-dimensional random correlated Gaussians (200 draws per dimensionality). Only a small fraction ($< 5\%$) surpass our target value of $D_{\rm KL}^{\rm sym} = 0.05$. *Right*: medians (solid lines) and interquartile ranges (shaded bands) of the KL divergences between GP models and their correspoding true posterior, for the same sets of Gaussians, as function of their number of accepted (finite) samples. The dashed vertical lines indicate the median number of accepted steps at which `CorrectCounter` reports convergence, and the shaded vertical bands the respective interquartilic ranges. As expected there is a trend towards higher values of $D_{\rm KL}^{\rm sym}$ visible as N_d increases, but it is well under control for the dimensionalities targeted in this study.

challenges, such as incorrectly detecting convergence when non-informative points are added to the GP or the costly nature of its computation. Nonetheless it can be preferable when the log-posterior function is extremely expensive to evaluate or when the posterior distribution exhibits unusual features, as this convergence criterion is not only sensitive to the acquired samples but also to the hyperparameters of the GP.

5 The full algorithm

In this section we present the full structure of the algorithm, entailing the generation of the initial set of training samples (section 5.1), the main acquisition loop that sequentially looks for optimal samples and checks convergence (section 5.2), and the final generation of a Monte Carlo sample of the trained GP surrogate model of the posterior, which can be used to get marginalised quantities (section 5.3), together with a comparison of computational costs of this algorithm against those of classic Monte Carlo.

5.1 Initial training set

In order to start the sequential acquisition of points we need an initial training set containing samples from our posterior distribution. These do not have to be very informative samples but need to be finite (according to the definition in section 3.3) and uncorrelated, in order to generate some very crude but meaningful initial interpolation of the log-posterior distribution.

Of course we want to choose this sample such that the ratio of finite to infinite log-posteriors is reasonably high, in order not to waste too many posterior evaluations on the initial point generation. In low dimensions and with small priors compared to the size of the mode, any random generator (such as draws from the prior itself, or from a uniform distribution within the prior bounds) would produce initial samples satisfying the requirement above. As the ratio of the prior to posterior volume grows with the number of dimensions,

JCAP10(2023)021

randomly drawing a finite point from the prior becomes increasingly unlikely. In this case, prior knowledge of the posterior can be incorporated, usually in the form of a "reference" distribution which is a rough guess of where the mode might be (the same that is commonly used to generate initial points for MCMC). In general any guess for reasonable parameter values that lead to a finite posterior can be used, which can be obtained from physical considerations of the underlying model.

5.2 Main algorithm

In algorithm 1 we show the main algorithm used within the `GPry` tool in pseudo-code, consisting mostly of the optimization and acquisition loops (the latter based on the Kriging believer approach). This pseudo-code mostly summarizes the ideas which are explained in the corresponding sections 3 and 4.

Note that the $n_{r,\mathrm{GP}}$ starting locations for the optimization of the hyperparameters in line 4 are sampled logarithmically in the hypervolume. The step of line 4 is currently the most expensive step, scaling as N_s^3 due to the required repeated matrix inversion required for computing $\log p(\theta|\boldsymbol{X}, \boldsymbol{y})$. This is why we only perform this step every n_{opt}-th time, and otherwise we optimize the hyperparameters starting only from the previous best fit. The next most expensive step is the acquisition function optimization in line 12, and scales approximately as N_s^2 due to the repeated evaluation of the acquisition function requiring the evaluation of $a(\boldsymbol{x})$, which itself requires matrix multiplications.

5.3 Modelling the marginalized posterior

As mentioned above, to compute marginalized 1D/2D posteriors, we have to compute a high-dimensional integral of our emulated posterior (see section 2.1). This can be achieved by integrating the GP numerically through the creation a Monte Carlo sample, either based on nested sampling, Metropolis Hastings sampling, or (using the backward differentiable nature of the GP) even Hamiltonian sampling. As `GPry` is interfaced with the `Cobaya` package [86], its standard samplers can also be used to generate the final MCMC sample. Currently, this sampling is performed using the GP's mean prediction according to equation (3.4) as the posterior distribution to sample.[12]

One important question that such an approach poses, however, is whether the emulation of the posterior with the GP with subsequent sampling of the surrogate posterior will be computationally more efficient than the direct sampling of the true posterior. For this, let us use a simple back-of-the-envelope computation. Consider the time to run a full sampling of the true posterior as $N_t t_t$, where t_t is the approximate time for a single evaluation and N_t the total number of samples required. Instead, the time to run a sampling of the GP posterior can be estimated as $N_g t_g$, where t_g is the average time for a single GP evaluation and N_g the total number of required GP samples. Additionally, and crucially, there is the additional overhead of constructing the GP in the first place, which we will denote simply as T_o for now (we will discuss this in more detail later). In that case, the construction of a GP is advantageous if

$$T_o + N_g t_g < N_t t_t \,. \tag{5.1}$$

Typically it can be assumed that $t_g \ll t_t$ except for very simple toy models. Furthermore, typically $N_g \simeq N_t$ if one uses MCMC/nested sampling methods to sample the GP, or even

[12] Technically, the information that is available through the covariance of the GP could be used to obtain an estimate of the uncertainty of emulation on our final posterior sample. As the acquisition procedure only stops if the posterior mode is mapped accurately enough, this assures that at convergence this variance is sufficiently small to safely be neglected.

JCAP10(2023)021

Input: $\boldsymbol{X}$ (initial samples), $\boldsymbol{y}$ (initial log-posterior values)

[1] **for** $n < N_{\max}$ **do**

[2] fit SVM with $\boldsymbol{X}$, $\boldsymbol{y}$

[3] **every** $n_{\text{opt}} - - -$ th **time**

[4] find $\theta_{\text{MAP}} = \text{argmax}[\log p(\theta|\boldsymbol{X}, \boldsymbol{y})]$ from $n_{r,\text{GP}}$ starting locations — equation (2.7)

[5] **otherwise**

[6] find $\theta_{\text{MAP}} = \text{argmax}[\log p(\theta|\boldsymbol{X}, \boldsymbol{y})]$ from last best-fit — equation (2.7)

[7] **end**

[8] `GP_fit`($\boldsymbol{X}$, $\boldsymbol{y}$)

[9] $\boldsymbol{X}_{\text{new}} = [\,]$

[10] $\boldsymbol{X}_{\text{lie}} = \boldsymbol{X}$ and $\boldsymbol{y}_{\text{lie}} = \boldsymbol{y}$

[11] **repeat** M **times**

[12] find $\boldsymbol{x}_{\text{add}} = \text{argmax}[a(\boldsymbol{x})]$ starting from $n_{r,\text{acq}}$ starting locations

[13] $\boldsymbol{X}_{\text{lie}}$ *append* $\boldsymbol{x}_{\text{add}}$ and $\boldsymbol{X}_{\text{new}}$ *append* $\boldsymbol{x}_{\text{add}}$

[14] $\boldsymbol{y}_{\text{lie}}$ *append* $\boldsymbol{\mu}(\boldsymbol{x}_{\text{add}})$ — Kriging believer

[15] `GP_fit`($\boldsymbol{X}_{\text{lie}}$, $\boldsymbol{y}_{\text{lie}}$)

[16] **end**

[17] $\boldsymbol{y}_{\text{true}} = \log \mathcal{L}(\boldsymbol{X}_{\text{new}}) + \log \pi(\boldsymbol{X}_{\text{new}})$ — parallelizable

[18] $\boldsymbol{X}$ *append* $\boldsymbol{X}_{\text{new}}$

[19] $\boldsymbol{y}$ *append* $\boldsymbol{y}_{\text{true}}$

[20] **if** *is_converged* (e.g. equation (4.4)) **then** *break*

[21] **end**

[22] Sample $\mu(\boldsymbol{x})$ with MC sampler

[23] **return** *MC sample*

[24] **Function** *GP_fit*($\boldsymbol{X}$, $\boldsymbol{y}$)

[25] Compute $K^{-1} = \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{X}|\theta_{\text{MAP}})^{-1}$ — matrix inversion

[26] $\mu(\boldsymbol{x}) = \mu_{\text{GP+SVM}}(\boldsymbol{x})$ — equations (2.5) and (3.4)

[27] $\sigma(\boldsymbol{x}) = \sqrt{\Sigma_{\text{GP+SVM}}(\boldsymbol{x})}$ — equations (2.6) and (3.5)

[28] $a(\boldsymbol{x}) = \exp[2\zeta\mu(\boldsymbol{x})]\{\exp[\sigma(\boldsymbol{x}) - \sigma_n] - 1\}$ — equation (4.3)

[29] **end**

Algorithm 1. The `GPry` algorithm in a condensed format, omitting the internal transformations that are made to the data. M is the number of Kriging believer steps made in each iteration. The overhead of the algorithm is dominated by the computations performed in lines 12 and 4.

$N_g \ll N_t$ if one can use Hamiltonian MC methods on the GP but not on the true posterior. Thus, as long as T_o remains reasonably lower than $N_t t_t$ (the total runtime of the MCMC), the use of a GP would always be advantageous. It is thus crucial to obtain a precise estimate for the overhead time T_o. This overhead depends strongly on the dimensionality of the problem, the non-Gaussianity of the posterior, and the underlying machine executing the code.

Looking at the timing information from the multivariate Gaussian cases of section 6.1, the overhead was dominated by the numerical optimization of the acquisition function (line 12 of algorithm 1), taking very roughly $100s \cdot (N_s/100)^{2.4}$ (to give an order-of-magnitude estimate). The next most important factor, the optimization of hyperparameters (line 4 of al-

JCAP10(2023)021

gorithm 1) only takes around $3s \cdot (N_s/100)^{3.2}$ (order of magnitude) in total. It has a smaller pre-factor since it is only performed every $n_{\rm opt}$-th iteration, while the acquisition optimization is performed $M \cdot n_{\rm r,acq}$ times per iteration, see algorithm 1. It is thus comparatively irrelevant for $N_s \ll 10^4$, which is almost always the case for the range of dimensionalities considered in this study.

In figure 5 we report the approximate expected total runtime of `GPry` compared to the `Cobaya` implementation of the MCMC sampler `CosmoMC` [1, 2, 86] and the nested sampler `PolyChord` [9, 10] (via its `Cobaya` interface). For each dimensionality, these estimates were generated by drawing a large set of random multivariate Gaussians, and computing the distrubution of total evaluations needed for convergence (according to their respective default criteria) for MCMC, `PolyChord` and `GPry`. We multiply these numbers of posterior evaluations with the posterior evaluation times on the x-axis and add the overhead of each algorithm to get the total runtimes on the y-axis. For `GPry`, the computational overhead is caused by the optimization of the acquisition function and the fitting of the GP hyperparameters, and it is constant with respect to the posterior evaluation time, producing the particular shape of the curve. We neglect the overhead of MCMC and `PolyChord` as it is tiny compared to the overhead of `GPry` [86].

For example, in the case of a $N_d = 12$ multivariate Gaussian, `GPry` would outperform the MCMC (which requires $\approx 1.5 \cdot 10^5$ evaluations) for posterior evaluation times larger than ~ 0.1 seconds. Comparing to the average runtime of a cosmological code such as `CLASS`, on average we find a significant speedup all the way up to 16 dimensions.

Note that in figure 5 we show single-core performance with as many Kriging believer steps as dimensions (while still evaluating the posterior sequentially). The curves shown for MCMC and especially for `PolyChord` would drop almost proportionally to the number of cores available, while `GPry` does not scale quite as well. However, for a similar amount of computational resources, up to a number of processes similar to the dimensionality of the problem, these results are expected to hold in order of magnitude. While the runtime of MCMC and `PolyChord` is dominated by the posterior evaluations, the overhead of `GPry` is considerable and might scale differently depending on the underlying architecture. Further improvements in runtime could be made by optimizing the underlying GP implementation.

