

---

# Parallelized Acquisition for Active Learning using Monte Carlo Sampling

---

**Jesús Torrado**

Dipartimento di Fisica e Astronomia “G. Galilei”  
Università degli Studi di Padova  
Via Marzolo 8, I-35131 Padova, Italy  
jesus.torrado@pd.infn.it

**Nils Schöneberg**

Institut de Ciències del Cosmos  
Universitat de Barcelona  
Martí i Franquès 1, Barcelona E08028, Spain  
nils.science@gmail.com

**Jonas El Gammal**

Department of Mathematics and Physics  
University of Stavanger  
NO-4036 Stavanger, Norway  
jonas.e.elgammal@uis.no

## Abstract

Bayesian inference remains one of the most important tool-kits for any scientist, but increasingly expensive likelihood functions are required for ever-more complex experiments, raising the cost of generating a Monte Carlo sample of the posterior. Recent attention has been directed towards the use of emulators of the posterior based on Gaussian Process (GP) regression combined with active sampling to achieve comparable precision with far fewer costly likelihood evaluations. Key to this approach is the batched acquisition of proposals, so that the true posterior can be evaluated in parallel. This is usually achieved via sequential maximisation of the highly multimodal acquisition function. Unfortunately, this approach parallelizes poorly and is prone to getting stuck in local maxima. Our approach addresses this issue by generating nearly-optimal batches of candidates using an almost-embarrassingly parallel Nested Sampler on the mean prediction of the GP. The resulting nearly-sorted Monte Carlo sample is used to generate a batch of candidates ranked according to their sequentially conditioned acquisition function values at little cost. The final sample can also be used for inferring marginal quantities. Our proposed implementation (NORA) demonstrates comparable accuracy to sequential conditioned acquisition optimization and efficient parallelization in various synthetic and cosmological inference problems.

## 1 Introduction

One of the fundamental tools of science is the comparison of observations with theory. In many Bayesian inference pipelines, this involves inferring the parameters of a model (or models themselves) given some observed or generated data. This is often realised directly using Bayes theorem: Given some model parameters  $\mathbf{x} \in \mathbb{R}^d$  and data  $\mathcal{D}$ , the conditioned probability  $p(\mathbf{x}|\mathcal{D})$  (the so-called *posterior*) is given by

$$p(\mathbf{x}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{x})p(\mathbf{x})}{p(\mathcal{D})}. \quad (1)$$

where  $p(\mathcal{D}|\mathbf{x}) \equiv L(\mathbf{x})$  is called the *likelihood*,  $p(\mathbf{x}) \equiv \pi(\mathbf{x})$  the *prior*, and  $p(\mathcal{D}) \equiv E$  the *evidence*. We are dropping the explicit dependence on  $\mathcal{D}$  as it is fixed for a given inference problem. Traditionally

the posterior distribution is sampled with Monte Carlo (MC) samplers such as Markov Chain Monte Carlo (MCMC) or Nested Sampling (NS). Unfortunately though,  $L(\boldsymbol{x})$  is often an expensive to evaluate black box function, either because calculating observables from the theoretical model involves expensive computations, because the amount of data is large, or both. This makes sampling in such circumstances unfeasible with MC samplers, since they typically require  $\mathcal{O}(10^3 - 10^6)$  posterior evaluations for dimensionalities up to  $\mathcal{O}(10)$ .

There exist multiple approaches to accelerating such inference problems using machine-learning: enhanced pre-conditioning to accelerate traditional MC methods [1–5], simulation-based, implicit-likelihood inference algorithms [6–12], and emulators of underlying physical quantities (for Cosmological applications, see [13–18]). In this work we are going to focus on emulating the likelihood as a function of its parameters, using a Gaussian Process (for previous approaches see [19–25] using GP, and [26–28] enhancing the GP with a variational approximation). Any such emulation (such as that based on a GP) will typically require a set of samples of the function at various parameter points. In order to maximize the amount of information about the behavior of the function captured by these samples, often times an active sampling approach is used: New samples are proposed, based on the current best emulation, where the greatest probability of the estimated improvement of the future emulation is located [29]. This is typically measured by an acquisition function. In this work we will tackle the question of how the active sampling algorithm can be performed in a highly parallel fashion while producing optimal or near-optimal batches of new proposed sampling locations.