6 Examples

After having discussed the design of the `GPry` code in sections 3 to 5, we now demonstrate the performance of the code using a variety of examples, both Gaussian and non-Gaussian distributions considered in the literature, as well as examples from cosmological applications.

For each of the examples in this section, we will analyze the performance of `GPry` in terms of convergence by producing a number of runs with identical `GPry` settings (same choices of kernel functions, acquisition function and other training settings) but different random seeds, so that they start from different initial training samples (uniformly drawn from the prior) and find generally different optima for the acquisition function and the GP hyperparameters (maximizations are started from random starting positions). On top of the intrinsic variability between runs, the covariance matrices and means of the Gaussian examples in section 6.1 and the log-Gaussian ones in section 6.2.1 are drawn randomly for every run to make the tests more robust, whereas for the rest of non-Gaussian and multimodal examples, as well as the cosmological ones, the posteriors are fixed.

JCAP10(2023)021

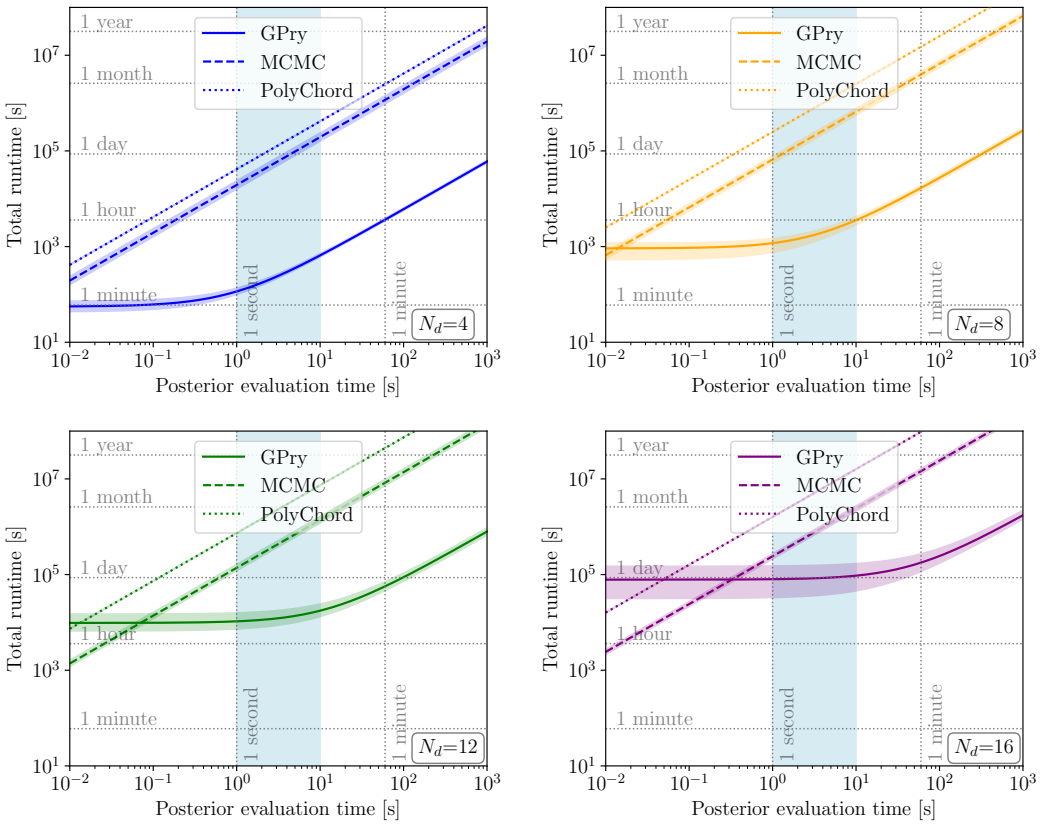


Figure 5. Order of magnitude estimate of total runtime comparison of `GPry` with the MCMC sampler `CosmoMC/Cobaya` and the nested sampler `PolyChord` (via its `Cobaya` interface). The comparison is done for multivariate Gaussians of various dimensionalities, and shows the median as a line and the 25% and 75% quantiles as a shaded area. The comparison is run with only a single CPU, but the orders of magnitude hold for similar computational resources for all three methods. The light blue band gives an approximate range of computation times of standard cosmological codes (like `camb` or `CLASS`) which depend strongly on the considered model and observables. Note that while MCMC and `PolyChord` are dominated by the posterior evaluation time everywhere, `GPry` is dominated by overhead for small posterior evaluation times.

6.1 Multivariate Gaussians

The example of a multivariate Gaussian distribution is enlightening as a benchmark for the average performance of the GP, as it can quite trivially be scaled with dimensionality and many likelihood functions can — at least around their maximum — be reasonably well approximated by Gaussian distributions. We can thus use it as a benchmark for performance and accuracy as a function of dimensionality, as well as to model critical scalings such as that of the ζ parameter from section 4.1, the factors involved in equation (4.4), and the timings relevant for section 5.3 (see appendix A).

We generate correlated multidimensional Gaussians with log-likelihood function

$$\log \mathcal{L}(x_0, \ldots, x_n) = -\frac{(\boldsymbol{x} - \mathbb{m})^T \boldsymbol{C}^{-1} (\boldsymbol{x} - \mathbb{m}) + \log((2\pi)^n |\boldsymbol{C}|)}{2} \tag{6.1}$$

by drawing a random covariance matrix that satisfies

$$\boldsymbol{C}_{i,j} = \sigma_i \sigma_j \mathrm{corr}_{i,j} \tag{6.2}$$

JCAP10(2023)021

where $\text{corr}_{i,j}$ is a randomly drawn correlation matrix with uniformly drawn eigenvalues,[13] and the standard deviations are uniformly drawn as $\sigma_i \in [0, 1]$. The mean vector $\mathfrak{m}$ is set to 0 and the prior fixed to $5\sigma_i$ in each direction. This ensures that the mode is centered within the prior. The case in which parts of the mode are cut off by the prior is discussed in section 6.2. We then conducted tests in $\{2, 4, 8, 12, 16\}$ dimensions recording the Gaussian KL divergence of equation (B.3), the number of posterior evaluations, and the overall overhead. The final results were already shown in figures 2, 4 and 5.

6.2 Non-Gaussian distributions

One of the main goals of our algorithm is to be robust with regards to the functional shape of the posterior distribution. We therefore tested the code also on non-Gaussian distributions with varying degrees of pathological features. All adopted priors are flat in the respective parameters.

6.2.1 Log-transformations

Our first example of a non-Gaussian feature is motivated by a common occurrence in Physics. In many applications, there are free scales in the problem which are not known across one or more dimensions in the parameter space. For these parameter one usually samples their logarithm with a flat prior (which is equivalent to imposing a logarithmic prior), distributing the prior probability density evenly across multiple orders of magnitude. If the likelihood is Gaussian in the (linear) parameter, this typically leads to a log-Gaussian distribution of the form

$$10^x \sim \mathcal{N}(\mu, \sigma) \tag{6.3}$$

across some dimensions.

To test whether our algorithm is robust with respect to these kind of likelihoods we drew randomly correlated 4-dimensional Gaussians according to equation (6.1) where the first two dimensions $\{x_0, x_1\}$ are sampled in log-space. The performance of the algorithm in this case is shown in figure 6. We recover the correct posterior shape and manage to sample the posterior accurately with only around 200 samples. An additional benefit of this test is that it shows that our algorithm is robust with respect to cases where the mode has a hard prior cutoff ($|x_i| < 2$ in this example).

In order to explore the limits of our algorithm, we perform the same test in 8 dimensions, with 4 of them being sampled in log-space. We set a budget of at most 2000 posterior evaluations. Figures 7 and 8 show the distribution of $D_{\text{KL}}^{\text{sym}}$ as a function of the number of posterior samples and at convergence for the default settings of `CorrectCounter` proposed in appendix A ($\epsilon_{\text{abs}} = 0.01[\Delta\chi^2](1)$, $\epsilon_{\text{rel}} = 0.01$) and for five times more accurate settings ($\epsilon_{\text{abs}} = 0.002[\Delta\chi^2](1)$, $\epsilon_{\text{rel}} = 0.002$). With default settings convergence tends to be declared prematurely while the more accurate settings mitigate this problem. Figure 9 shows corner plots of two example runs at declared convergence by `CorrectCounter` with default settings, one where convergence is declared prematurely while the mode is still being explored, and one where the mode has been characterized correctly (our target $D_{\text{KL}}^{\text{sym}}$ of 0.05 has not been reached in either case). This higher-dimensional example highlights two limitations of our algorithm. On one hand, the overhead of the GP regressor after such a large number of samples

[13] They are uniformly drawn between 0 and 1, then multiplied by a normalization constant such that their sum equals the number of dimensions, in order to avoid cases where many of the eigenvalues are close to zero simultaneously.

JCAP10(2023)021

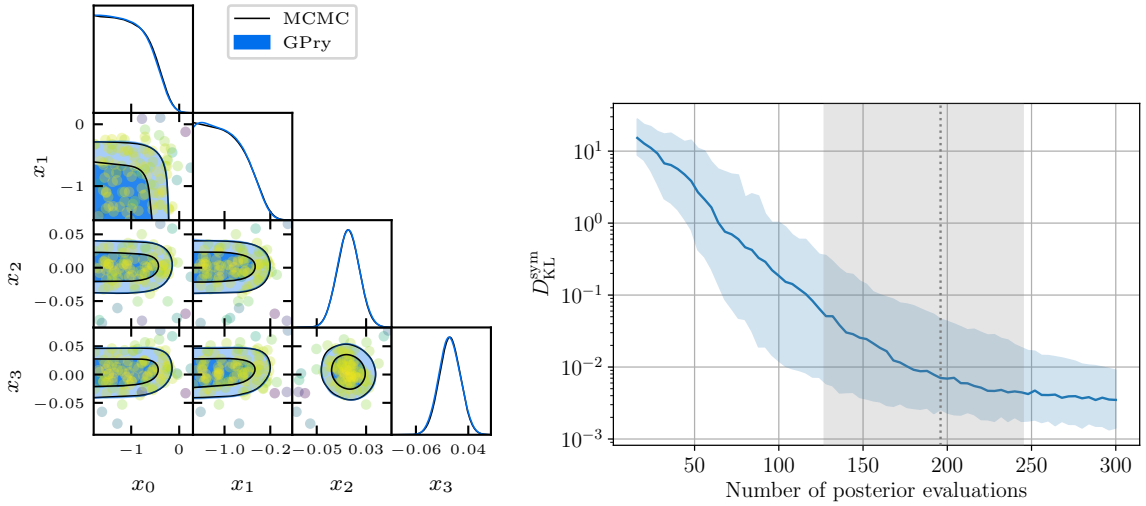


Figure 6. 2d and 1d posterior distributions of a typical four-dimensional log-gaussian distribution (left) at convergence (180 posterior evaluations), and convergence with respect to the true model against number of accepted steps for 200 realizations, where the blue band shows the $\{25, 50, 75\}$%-quantiles for the KL-divergence, and the grey band does the same for convergence as defined by the `CorrectCounter` criterion. (Right) The posterior distribution is cut off in x_0 and x_1, which is correctly captured by `GPry`.

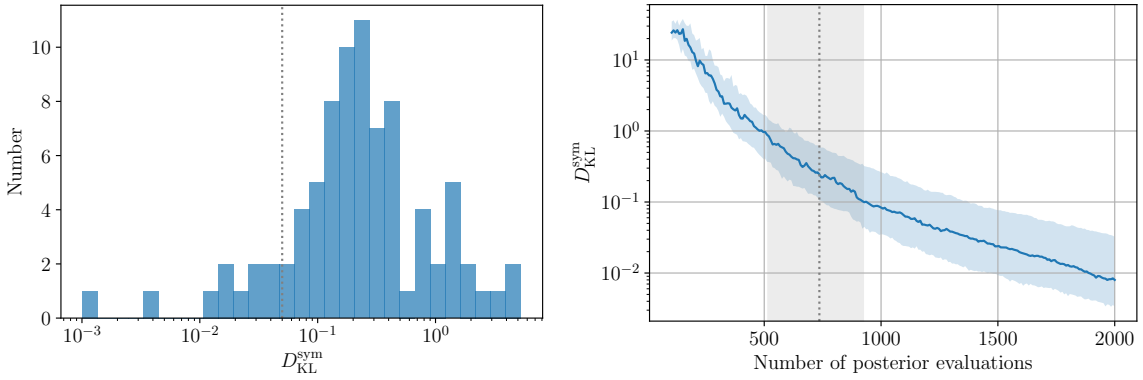


Figure 7. *Left*: distribution of KL divergences at convergence, according to `CorrectCounter` with default settings, of the 8-dimensional log-gaussian draws. *Right*: convergence against number of accepted steps, where the blue and grey bands are defined as in figure 6. Even though most of the runs converge within an acceptable accuracy, our convergence criterion declares convergence prematurely. 83 of the 84 runs we performed were declared as converged. Figure 9 shows the contour plots for a two examples of a prematurely and a correctly reported convergence with these default settings of `CorrectCounter`.

(~ 2000) reduces the advantage of our algorithm with respect to traditional MC samplers. On the other hand, the prematurely reported convergence seems to be a consequence of the combination of long tails and higher-dimensionality: the sequential optimization algorithm fails to propose points at the tails, which occupy a small fraction of the hypervolume in higher dimensionality. An active sampling scheme that explores the parameter space more thoroughly may mitigate this problem [87].

JCAP10(2023)021

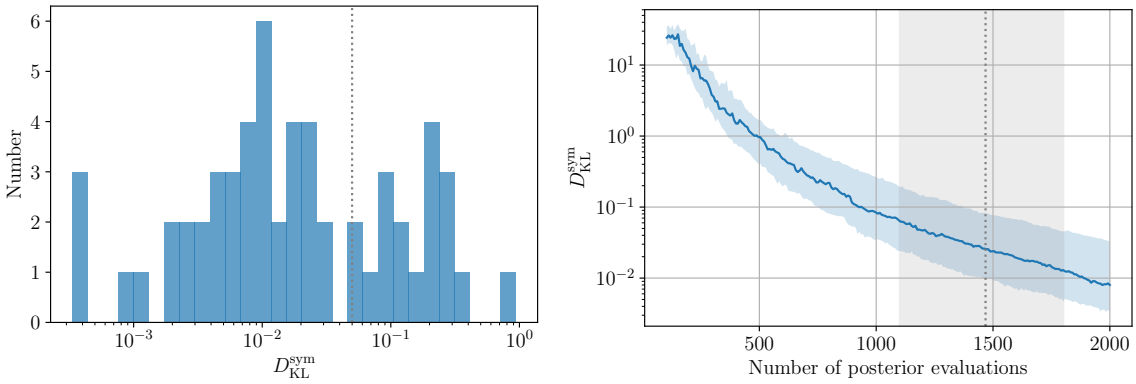


Figure 8. Same as figure 7, with the required accuracy of the `CorrectCounter` convergence criterion increased by a factor of 5 ($\epsilon_{\mathrm{abs}} = 0.002[\Delta\chi^2](1)$, $\epsilon_{\mathrm{rel}} = 0.002$). The KL-divergence of the converged runs is much better than in figure 7. However, only 58 of 84 runs are declared as converged by the convergence criterion when the evaluation budget has been exhausted.

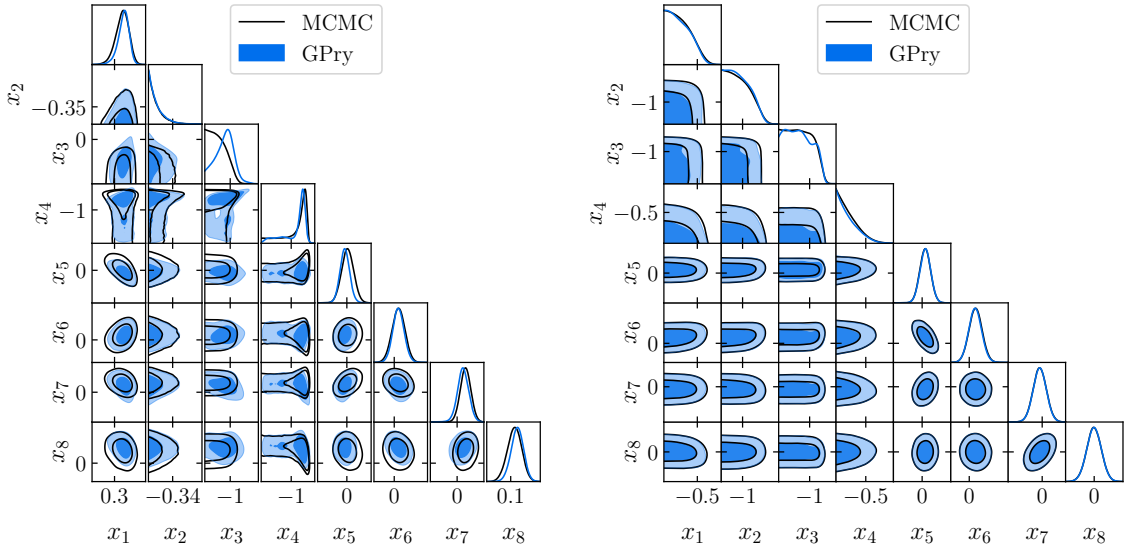


Figure 9. Triangle plots of the 8-dimensional log-Gaussian example at convergence according to `CorrectCounter` with default settings. *Left*: premature convergence (928 posterior evaluations, $D_{\mathrm{KL}}^{\mathrm{sym}} = 1.49$). The mode has been found but not been explored completely. *Right*: correct classification as converged (736 posterior evaluations, $D_{\mathrm{KL}}^{\mathrm{sym}} = 0.08$). Even though our target $D_{\mathrm{KL}}^{\mathrm{sym}}$ of 0.05 has not been reached the contours are still recovered correctly.

6.2.2 Curved degeneracies

We also investigated whether more general curved degeneracies with different length-scales in the different parameter dimensions could be modeled correctly. We use three examples.

1. Example one is a "banana"-shaped curved degeneracy, a slightly modified version of a benchmark found in [88], which is based upon an eight-order polynomial in the exponent and exhibits a long tail in the $x_1 \approx 4x_0^4$ direction. The log-likelihood of this

JCAP10(2023)021

distribution is

$$\log \mathcal{L}(x_0, x_1) = -(10 \cdot (0.45 - x_0))^2/4 - (20 \cdot (x_1/4 - x_0^4))^2 \,. \tag{6.4}$$

Figure 10(a) shows how `GPry` performs at sampling this distribution. The posterior shape is correctly recovered (at around ~ 40 posterior evaluations) and shows good match with MCMC.

2. Example two has a fourth-order polynomial in the exponent, but in this case the parameters are tuned in order to exhibit an extremely sharp cutoff away from the degeneracy direction and an extremely long tail along the degeneracy. This particularly pathological case is the Rosenbrock function, commonly used to test minimization algorithms. It is described by

$$\log \mathcal{L}(x_0, x_1) = -\frac{1}{2}\left[(a - x_0)^2 + b(x_1 - x_0^2)^2\right] \,, \tag{6.5}$$

where we set the parameters to their typical values of $a = 1$ and $b = 100$. It has a long, narrow, parabolic "ridge" along which the maximum lies. Since the parabolic degeneracy direction changes throughout, this is a good test for the robustness of `GPry` for distributions which do not show a clear axis of correlation or symmetry. We impose a uniform prior between $[-4, 4]$ for both x_0 and x_1. The results for this posterior are displayed in figure 10(b), which shows that even such a pathological posterior function can be accurately described by the `GPry` code, while still requiring a reasonably small number of posterior evaluations (~ 60).

3. The third example is a sharp ring-like posterior. The log-likelihood of this distribution is given by

$$\log \mathcal{L}(x_0, x_1) = -\frac{1}{2}\left[\frac{\left(\sqrt{x_0^2 + x_1^2} - \mu\right)^2}{\sigma} + \log(2\pi\sigma^2)\right] \,, \tag{6.6}$$

with $\mu = 1$ and $\sigma = 0.05$. This produces a ring-shaped posterior distribution with the two very different scales μ (the location of the ring) and σ (the width of the ring). Furthermore the maximum of this function is the ridge of the ring, making it especially hard to capture the full mode and sample the distribution correctly. Nevertheless our algorithm efficiently captures this mode within ~ 75 posterior evaluations and agrees well with MCMC.

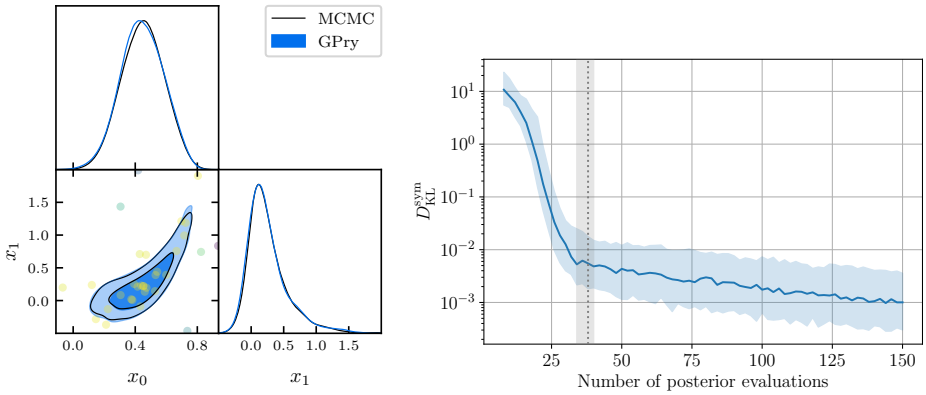
We note that for all of these non-Gaussian examples more posterior evaluations are required for convergence compared to the multivariate Gaussian examples with the same dimensionalities. This is because the surrogate model requires more training samples to correctly capture the non-trivial shape and the extended tails.