In order to acquire a nearly optimal batch of proposed sampling locations for the active learning algorithm in a highly parallel fashion, naive maximization of the acquisition function is not sufficient. This is not only due to the multi-modal nature of a typical acquisition function – making it easy for the optimizer to get stuck in local maxima, especially in moderately high dimensionality – but also due to the inherent lack of parallelization of standard optimization routines. This is caused by the sequential nature of the maximization algorithm, and more importantly requiring the result of a given maximization in order to compute the conditional acquisition function, which will be used for a subsequent maximization.

Our implementation combines the solution to both of these problems in an efficient way: First, by making use of a MC sampling algorithm it is possible to acquire samples of growing function value in a parallel fashion that is much more likely to find the global maximum. Second, through the usage of a ranked pool (see Section 3) we are also able to create a batch of multiple proposed sampling locations simultaneously. Both of these solutions combine to give us a highly efficient algorithm to acquire multiple near-optimal active learning sampling positions.

In Section 3 we describe the general methodology employed in our algorithm. In Section 4 we show the scaling with MPI processes as well as the acquisition histories for a number of toy examples, and we conclude in Section 5. We also show further examples in the context of cosmological inference in appendix E.

## 2 Theoretical background

### 2.1 Gaussian Processes

In this section we briefly summarize the main notation and theory. For a review, see [30]. A Gaussian process (GP) is based on a probabilistic model of a function value at any point  $x$ , which follows a conditioned Gaussian with a mean  $\mu$  and a standard deviation  $\sigma$ . Any two sampling locations  $x$  and  $x'$  are correlated in a multivariate Gaussian way with a correlation function given by  $k(x, x')$ , the kernel. The choice of the kernel and its hyperparameters encodes assumptions about the behavior of the underlying function (such as differentiability) into the GP. Given a set of sampling locations  $X_1, \dots, X_N$  and corresponding function values  $y_1, \dots, y_N$  the conditioned mean of the GP gives an emulation of the underlying function, while its conditional standard deviation describes the uncertainty in the emulation. A common choice for the kernel function and the one that we will use throughout this paper is the radial basis function (RBF) kernel given in one dimension by

$$k(x, x') = C \cdot \exp\left(\frac{(x - x')^2}{2l^2}\right) \quad (2)$$

where we will call  $C$  the *output scale* and  $l$  the length scale of the kernel. In multiple dimensions ( $\boldsymbol{x} \in \mathbb{R}^d$ ) we construct the kernel function as a product of RBF kernels, each acting on one dimension

and have different length scales  $l_i$ :

$$k(\mathbf{x}, \mathbf{x}') = C \cdot \exp \left( \sum_{i=1}^d \frac{(x_i - x'_i)^2}{2l_i^2} \right) \quad (3)$$

By optimizing the marginal log-likelihood of the hyperparameters  $\theta = \{C, l\}$  one can fit the GP to a set of sampled points. A good choice of such samples, such as through an acquisition procedure for obtaining new locations is fundamental to the final performance of the GP emulation.

## 2.2 Acquisition procedure

The second part, the acquisition of samples with which to train the GP relies on maximizing a so called *acquisition function* which, given the current GP, is a measure of the assumed information gained by sampling at any given location. We will denote the already sampled point as the training set in this context. As we want more precision towards the top of the mode for the final inference steps we encode this by choosing the acquisition function

$$a(\mu, \sigma | \mathbf{x}) = 2\zeta(\mu(\mathbf{x}) - p_{\max}) + \log(\sigma(\mathbf{x})) . \quad (4)$$

where  $\zeta$  is an empirically determined dimensional regularization factor,<sup>1</sup> and  $p_{\max}$  is the current maximum log-posterior value of the training set. We introduce this current maximum, as in most realistic cases the posterior distribution is not necessarily normalized and hence the scale of the peak not known.

In order to make use of the massive parallelization allowed for by current scientific computing systems, we require not a single optimal point, but a set of simultaneously-optimal sampling locations (batch acquisition). This would in principle require maximizing a joint acquisition function (as a function of multiple locations), which is a high-dimensional multi-modal problem. However, this can be approximated in a simpler way by sequentially acquiring a batch of points, each conditioned to the previous ones using the Kriging believer method [31–36]. In that method one optimizes the acquisition function, conditions the GP on the emulated mean  $\mu$  at the previous maximum (which is a comparatively cheap operation)<sup>2</sup> and recomputes the acquisition function using this conditioned GP. The true posterior can then be evaluated in parallel at these locations. Throughout this paper, we call this procedure *sequential optimization*.