6.2.3 Multi-modal posteriors

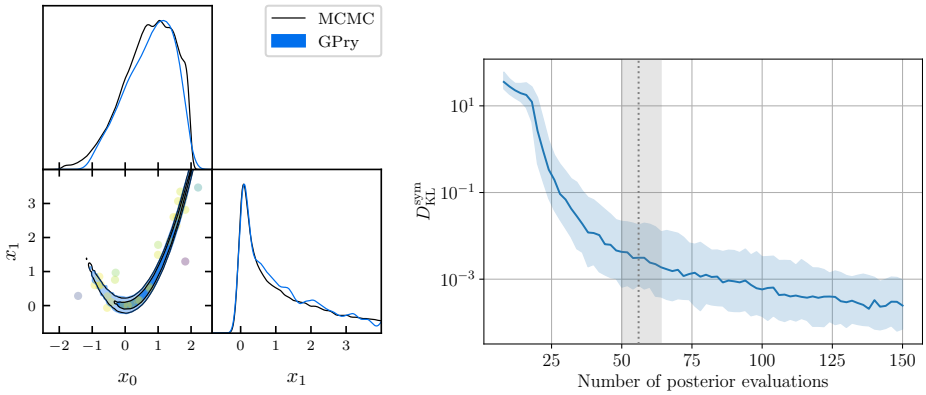
We also want to check the robustness of the `GPry` tool against mild multi-modality. For this, we make use of a modified Himmelblau function (which is commonly used in minimization studies). The log-posterior is defined as

$$\log \mathcal{L}(x_0, x_1) = -\frac{1}{2}\left[a \cdot (x_0^2 - x_1 - 11)^2 + (x_0 + x_1^2 - 7)^2\right] \tag{6.7}$$

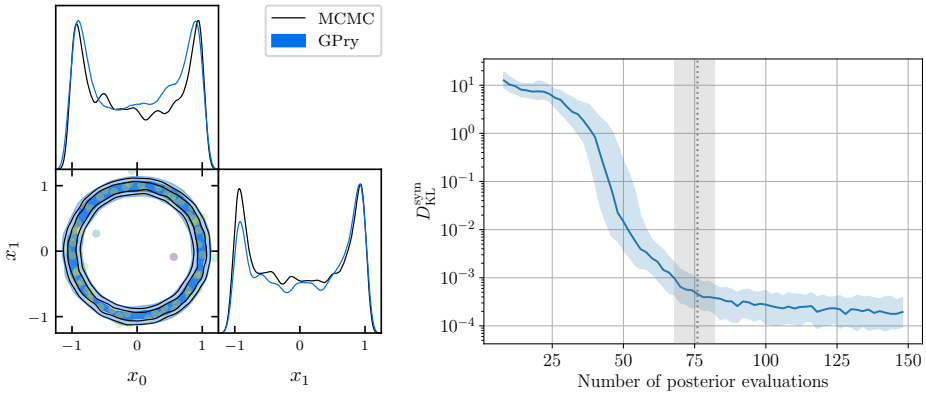
JCAP10(2023)021



(a) "Banana"-shaped degeneracy.



(b) Rosenbrock likelihood.



(c) Gaussian ring.

Figure 10. Performance tests for the non-Gaussian likelihoods with curved degeneracies presented in section 6.2.2. For each case *left*: 2d and 1d posterior distributions for typical converged runs (40, 62 and 68 posterior evaluations, respectively); *right*: convergence against number of accepted steps, where the blue and grey bands are defined as in figure 6. Even though these distributions display very non-Gaussian behaviours, their shape is correctly recovered without needing a large number of samples.

JCAP10(2023)021

where the term in the brackets corresponds to the Himmelblau function for $a = 1$. We include this scaling factor a in the first term in order to create a "mild" multi-modal posterior ($a = 0.1$) with relatively connected modes which we compare to the full Himmelblau function ($a = 1$).

We show the results of sampling this distribution in figures 11 to 13. We observe that many runs do not correctly capture the modes. In general, we can distinguish three modes of failure of the `GPry` algorithm, nicely demonstrated in these examples.

1. The algorithm can find and sample all modes, but not weigh them correctly in relation to each other. This is clearly visible in figure 11, where all modes are reliably sampled, but the 1D posterior reveals the incorrect weighting.

2. The `CorrectCounter` criterion may falsely claim convergence and stop the sampling when some of the modes have been well explored, while further sampling might have revealed modes that have not been mapped. This is shown in figure 12, where we compare the convergence to the true distribution (through the $D_{\mathrm{KL}}^{\mathrm{sym}}$) when the `CorrectCounter` criterion has claimed convergence, with that of the runs at a larger number of samples (150 in this case). We observe that if the sampling had continued further, they would have been able to better map the underlying modes. See also figure 13 for two examples of these first two failure modes for the $a = 1$ case.

3. The SVM could characterize a whole region as irrelevant due to a very deep intermediate valley even though a mode is present there. In that case, no amount of additional sampling would reveal the hidden mode. This failure mode does not occur for the $a = 0.1$ or $a = 1$ cases as the valleys are not deep enough there to be characterized as irrelevant.

As such, we would like to stress that this package was designed with a focus on uni-modal distributions and that there is no guarantee that *in general* all modes are captured or weighed correctly by the algorithm. Deeper investigations into multi-modal GP algorithms are left for future work. Note that for this distribution we used the PolyChord nested sampler [9, 10] for generating our reference contours and MC samples of the GP surrogate as it — unlike MCMC — reliably finds and explores all modes.

6.2.4 Performance for non-Gaussian and multi-modal distributions

In section 6.2 we have demonstrated that non-Gaussian distributions need a larger number of training samples in order to converge when compared to Gaussian distributions with equal dimensionality (in the particular examples presented in this section, the ratio of required samples seems to be approximately 5). This need for a larger number of posterior evaluations for convergence is also true for traditional algorithms. We could perform a similar analysis to the one presented in figure 5 to check whether the comparison with MCMC and `PolyChord` generalizes to non-Gaussian cases. Due to the wide variety of possible non-Gaussian shapes, the required number of samples and the corresponding overhead will depend dramatically on the distribution at hand. We therefore refrain from performing such an analysis at this point, and leave it for future work, where a range of more realistic non-Gaussian distributions would be tested, instead of the particularly pathological cases discussed here.

JCAP10(2023)021

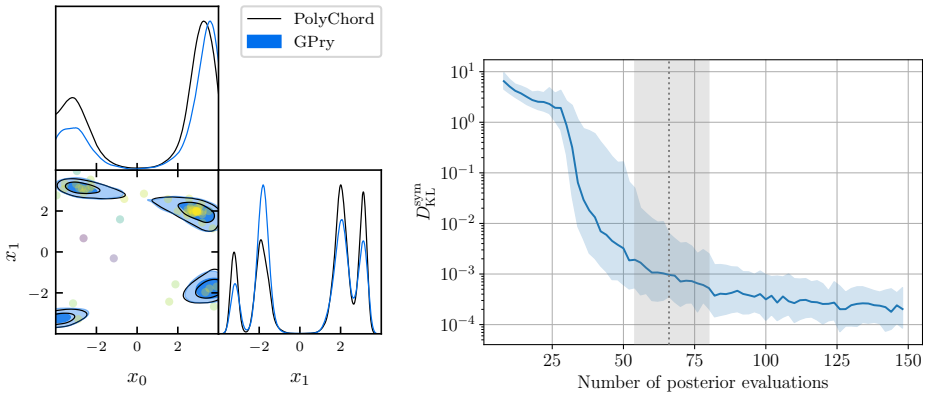


Figure 11. 2d and 1d posterior distributions of a typical, converged runs of the "mild" Himmelblau function (left) at convergence (58 posterior evaluations) and convergence against number of accepted steps (right), where the blue and grey bands are defined as in figure 6. The function has four modes which are all sampled but not weighed correctly by `GPry`. `GPry` on average needs few ($\lesssim 75$) samples to claim convergence.

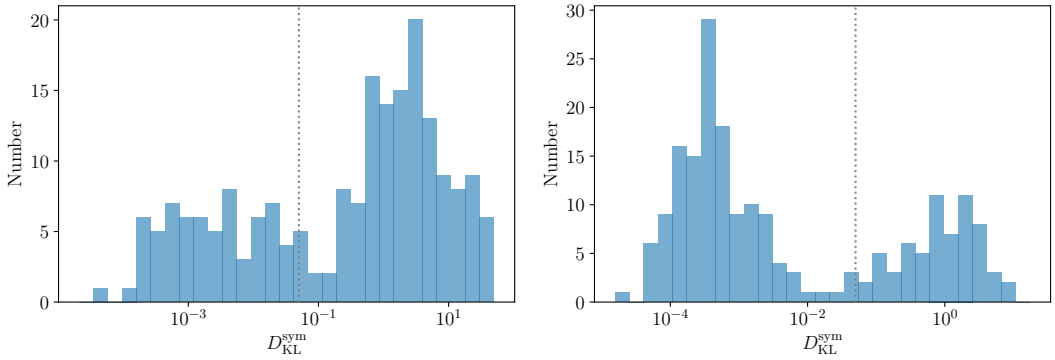


Figure 12. *Left*: distribution of KL divergences at convergence according to `CorrectCounter` of the standard Himmelblau function. In many cases convergence is declared while not all of the four modes of the function have been explored, leading to large values of $D_{\mathrm{KL}}^{\mathrm{sym}}$. *Right*: distribution of KL divergences for the same Himmelblau function at a budgeted, large number of samples (in this case 150). The distribution shows that sampling beyond reported convergence of the `CorrectCounter` criterion would aid in improving the interpolation. Nonetheless, there still remain two modes: one at low values of $D_{\mathrm{KL}}^{\mathrm{sym}}$ (around $D_{\mathrm{KL}}^{\mathrm{sym}} = 10^{-3}$) where all modes have been found and one at high values (around $D_{\mathrm{KL}}^{\mathrm{sym}} = 1$) where some of the modes were not explored. Examples of this behaviour are shown in figure 13.

6.3 Cosmology

We also test the `GPry` tool in the context of cosmological applications, such as the inference of the posterior for Planck CMB anisotropy measurements (using the nuisance-marginalised Planck Lite likelihood of [89, 90] in the context of the 6-dimensional ΛCDM model). We performed 75 separate runs of the `GPry` tool, converging on average within only around 500 evaluations of the underlying theory code.[14] The convergence history as well as the final

[14]Here we happen to be using `CLASS` [91], but since the `GPry` tool is fully interfaced with `Cobaya` [86], other theory codes can be used as well.

JCAP10(2023)021

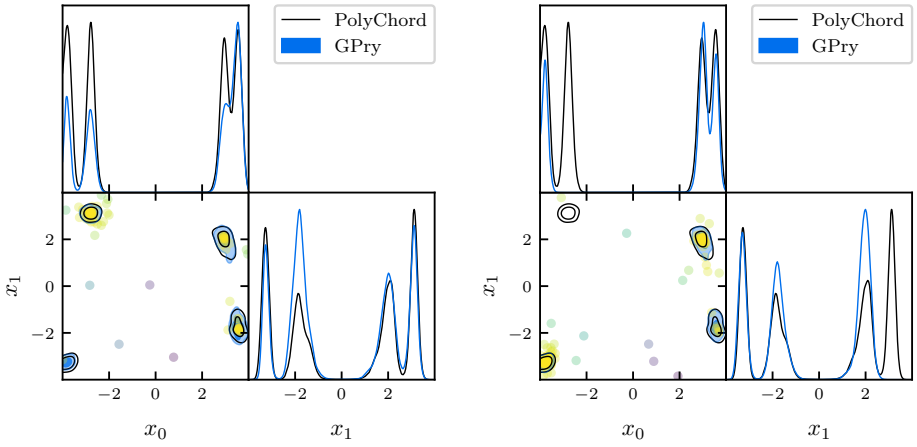


Figure 13. Exemplary 2d and 1d posterior distributions of the full Himmelblau function ($a = 1$). *Left*: contours of the algorithm finding all modes and converging at 102 posterior evaluations (although the 1D posteriors are not weighed correctly). *Right*: example of the algorithm missing a mode completely and falsely claiming convergence. This problem arises when the posterior distribution to map has several disconnected modes. If one of the modes is missed completely early in the sampling procedure the GP surrogate and hence the acquisition procedure may deem this region irrelevant and not sample there. This behaviour is especially severe when the SVM classifies the region which contains the additional mode(s) as infinite.

KL-divergence upon termination through the convergence criterion are shown in figure 14. An exemplary case (close to the median in terms of required number of samples) is also shown in figure 14, where we can see that the constraints are very well aligned with those of the true posterior.

We note that the full Planck likelihood (including nuisance parameters) can also be modeled with `GPry`, but in this case the high dimensionality of the parameter space (27 dimensions in our case) makes the proposal of new points to start the acquisition optimization from (see line 12 of algorithm 1) rather difficult. If one uses the bestfit and covariance matrix of the Planck chains to propose these points instead, the acquisition function can be well optimized and the run does correctly map the posterior. We leave investigations of reaching convergence for the full Planck likelihood without any kind of a priori information (such as covariance matrix or bestfit) for future work.

Another illustrative example is that of the combined Big Bang Nucleosynthesis (BBN) and Baryon Acoustic Oscillatons (BAO) measurements. We combine low redshift BAO from 6dFGS galaxies, the DR7 main galaxy sample, DR12 luminous red galaxies (together low-z), as well as high redshift BAO from DR16 quasars, DR16 Lyman-α based BAO, and their cross-correlations (together high-z), in order to constrain the Hubble constant and the matter composition of the Universe. The BBN likelihood we adopt is the same as in [92]. For this case we vary the number of effective neutrinos $N_{\rm eff}$, corresponding to the addition of dark radiation to the ΛCDM model. This results in three possible data likelihood combinations, depending on whether we combine with the BBN data the low redshift galaxy based BAO likelihoods ("low-z"), with the higher redshift Lyman-α and quasar BAO likelihoods ("high-z"), or we use both redshift samples ("combined"). For every combination, we sample the four-dimensional posterior using both MCMC and `GPry`. In figure 15 we show the resulting triangle plot which is in excellent agreement, demonstrating again the flexibility of GPry even

JCAP10(2023)021

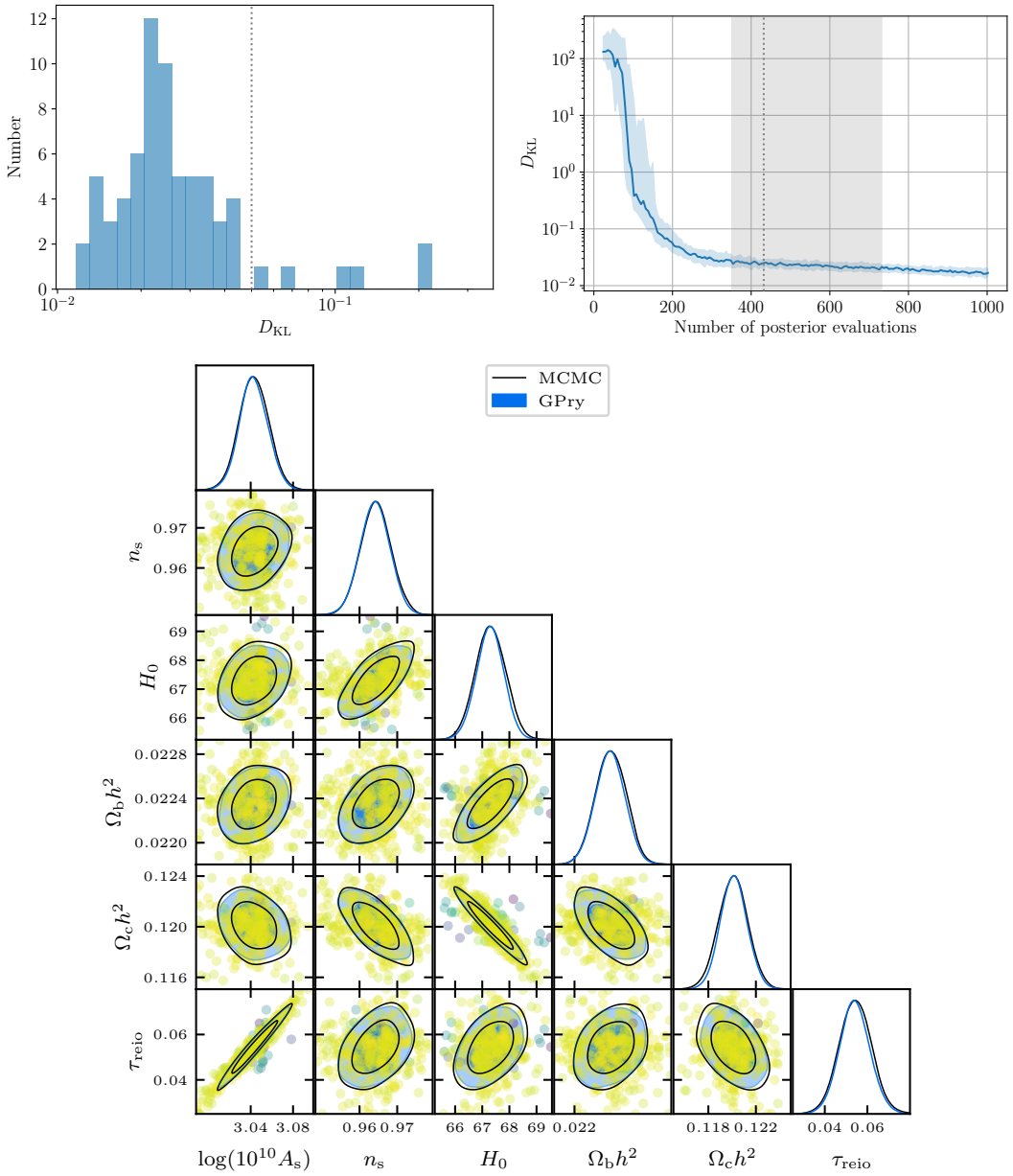


Figure 14. Constraints and convergence statistics in a ΛCDM model from Planck 2018 (TT, TE, EE, lensing) using the nuisance-marginalised Planck Lite likelihood. The given constraints could be obtained sampling only around ~ 500 (in this case 420) evaluations of the underlying theory code and likelihood. *Top Left*: distribution of KL divergences at convergence according to `CorrectCounter` with default settings. *Top Right*: KL divergences against number of accepted steps, as defined in figure 6. For both top plots, these distributions refer to 75 independent runs with identical training settings. *Bottom*: 1D posteriors and (68.3%, 95.4%) contours of the 2D posteriors for one representative run at convergence. The dots show the training samples in the order in which they were acquired, where darker samples were added early and yellow ones late.

when the underlying model or the used data sets are varied. The contours can be recovered with only around ~ 100 posterior evaluations (as opposed to the $\sim 10^4$ points used for the MCMC chains).

JCAP10(2023)021

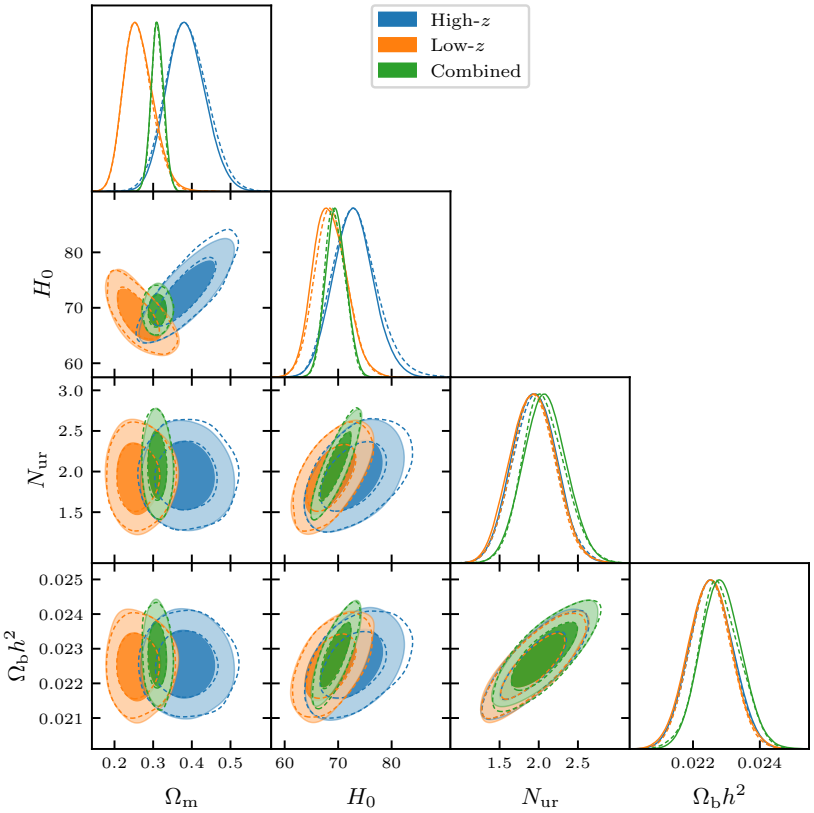


Figure 15. Triangle plot showing the marginalised constraints of the four-dimensional likelihood of BBN+BAO measurements for high-z, low-z and combined likelihoods. `GPry` is able to recover all contours correctly with only very few (124, 108, 80) posterior evaluations. The contours that we recover are in excellent agreement with the constraints from MCMC.

7 Conclusions

In this paper we presented the `GPry` algorithm and Python package implementation. As shown with both synthetic and cosmological likelihoods our algorithm requires vastly less posterior evaluations for generating a fair Monte Carlo sample for Bayesian Inference than current state-of-the-art MCMC and nested samplers. We report up to multiple orders of magnitude improvements in the number of posterior evaluations required, as well as in wall-clock computation time savings, making this algorithm very promising for slow likelihood codes. This not only speeds up inference significantly but also reduces its carbon footprint. Furthermore, we open a new window of possibilities by enabling inference from extremely slow likelihoods ($\gtrsim$ minutes per evaluation), which otherwise would be impossible to sample, since traditional samplers might take months to converge. In addition, since our algorithm does not rely on specialized hardware (such as GPUs) or any kind of pre-training, it can be used as a drop-in replacement for traditional Monte Carlo samplers. Particularly in the case of cosmological applications it benefits from an interface with `Cobaya`.

Despite the algorithm's impressive performance, there is still ample room for improvement both in terms of speed and robustness. In a future series of papers we plan to explore

JCAP10(2023)021

four main avenue: (i) the overhead of constructing the GP surrogate model and the acquisition procedure could be further minimized by using clever numerical techniques, allowing `GPry` to outcompete traditional MC samplers even for fast likelihoods. (ii) As discussed in section 6.2.3 `GPry` currently is optimized for unimodal posterior distributions; it would be desirable to increase the robustness towards strongly multi-modal posteriors by generating the starting points for the acquisition optimization in a special way, and using clustering algorithms to track different modes separately. (iii) For likelihood distributions with significant stochastic or numerical noise, it would be beneficial to automatically adapt the noise term in equation (2.3) without requiring prior knowledge. (iv) For high dimensionalities the current methods of proposing additional points for restarting hyperparameter optimization and sample acquisition are still relatively naive. Similarly, the overhead of the underlying operations performed on the GP increases strongly with the number of acquired samples. Both of these hurdles can be overcome with novel approaches, potentially unlocking even the regime of high-dimensional likelihoods for further optimization with `GPry`.

The `GPry` algorithm and python package presented in this work enables parameter inference in cosmology without high computational and environmental costs. This opens up new possibilities for Bayesian inference on costly likelihood functions which have been computationally unfeasible before. With many avenues of optimization of the code-base and algorithm still left to explore, `GPry` will only continue to improve in efficiency and accuracy.

Acknowledgments

We thank Julien Lesgourgues, Antony Lewis, Andrew Liddle and Marcos Pellejero Ibáñez for useful discussions. This project was initiated when all authors were working at the TTK institute of RWTH Aachen University. We also acknowledge the use of the JARA computing cluster of the RWTH Aachen University under project `jara0184`. N.S. acknowledges support from the Maria de Maetzu fellowship grant: CEX2019-000918-M, financiado por MCIN/AEI/10.13039/501100011033. J.E. acknowledges support by the ROMFORSK grant project no. 302640. The authors thank the referee for the many helpful comments and suggestions that helped improve the quality of this manuscript. J.T. acknowledges support from the STARS@UNIPD2021 project GWCross.