## 2.3 Nested sampling

Nested sampling (NS) [37–44] is a family of Monte Carlo sampling algorithms and, simultaneously, an integrator for probability density functions (or positive functions in general). It is based on the idea that the marginal likelihood computation can be substituted by a one-dimensional integration:

$$\int L(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} = \int_0^1 \mathfrak{L}(X)dX , \quad (5)$$

where  $\mathfrak{L}(X)$  is defined as the inverse of the cumulant prior mass containing only likelihood values greater than a given threshold  $\lambda$ :

$$X(\lambda) = \int_{L(\mathbf{x}) > \lambda} \pi(\mathbf{x})d\mathbf{x} . \quad (6)$$

The function  $\mathfrak{L}(X)$  is then sampled in increasing order by narrowing (nested) regions that contain only posterior values greater than this threshold. This is performed by tracking a set of *live* points, and sequentially discarding the one with the lowest likelihood value and substituting it for a newly-sampled one. The discarded point is weighed correspondingly to the estimated posterior volume contained within the prior shell defined between the likelihood value of the discarded point and the one of the next lowest-likelihood live point. Due to the nature of NS as an integration, Monte Carlo

---

<sup>1</sup>We use  $\zeta = d^{-0.85}$ , which has been shown in [25] to provide a good balance between exploration and exploitation in a variety of dimensionalities.

<sup>2</sup>This is because only the kernel matrix changes in this step, while the hyperparameters do not need to be refitted. Indeed, the highest cost of this operation is solely a single kernel matrix decomposition and inversion required for future predictions on this conditioned GP.

samples from NS produce a better representation of the dynamic range of the distribution than other MC samplers. A review of possible implementations and application to physical sciences can be found in [45].

In this paper, we will use the publicly available POLYCHORD code [41, 42], in particular the POLYCHORDLITE python wrapper available at <https://github.com/PolyChord/PolyChordLite>. The advantage of using this implementation: the code is well known to allow for massively parallel exploration of the desired function (see [42] for the weak and strong scaling), due to the use of slice sampling to sample from constrained likelihood contours it scales mildly with dimensionality, and due to its cluster identification algorithm it also very good at identifying global maxima even when multiple local maxima are present (see e.g. Rastrigin example in [42]).

### 3 Method

#### 3.1 Monte Carlo sampling

The basis of our method is substituting the sequential optimisation of the acquisition function using Kriging believer by the exploitation of a Monte Carlo sample of the mean of the GP. Individual samples are ranked according to their acquisition function, as explained below, in a way that reproduces a conditioned ranking similar to what would be obtained via sequential optimization.

Since the target of the acquisition procedure is the optimisation of the aquisition function, it might seem most logical to generate the MC sample directly from it. Nevertheless, there are a number of convincing arguments in favor of sampling on the mean of the GP instead:

**Speed:** Predicting the GP mean and standard deviation at multiple points simultaneously is much faster due to the possible use of vectorized matrix multiplication routines, but such vectorization is hard to exploit during optimization. While the prediction of the mean is a matrix multiplication of size  $(N_{\text{new}}, N_{\text{train}}) \times N_{\text{train}}$ , the evaluation of the standard deviation requires at least the matrix multiplication of size  $(N_{\text{train}}, N_{\text{train}}) \times (N_{\text{train}}, N_{\text{new}})$ .<sup>3</sup> As such, it is often times cheaper to first predict only the mean during the sampling (sequential) and then evaluate the standard deviation. These are then used for the acquisition function computation in a single vectorized call.

**Simplicity:** The acquisition function is often very multi-modal and rather difficult to sample while the mean for a typical well-behaved likelihood is comparatively simple. This reduces the runtime of the MC sampler (sometimes quite drastically) for a given convergence criterion of the MC sample.

**Regions of Interest:** While there almost surely exist regions far away from the mode with large standard deviations and corresponding acquisition function values, it is not always a good idea to actually sample these. This is because the actual posterior mode defines a region of interest where the accuracy of the GP is desired to be high, while other regions are not necessarily important to sample. This becomes especially interesting in moderately-high dimensionality where the volume contained by the mode becomes an ever smaller fraction of the total prior volume. However, we stress that the nested sampling employed in this work typically explores all regions relevant to the acquisition function in our examples – This region of interest is thus not an entirely strict notion.

**Reusability:** Since the nested sampling run is performed on the mean of the GP, this is effectively giving us a sample of the emulated posterior at this step, useful for inferring marginal quantities (such as credible intervals, means, variances, marginal distributions, etc.).

The use of NS in particular is advantageous with respect to Markov-chain Monte Carlo methods in this particular case: it naturally balances *exploration* and *exploitation*, since it samples the full dynamic range of the target distribution, including its tails, where low-value-but-high-variance optimal locations dwell; it is also almost-embarrassingly parallel up a number of processes similar to the number of *live* points tracked during sampling, and, depending on implementation, has a mild divergence with dimensionality (true for POLYCHORD).

---

<sup>3</sup> There is also a trace of a matrix product, requiring additional  $N_{\text{train}} \cdot N_{\text{new}}$  operations, but this is always subdominant in runtime.

After all samples have been drawn, we compute the acquisition function at these locations simultaneously and use the result to create a batch of new active sampling location proposals.<sup>4</sup>

### 3.2 Ranked acquisition pool

Instead of the sequential optimisation approach discussed in Section 2.2, we develop an algorithm to rank the MC samples according to their acquisition function value conditioned to the rest of the candidates, using Kriging believer, until an optimal batch of candidates is found. We call this approach a ranked acquisition pool (RAP). To rank a set of points, we start with the sample with the highest unconditioned acquisition as our accepted starting point. From there, we condition the GP to the already accepted samples, and rank all other points according to their acquisition function value conditioned to those accepted samples (i.e. compute the acquisition using the uncertainty of the GP conditioned to the accepted samples). We include an empty slot at the bottom of the pool for temporary sorting. Any sample in that slot will be eventually discarded. Importantly, for any acquisition function monotonic in the GP uncertainty the conditioning can only lower the acquisition value of a point.

We separate the algorithm of proposing a new sample into three main steps, and make use of Figure 1 to show examples for each (description in italics at the end of a step).

1. Initial rejection: A sample is only added if its unconditioned acquisition function is larger than the lowest conditioned acquisition function. *The sample d is rejected from the acquisition pool since its unconditioned acquisition function is smaller than those of samples a, b, c already present in the pool.*
2. Insertion and conditioning: If a sample is not rejected, it is initially inserted at the rank corresponding to its unconditioned acquisition function. If it isn't inserted at the top, it has to subsequently be conditioned to all the points above it (which typically decreases its acquisition function). If it is now lower than the next rank, it is inserted and re-conditioned there. This process is repeated until it is higher than the next rank (goes to step 3), or at the bottom of the pool and thus rejected. *Sample e is proposed to the pool, and in its unconditioned state ranks in the second position. However, after conditioning it to the first point, it performs worse than sample b and is pushed one rank down. It is then conditioned to the two points above it. This time, it performs better than sample c and thus is inserted into its current position. Since its current position is the last position of the pool no resorting is necessary.*
3. Resorting: If a sample has been inserted at any rank but the lowest, all the other ranks below are now conditioned to the wrong samples, and need to be re-conditioned and correspondingly re-ranked. This happens in an iterative fashion, where all samples in the current pool compete for the next highest position under the inserted sample (using the same conditioned GP), and the highest conditional acquisition sample is inserted there. Then the process repeats until all the slots have been filled. *The element f is added to the pool. Its unconditioned acquisition function places it at the top, and it does not need to be conditioned. This invalidates all other ranks, necessitating a full re-sorting of the pool. Next, all of (a, b, e) compete for the second slot by computing the acquisition function value conditioned to the first rank (here sample b wins). Samples a and e now compete for the third slot by computing their acquisition value when conditioned to ranks 1 and 2 simultaneously (sample e wins).*

In order to speed up especially the computations of the conditional acquisition function, the ranked pool works with a cached model of the GP regressor instances, in order to quickly compute acquisition function values conditioned to a certain rank (and those above it). A technical description of the algorithm can be found in Algorithm 1.