A Posterior scale in higher dimensions

When considering a problem with a larger number of dimensions, there are a few aspects of the problem that require special care. It is a well-known fact that for a 1-dimensional Gaussian the region defined by one standard deviation around the mean contains $\approx 68\%$ of the total probability mass. The generalisation to higher dimensionality is non-trivial: for multivariate Gaussians, considering distances defined in units of the covariance matrix (*Mahalanobis distance*), the region defined by a unit away from the mean contains a smaller and smaller fraction of the total probability mass as dimensionality goes up. This is, of course, nothing more than the *curse of dimensionality*, and it will be present in most of the inference problems that we target in this study.[15]

In our context of modelling a probability density function, this is reflected in the dynamic range of log-probability that needs to be carefully modelled, meaning that given some

[15] It affects distributions with significant tails, and most well-behaved distributions show tails, including those in the exponential family (which includes multivariate Gaussians).

JCAP10(2023)021

confidence limit (CL) up to which we want our model to be especially precise, the difference between the log-posterior corresponding to that CL and the maximum log-posterior will depend on dimensionality. This dynamic range will show up at three different steps of the algorithm, explicitly in the treatment of infinities and extreme values in section 3.3 and the convergence criterion in section 4.3, and implicitly in the choice of the acquisition hyperparameter in section 4.1.2. Taking into account this dimensionality scaling in the ways explained below has proven to dramatically improve the performance of our algorithm.

In order to give a rough order-of-magnitude estimate for this log-posterior scaling, we can turn towards a multivariate Gaussian distribution of the same dimensionality. Treated as a random variable itself, a multivariate Gaussian log-probability is proportional to the sum of N_d independent standard 1-dimensional Gaussian random variables (up to a linear covariance-diagonalizing transformation). Thus the value of the Gaussian log-posterior when multiplied by -2 follows a χ^2 distribution with N_d degrees of freedom. Defining $\Delta\chi^2 = 2[\max(\log p) - \log p]$ we find $\Delta\chi^2 \sim {\chi^2}_{N_d}$.

We can use this to compute the posterior range corresponding to different CLs defined by the posterior mass ϵ that they leave out, using the χ^2 cumulative distribution function F_{N_d} (where N_d is the number of degrees of freedom):

$$1 - \varepsilon = F_{N_d}(\Delta\chi^2)\,. \tag{A.1}$$

When referring to CLs in higher dimensions, we can alternatively name them as their 1D equivalent normal Gaussian extent ($1-\epsilon = 0.683$ for 1-σ, $1-\epsilon = 0.954$ for 2-σ, etc.). As such, in the following when we refer to a n–σ contour in an arbitrary dimensionality within this paper, we explicitly refer to the CL corresponding to that number of standard deviations in a $1D$ Gaussian. Explicitly, since $F_1(x) = \mathrm{erf}(\sqrt{x/2})$, and given that the value of a χ^2_1 random variable represents the squared number of standard deviations away from the mean in the corresponding Gaussian, we can simply write $1 - \varepsilon = \mathrm{erf}(n/\sqrt{2})$ for a given n-σ CL.

With this, we can get the expected scaling in N_d dimensions corresponding to a n-σ probability mass as

$$[\Delta\chi^2](n) = F^{-1}_{N_d}\left[\mathrm{erf}(n/\sqrt{2})\right]\,. \tag{A.2}$$

As an example, the 2-σ ($1 - \epsilon = 0.954$) contour corresponds to a range $[\Delta\chi^2](2) = 9.72$ in 4 dimensions and $[\Delta\chi^2](2) = 15.79$ in 8 dimensions.

In section 3.3 we have used this result to derive the threshold value T for the SVM (the criterion for deciding if a sample has a sufficient log-posterior to be added to the GP) by imposing $T = [\Delta\chi^2](n_T)/2$ which is the scaling of the log-posterior for $n_T = 20$ ($\epsilon \approx 5.5 \cdot 10^{-89}$), and has been found to work well for most practical applications (including all examples in this work). This prescription ensures dimensional consistency: choices of T as absolute values do not work well across different dimensions, causing the SVM to be too permissive in low dimensions (does not capture extreme values efficiently) and too stringent in high dimensions (points with significant log-posterior are excluded).

We have also used this result to scale the tolerance of the convergence criterion in section 4.3. In particular, since the absolute threshold ϵ_{abs} is compared against differences in absolute values of $\log p$, we are scaling it as these differences do for a fixed difference in credibility, in particular that of the first σ credible (hyper)volume: $\epsilon_{\mathrm{abs}} = 0.01[\Delta\chi^2](1)$.

Regarding the dimensionality scaling of the learning hyperparameter, it is not trivial to find an analytic prescription to write ζ as a function of $[\Delta\chi^2](n)$. As discussed in section 4.1.2 we have derived an experimental scaling $\zeta = N_d^{-0.85}$, which corresponds to the value of ζ

JCAP10(2023)021

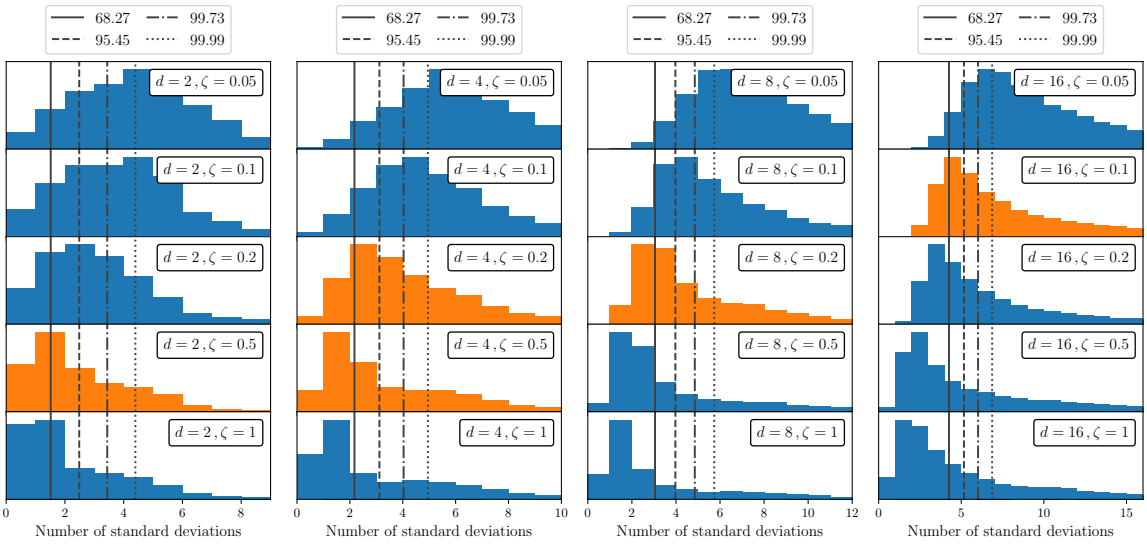


Figure 16. Histograms of aggregated Mahalanobis distances of the points in the training sets of the realizations used in figure 2, for different dimensionalities (columns) and values of ζ (rows). The optimal ζ's from the experimental relation $\zeta = N_d^{-0.85}$ are highlighted in orange/clear (for $d = 4$, the two closest values are both highlighted). A remarkable result is that efficiency at converging (the criterion imposed to get the optimal ζ's) is maximised when the distribution of training points are centered around the same CL (68%) in all dimensionalities, likely imposed indirectly by using the dimensionally-consistent KL divergence to assess convergence when selecting optimal ζ's.

that leads to convergence in the smallest number of posterior evaluations, as demonstrated in figure 2. We can check a posteriori how these optimal dimensional-dependent ζ's relate to the CL's at these dimensionalities. To do that, we compute the Mahalanobis distances of all points in the training sets of all the realizations used for figure 2, and create histograms of these distances for each dimensionality and ζ in figure 16. In this figure, we highlight the cases that converged most efficiently in orange/clear, as assessed by the dimensionally-consistent Kullback-Leibler divergence. We observe that convergence is achieved more efficiently when ζ is such that the distribution of training points is centered around the same CL (68%) in all dimensionalities. This underlines the idea that the dimensionally-dependent CL's should set the relevant scales in the surrogate model for optimal efficiency, and is possibly in fact a consequence of having used a dimensionally-consistent method (the Kullback-Leibler divergence) to assess convergence.

B KL divergence

A natural way of assessing how well a given distribution can approximate another reference distribution is the Kullback-Leibler (KL) divergence. The KL divergence of the continuous probability distribution P from the distribution Q, with respective probability density functions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, is defined as [93]

$$D_{\mathrm{KL}}(P||Q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) \mathrm{d}\boldsymbol{x}\,. \tag{B.1}$$

The KL divergence as defined above more strongly weighs disagreements between the two probability distributions where $p(\boldsymbol{x})$ is large. Since we want the approximation to be equally

JCAP10(2023)021

accurate in all regions where either distribution is large, we use a symmetrized version of the divergence (often called Jeffreys divergence). It is defined as

$$D_{\mathrm{KL}}^{\mathrm{sym}}(P,Q) = \frac{1}{2}\left(D_{\mathrm{KL}}(P||Q) + D_{\mathrm{KL}}(Q||P)\right) . \tag{B.2}$$

A smaller value means that the two posteriors are in better agreement, and one typically wants $D_{\mathrm{KL}}^{\mathrm{sym}}(P||Q) \ll 1$ for good agreement. The dimensionality consistency of the KL divergence guarantees that a given value for the divergence characterizes similar differences across dimensionalities.

To compute the KL divergence explicitly, one can use the fact that the points in a Monte Carlo sample of P are distributed as $p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. One can thus approximate the integral as a sum of the quantity $\log p(\boldsymbol{x}_i) - \log q(\boldsymbol{x}_i)$ over all points in the MC sample (multiplied by their respective weights/multiplicities).

We can use the KL divergence to assess the convergence towards the true distribution of a GP surrogate model, if a sample from the true distribution can be obtained with the usual MC methods (e.g. in the test cases presented in section 6). In that case, $\log p(\boldsymbol{x}_i)$ would be the true log-posterior at point i in the MC sample, and $\log q(\boldsymbol{x}_i)$ would be the emulated log-posterior from `GPry` at that same point. In practical applications where an MC sample of the true posterior is not possible to obtain, the KL divergence can be used in a similar fashion to define a convergence criterion by comparing GP surrogate models at consecutive iterations of the `GPry` algorithm, summing over an MC sample of the GP surrogate model at a particular step (see section 4.3).

In order to save a significant amount of memory, when using the KL divergence for the purpose of a convergence criterion, instead of integrating a full MCMC, we only store the information from the mean and the covariance matrix. This is equivalent to approximating the underlying distributions as multivariate Gaussian distributions (with mean $\mathfrak{m}$ and covariance $\boldsymbol{C}$). While this is a bad description for the distribution itself, it is often the case that when the multivariate Gaussian approximation of a distribution agrees to a high level of precision with that of another distribution, so do the underlying distributions. Under this approximation the KL divergence is simply given by

$$D_{\mathrm{KL}}(P||Q) \approx \frac{1}{2}\left(\mathrm{tr}\left(\boldsymbol{C}_Q^{-1}\boldsymbol{C}_P\right) - d + (\mathfrak{m}_Q - \mathfrak{m}_P)^T\boldsymbol{C}_Q^{-1}(\mathfrak{m}_Q - \mathfrak{m}_P) + \log\left(\frac{\det\boldsymbol{C}_Q}{\det\boldsymbol{C}_P}\right)\right) . \tag{B.3}$$

Whether using the MC-summed or the Gaussian approximation for the KL divergence, using it to naively define a convergence criterion, can be problematic, since running a full Monte Carlo sample at every acquired point, or at every iteration, would dominate the overhead of the algorithm. To reduce this computational cost, we take a number of decisions: before deciding whether to re-run the Monte Carlo sample, we reweigh the previous one and compute the KL divergence between it and the previous estimate. We then re-use the reweighed one if the KL divergence between original and reweighed is small enough. We also relax the convergence criterion of the Monte Carlo algorithm early in the sampling procedure as convergence there is rather unlikely, so we do not need a high-quality estimation of the mean and covariance at that point.

Convergence is then determined by defining a threshold c value such that the algorithm stops when $D_{\mathrm{KL}}^{\mathrm{sym}} < c$ during n iterations, suggesting that the interpolation of the posterior distribution has stabilised. We set $n = 2$ as the default.

JCAP10(2023)021

Even with these improvements this method still produces considerable computational overhead, mainly due to the fact that running a Monte Carlo chain needs a large number of samples from the GP, especially as the number of dimensions increases.

References

[1] A. Lewis and S. Bridle, *Cosmological parameters from CMB and other data: A Monte Carlo approach*, *Phys. Rev. D* **66** (2002) 103511 [astro-ph/0205436] [INSPIRE].

[2] A. Lewis, *Efficient sampling of fast and slow cosmological parameters*, *Phys. Rev. D* **87** (2013) 103529 [arXiv:1304.4473] [INSPIRE].

[3] D. Foreman-Mackey, D.W. Hogg, D. Lang and J. Goodman, *emcee: The MCMC Hammer*, *Publ. Astron. Soc. Pac.* **125** (2013) 306 [arXiv:1202.3665] [INSPIRE].

[4] J. Akeret, S. Seehars, A. Amara, A. Refregier and A. Csillaghy, *CosmoHammer: Cosmological parameter estimation with the MCMC Hammer*, *Astron. Comput.* **2** (2013) 27.

[5] J. Skilling, *Nested Sampling*, *AIP Conf. Proc.* **735** (2004) 395.

[6] F. Feroz and M.P. Hobson, *Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis*, *Mon. Not. Roy. Astron. Soc.* **384** (2008) 449 [arXiv:0704.3704] [INSPIRE].

[7] F. Feroz, M.P. Hobson and M. Bridges, *MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics*, *Mon. Not. Roy. Astron. Soc.* **398** (2009) 1601 [arXiv:0809.3437] [INSPIRE].

[8] F. Feroz, M.P. Hobson, E. Cameron and A.N. Pettitt, *Importance Nested Sampling and the MultiNest Algorithm*, *Open J. Astrophys.* **2** (2019) 10 [arXiv:1306.2144] [INSPIRE].

[9] W.J. Handley, M.P. Hobson and A.N. Lasenby, *PolyChord: nested sampling for cosmology*, *Mon. Not. Roy. Astron. Soc.* **450** (2015) L61 [arXiv:1502.01856] [INSPIRE].

[10] W.J. Handley, M.P. Hobson and A.N. Lasenby, *polychord: next-generation nested sampling*, *Mon. Not. Roy. Astron. Soc.* **453** (2015) 4385 [arXiv:1506.00171] [INSPIRE].

[11] E. Higson, W. Handley, M. Hobson and A. Lasenby, *Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation*, *Stat. Comput.* **29** (2018) 891.

[12] J.S. Speagle, *dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences*, *Mon. Not. Roy. Astron. Soc.* **493** (2020) 3132 [arXiv:1904.02180] [INSPIRE].

[13] E.D. Feigelson, R.S. de Souza, E.E.O. Ishida and G.J. Babu, *21st Century Statistical and Computational Challenges in Astrophysics*, *Ann. Rev. Stat. App.* **8** (2021) 493 [arXiv:2005.13025] [INSPIRE].

[14] R. Alves Batista et al., *EuCAPT White Paper: Opportunities and Challenges for Theoretical Astroparticle Physics in the Next Decade*, arXiv:2110.10074 [INSPIRE].

[15] A.R.H. Stevens, S. Bellstedt, P.J. Elahi and M.T. Murphy, *The imperative to reduce carbon emissions in astronomy*, *Nature Astron.* **4** (2020) 843 [arXiv:1912.05834] [INSPIRE].

[16] S. Portegies Zwart, *The Ecological Impact of High-performance Computing in Astrophysics*, *Nature Astron.* **4** (2020) 819 [arXiv:2009.11295] [INSPIRE].

[17] M. Kaplinghat, L. Knox and C. Skordis, *Rapid calculation of theoretical CMB angular power spectra*, *Astrophys. J.* **578** (2002) 665 [astro-ph/0203413] [INSPIRE].

[18] R. Jimenez, L. Verde, H. Peiris and A. Kosowsky, *Fast cosmological parameter estimation from microwave background temperature and polarization power spectra*, *Phys. Rev. D* **70** (2004) 023005 [astro-ph/0404237] [INSPIRE].

[19] T. Auld, M. Bridges, M.P. Hobson and S.F. Gull, *Fast cosmological parameter estimation using neural networks*, *Mon. Not. Roy. Astron. Soc.* **376** (2007) L11 [astro-ph/0608174] [INSPIRE].

JCAP10(2023)021

[20] T. Auld, M. Bridges and M.P. Hobson, *CosmoNet: Fast cosmological parameter estimation in non-flat models using neural networks*, *Mon. Not. Roy. Astron. Soc.* **387** (2008) 1575 [astro-ph/0703445] [INSPIRE].

[21] J. Albers, C. Fidler, J. Lesgourgues, N. Schöneberg and J. Torrado, *CosmicNet. Part I. Physics-driven implementation of neural networks within Einstein-Boltzmann Solvers*, *JCAP* **09** (2019) 028 [arXiv:1907.05764] [INSPIRE].

[22] A. Manrique-Yus and E. Sellentin, *Euclid-era cosmology for everyone: neural net assisted MCMC sampling for the joint* 3×2 *likelihood*, *Mon. Not. Roy. Astron. Soc.* **491** (2020) 2655 [arXiv:1907.05881] [INSPIRE].

[23] A. Mootoovaloo, A.F. Heavens, A.H. Jaffe and F. Leclercq, *Parameter Inference for Weak Lensing using Gaussian Processes and MOPED*, *Mon. Not. Roy. Astron. Soc.* **497** (2020) 2213 [arXiv:2005.06551] [INSPIRE].

[24] A. Nygaard, E.B. Holm, S. Hannestad and T. Tram, *CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference*, *JCAP* **05** (2023) 025 [arXiv:2205.15726] [INSPIRE].

[25] J. Donald-McCann, F. Beutler, K. Koyama and M. Karamanis, *matryoshka: halo model emulator for the galaxy power spectrum*, *Mon. Not. Roy. Astron. Soc.* **511** (2022) 3768 [arXiv:2109.15236] [INSPIRE].

[26] J. Donald-McCann, K. Koyama and F. Beutler, *matryoshka II: accelerating effective field theory analyses of the galaxy power spectrum*, *Mon. Not. Roy. Astron. Soc.* **518** (2022) 3106 [arXiv:2202.07557] [INSPIRE].

[27] M. Bonici, L. Biggio, C. Carbone and L. Guzzo, *Fast emulation of two-point angular statistics for photometric galaxy surveys*, arXiv:2206.14208 [INSPIRE].

[28] A. Mootoovaloo, A.H. Jaffe, A.F. Heavens and F. Leclercq, *Kernel-based emulator for the* 3*D matter power spectrum from CLASS*, *Astron. Comput.* **38** (2022) 100508 [arXiv:2105.02256] [INSPIRE].

[29] S. Günther et al., *CosmicNet II: emulating extended cosmologies with efficient and accurate neural networks*, *JCAP* **11** (2022) 035 [arXiv:2207.05707] [INSPIRE].

[30] A. Spurio-Mancini, D. Piras, J. Alsing, B. Joachimi and M.P. Hobson, *CosmoPower: emulating cosmological power spectra for accelerated Bayesian inference from next-generation surveys*, *Mon. Not. Roy. Astron. Soc.* **511** (2022) 1771 [arXiv:2106.03846] [INSPIRE].

[31] C.-H. To, E. Rozo, E. Krause, H.-Y. Wu, R.H. Wechsler and A.N. Salcedo, *LINNA: Likelihood Inference Neural Network Accelerator*, *JCAP* **01** (2023) 016 [arXiv:2203.05583] [INSPIRE].

[32] S. Khan and R. Green, *Gravitational-wave surrogate models powered by artificial neural networks*, *Phys. Rev. D* **103** (2021) 064015 [arXiv:2008.12932] [INSPIRE].

[33] M. Chianese, A. Coogan, P. Hofma, S. Otten and C. Weniger, *Differentiable Strong Lensing: Uniting Gravity and Neural Nets through Differentiable Probabilistic Programming*, *Mon. Not. Roy. Astron. Soc.* **496** (2020) 381 [arXiv:1910.06157] [INSPIRE].

[34] F. Lanusse, R. Mandelbaum, S. Ravanbakhsh, C.-L. Li, P. Freeman and B. Póczos, *Deep generative models for galaxy image simulations*, *Mon. Not. Roy. Astron. Soc.* **504** (2021) 5543.

[35] K.K. Rogers, H.V. Peiris, A. Pontzen, S. Bird, L. Verde and A. Font-Ribera, *Bayesian emulator optimisation for cosmology: application to the Lyman-*α *forest*, *JCAP* **02** (2019) 031 [arXiv:1812.04631] [INSPIRE].

[36] T. McClintock et al., *The Aemulus Project. Part II. Emulating the Halo Mass Function*, *Astrophys. J.* **872** (2019) 53 [arXiv:1804.05866] [INSPIRE].

[37] M.-F. Ho, S. Bird and C.R. Shelton, *Multifidelity emulation for the matter power spectrum using Gaussian processes*, *Mon. Not. Roy. Astron. Soc.* **509** (2021) 2551 [arXiv:2105.01081] [INSPIRE].

JCAP10(2023)021

[38] C.J. Moore and J.R. Gair, *Novel Method for Incorporating Model Uncertainties into Gravitational Wave Parameter Estimates*, *Phys. Rev. Lett.* **113** (2014) 251101 [arXiv:1412.3657] [INSPIRE].

[39] C. Chen, Y. Li, F. Villaescusa-Navarro, S. Ho and A. Pullen, *Learning the Evolution of the Universe in N-body Simulations*, in proceedings of the *34th Conference on Neural Information Processing Systems*, online conference, Canada, 6–12 December 2020, arXiv:2012.05472 [INSPIRE].

[40] S. Bird, K.K. Rogers, H.V. Peiris, L. Verde, A. Font-Ribera and A. Pontzen, *An Emulator for the Lyman-α Forest*, *JCAP* **02** (2019) 050 [arXiv:1812.04654] [INSPIRE].

[41] K. Cranmer, J. Brehmer and G. Louppe, *The frontier of simulation-based inference*, *Proc. Nat. Acad. Sci.* **117** (2020) 30055 [arXiv:1911.01429] [INSPIRE].

[42] J.-M. Lueckmann, J. Boelts, D.S. Greenberg, P.J. Gonçalves and J.H. Macke, *Benchmarking Simulation-Based Inference*, arXiv:2101.04653.

[43] A. Delaunoy et al., *Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization*, arXiv:2010.12931 [INSPIRE].

[44] J. Alsing, T. Charnock, S. Feeney and B. Wandelt, *Fast likelihood-free cosmology with neural density estimators and active learning*, *Mon. Not. Roy. Astron. Soc.* **488** (2019) 4440 [arXiv:1903.00007] [INSPIRE].

[45] B.K. Miller, A. Cole, G. Louppe and C. Weniger, *Simulation-efficient marginal posterior estimation with swyft: stop wasting your precious time*, arXiv:2011.13951 [INSPIRE].

[46] J. Hermans, N. Banik, C. Weniger, G. Bertone and G. Louppe, *Towards constraining warm dark matter with stellar streams through neural simulation-based inference*, *Mon. Not. Roy. Astron. Soc.* **507** (2021) 1999 [arXiv:2011.14923] [INSPIRE].

[47] F. Gerardi, S.M. Feeney and J. Alsing, *Unbiased likelihood-free inference of the Hubble constant from light standard sirens*, *Phys. Rev. D* **104** (2021) 083531 [arXiv:2104.02728] [INSPIRE].

[48] D. Huppenkothen and M. Bachetti, *Accurate X-ray timing in the presence of systematic biases with simulation-based inference*, *Mon. Not. Roy. Astron. Soc.* **511** (2022) 5689 [arXiv:2104.03278] [INSPIRE].

[49] A. Rouhiainen, U. Giri and M. Münchmeyer, *Normalizing flows for random fields in cosmology*, arXiv:2105.12024 [INSPIRE].

[50] K. Zhang, J.S. Bloom, B.S. Gaudi, F. Lanusse, C. Lam and J.R. Lu, *Real-time Likelihood-free Inference of Roman Binary Microlensing Events with Amortized Neural Posterior Estimation*, *Astron. J.* **161** (2021) 262.

[51] C.-H. Hahn et al., *SIMBIG: A Forward Modeling Approach To Analyzing Galaxy Clustering*, arXiv:2211.00723 [INSPIRE].

[52] M. Reza, Y. Zhang, B. Nord, J. Poh, A. Ciprijanovic and L. Strigari, *Estimating Cosmological Constraints from Galaxy Cluster Abundance using Simulation-Based Inference*, in proceedings of the *39th International Conference on Machine Learning Conference*, Baltimore, MD, U.S.A., 17–23 July 2022, arXiv:2208.00134 [INSPIRE].

[53] S.S. Boruah, T. Eifler, V. Miranda and P.M. Sai Krishanth, *Accelerating cosmological inference with Gaussian processes and neural networks — an application to LSST Y1 weak lensing and galaxy clustering*, *Mon. Not. Roy. Astron. Soc.* **518** (2022) 4818 [arXiv:2203.06124] [INSPIRE].

[54] K.H. Scheutwinkel, W. Handley and E. de Lera Acedo, *Bayesian evidence-driven likelihood selection for sky-averaged 21 cm signal extraction*, *Publ. Astron. Soc. Austral.* **40** (2023) e016 [arXiv:2204.04491] [INSPIRE].

[55] D. Grandón and E. Sellentin, *Bayesian error propagation for neural-net based parameter inference*, *Open J. Astrophys.* **5** (2022) 12 [arXiv:2205.11587] [INSPIRE].

JCAP10(2023)021

[56] P. Lemos et al., *Robust simulation-based inference in cosmology with Bayesian neural networks*, *Mach. Learn. Sci. Tech.* **4** (2023) 01LT01 [arXiv:2207.08435] [INSPIRE].

[57] C.E. Rasmussen and C.K.I. Williams, *Gaussian processes for machine learning*, in *Adaptive Computation and Machine Learning*, MIT Press, Cambridge, MA, U.S.A. (2006).

[58] K.P. Murphy, *Machine Learning — A Probabilistic Perspective*, MIT Press, Cambridge, MA, U.S.A. (2012).

[59] C. Cortes and V. Vapnik, *Support-vector networks*, *Mach. Learn.* **20** (1995) 273.

[60] T. Gunter, M. Osborne, R. Garnett, P. Hennig and S. Roberts, *Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature*, in proceedings of the *27th International Conference on Neural Information Processing Systems (NIPS'14)*, Montréal, QC, Canada, 8–13 December 2014, Curran Associates, Inc. (2014), pp. 2789–2797 http://papers.nips.cc/paper/5483-sampling-for-inference-in-probabilistic-models-with-fast-bayesian-quadrature.pdf.

[61] M. Osborne, R. Garnett, Z. Ghahramani, D.K. Duvenaud, S.J. Roberts and C. Rasmussen, *Active Learning of Model Evidence Using Bayesian Quadrature*, in proceedings of the *25th International Conference on Neural Information Processing Systems (NIPS'12)*, Lake Tahoe, Nevada, U.S.A., 3–6 December 2012, *Advances in Neural Information Processing Systems* **25**, F. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger eds., Curran Associates, Inc. (2012), pp. 46–54 https://proceedings.neurips.cc/paper/2012/file/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Paper.pdf.

[62] K. Kandasamy, J. Schneider and B. Póczos, *Bayesian active learning for posterior estimation*, in proceedings of the *24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 25–31 July 2015, pp. 3605–3611.

[63] H. Wang and J. Li, *Adaptive Gaussian Process Approximation for Bayesian Inference with Expensive Likelihood Functions*, *Neural Comput.* **30** (2018) 3072 [arXiv:1703.09930].

[64] H. Chai and R. Garnett, *Improving Quadrature for Constrained Integrands*, arXiv:1802.04782.

[65] M. Pellejero-Ibañez, R.E. Angulo, G. Aricó, M. Zennaro, S. Contreras and J. Stücker, *Cosmological parameter estimation via iterative emulation of likelihoods*, *Mon. Not. Roy. Astron. Soc.* **499** (2020) 5257 [arXiv:1912.08806] [INSPIRE].

[66] K.K. Rogers, H. Peiris, A. Pontzen, S. Bird, L. Verde and A. Font-Ribera, *Bayesian emulator optimisation for cosmology: application to the Lyman-α forest*, *JCAP* **02** (2019) 031 [arXiv:1812.04631] [INSPIRE].

[67] L. Acerbi, *Variational Bayesian Monte Carlo*, in proceedings of the *32nd International Conference on Neural Information Processing Systems*, Montréal, QC, Canada, 3–8 December 2018, *Advances in Neural Information Processing Systems* **31**, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett eds., Curran Associates, Inc. (2018), pp. 8223–8233 https://proceedings.neurips.cc/paper_files/paper/2018/file/747c1bcceb6109a4ef936bc70cfe67de-Paper.pdf.

[68] L. Acerbi, *Variational Bayesian Monte Carlo with Noisy Likelihoods*, in proceedings of the *34th International Conference on Neural Information Processing Systems (NIPS'20)*, Vancouver, BC, Canada, 6–12 December 2020, *Advances in Neural Information Processing Systems* **33**, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin eds., Curran Associates, Inc. (2020), pp. 8211–8222 [arXiv:2006.08655].

[69] B. Huggins, C. Li, M. Tobaben, M.J. Aarnos and L. Acerbi, *PyVBMC: Efficient Bayesian inference in Python*, arXiv:2303.09519 [DOI:10.21105/joss.05428].

[70] C. Li, G. Clarté and L. Acerbi, *Fast post-process Bayesian inference with Sparse Variational Bayesian Monte Carlo*, arXiv:2303.05263.

[71] A. Saha, K. Bharath and S. Kurtek, *A Geometric Variational Approach to Bayesian Inference*, arXiv:1707.09714.

JCAP10(2023)021

[72] P. Frank, R. Leike and T.A. Enßlin, *Geometric variational inference*, arXiv:2105.10470 [DOI:10.3390/e23070853].

[73] C.A. Micchelli, Y. Xu and H. Zhang, *Universal Kernels*, *J. Mach. Learn. Res.* **7** (2006) 2651.

[74] M. Kupperman, *Probabilities of Hypotheses and Information-Statistics in Sampling from Exponential-Class Populations*, *Ann. Math. Stat.* **29** (1958) 571.

[75] C. Zhu, R.H. Byrd, P. Lu and J. Nocedal, *Algorithm 778: L-BFGS-B*, *ACM Trans. Math. Software* **23** (1997) 550.

[76] Y. Sui, V. Zhuang, J. Burdick and Y. Yue, *Stagewise Safe Bayesian Optimization with Gaussian Processes*, in proceedings of the *35th International Conference on Machine Learning*, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018, *Proceedings of Machine Learning Research* **80**, J. Dy and A. Krause eds., PMLR (2018), pp. 4781–4789 http://proceedings.mlr.press/v80/sui18a.html.

[77] F. Berkenkamp, A. Krause and A.P. Schoellig, *Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics*, arXiv:1602.04450.

[78] L. Acerbi, *An Exploration of Acquisition and Mean Functions in Variational Bayesian Monte Carlo*, in proceedings of the *1st Symposium on Advances in Approximate Bayesian Inference*, Montréal, QC, Canada, 2 December 2018, F. Ruiz, C. Zhang, D. Liang and T. Bui eds., *Proceedings of Machine Learning Research* **96**, PMLR (2019), pp. 1–1 https://proceedings.mlr.press/v96/acerbi19a.html.

[79] F.L. Fernández, L. Martino, V. Elvira, D. Delgado and J. López-Santiago, *Adaptive Quadrature Schemes for Bayesian Inference via Active Learning*, *IEEE Access* **8** (2020) 208462.

[80] K. Kandasamy, J. Schneider and B. Póczos, *Bayesian active learning for posterior estimation*, in proceedings of the *24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 25–31 July 2015, pp. 3605–3611.

[81] T. Desautels, A. Krause and J.W. Burdick, *Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization*, *J. Mach. Learn. Res.* **15** (2014) 4053.

[82] C. Chevalier and D. Ginsbourger, *Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection*, in *Learning and Intelligent Optimization*, proceedings of the *7th International Conference (LION 7)*, Catania, Italy, 7–11 January 2013, *Lecture Notes in Computer Science* **7997**, Springer (2013), pp. 59–69 [DOI:10.1007/978-3-642-44973-4_7].

[83] D. Ginsbourger, R. Le Riche and L. Carraro, *A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes*, hal-00260579 (2008).

[84] D. Ginsbourger, R.L. Riche and L. Carraro, *Kriging Is Well-Suited to Parallelize Optimization*, in *Computational Intelligence in Expensive Optimization Problems*, Springer (2010), pp. 131–162 [DOI:10.1007/978-3-642-10701-6_6].

[85] J. González, Z. Dai, P. Hennig and N. Lawrence, *Batch Bayesian Optimization via Local Penalization*, in proceedings of the *19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Cadiz, Spain, 7–11 May 2016, *Proceedings of Machine Learning Research* **51**, PMLR (2016), pp. 648–657 https://proceedings.mlr.press/v51/gonzalez16a.html.

[86] J. Torrado and A. Lewis, *Cobaya: Code for Bayesian Analysis of hierarchical physical models*, *JCAP* **05** (2021) 057 [arXiv:2005.05290] [INSPIRE].

[87] J. Torrado, N. Schöneberg and J.E. Gammal, *Parallelized Acquisition for Active Learning using Monte Carlo Sampling*, arXiv:2305.19267 [INSPIRE].

[88] E. Cameron and A. Pettitt, *Recursive Pathways to Marginal Likelihood Estimation with Prior-Sensitivity Analysis*, *Statist. Sci.* **29** (2014) 397.

[89] PLANCK collaboration, *Planck 2018 results. Part V. CMB power spectra and likelihoods*, *Astron. Astrophys.* **641** (2020) A5 [arXiv:1907.12875] [INSPIRE].

JCAP10(2023)021

[90] Planck collaboration, *Planck 2018 results. Part VIII. Gravitational lensing*, *Astron. Astrophys.* **641** (2020) A8 [arXiv:1807.06210] [inSPIRE].

[91] D. Blas, J. Lesgourgues and T. Tram, *The Cosmic Linear Anisotropy Solving System (CLASS). Part II. Approximation schemes*, *JCAP* **07** (2011) 034 [arXiv:1104.2933] [inSPIRE].

[92] N. Schöneberg, J. Lesgourgues and D.C. Hooper, *The BAO+BBN take on the Hubble tension*, *JCAP* **10** (2019) 029 [arXiv:1907.11594] [inSPIRE].

[93] S. Kullback and R.A. Leibler, *On Information and Sufficiency*, *Ann. Math. Stat.* **22** (1951) 79 [inSPIRE].

JCAP10(2023)021