By giving up maximization in favour of sampling, our candidates are not the true optima of information gain, but they will be close enough to them. It is more important to get a batch of near-optimal candidates at the same time than getting just a few perfect ones.

---

<sup>4</sup>The authors of [24] also perform an MC of the mean GP, but do not take care of conditioning when selecting optimal candidates, as we do in the next section.

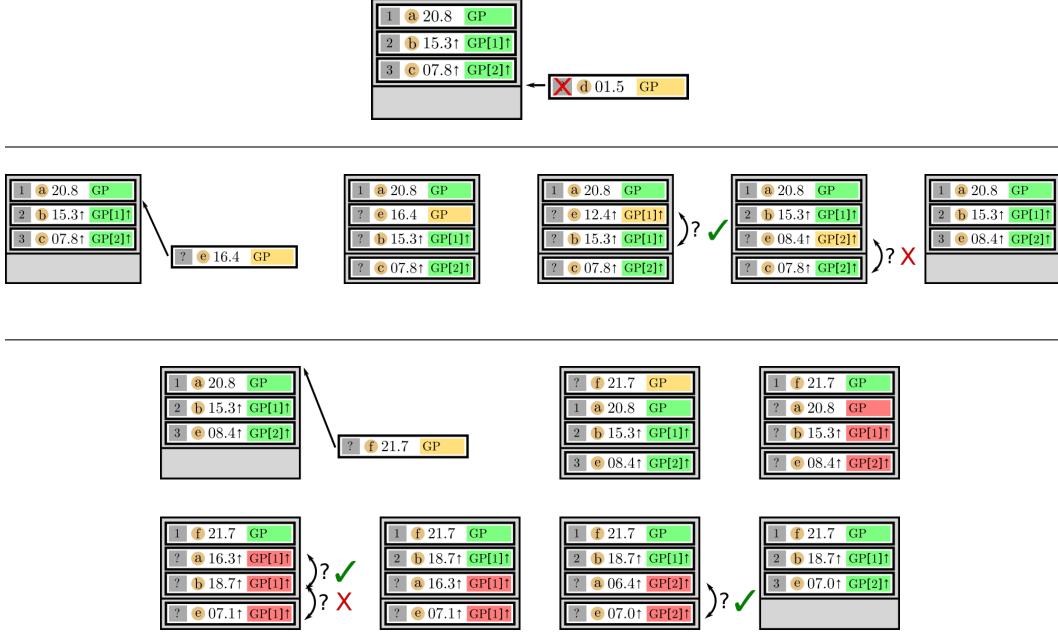


Figure 1: Three insertion cases in a ranked pool of size 3, with one empty slot below it. Each rectangular box represents a single sample. The number in the grey box represents current known rank of the sample (? means un-ranked), the letter in the orange circles is an identifier of the points, the number next to it is the current (conditioned) acquisition function, and the green/orange/red box at the end shows if a point is conditioned to a given rank and those above it (number in angular brackets) and the status of the acquisition value (green=up-to-date, orange=newly inserted and possibly in need of conditioning, red=invalidated by insertion at higher rank). The three cases are described in the main text.

---

**Algorithm 1** The ranked pool updating routine in pythonic pseudo-code.

---

**Known:** Samples  $X_1 \dots X_N$  with stored conditional acquisitions  $a[0] \dots a[N]$   
**Require:** New sample  $X$ ,  $a(X)$

```

1:  $i \leftarrow N$                                      ▷ Rejection check
2: if  $a(X) > a[N]$  then
3:    $\text{reject}(X)$ 
4: end if                                         ▷ Finding the correct insertion position
5: while  $i > 0$  do
6:    $c = a(X)|((i-1), \dots, 1)$ 
7:   if  $c > a[i-1]$  then
8:      $i \leftarrow i - 1$ 
9:   else
10:     $\text{insert}(X, i)$ 
11:   end if
12: end while                                     ▷ Resorting + rebuilding the cache:
13: while  $i < N-1$  do
14:   for  $j$  in range( $i+1, N$ ) do
15:      $\text{compute } c_j = a(X_j)|(i, \dots, 1)$           ▷ Update conditioned acquisitions
16:   end for
17:    $m = \text{argmax}[c_{i+1}, \dots, c_n]$ 
18:    $\text{swap}(X_{i+1}, X_m)$ 
19:    $a[i+1] \leftarrow c_m$ 
20: end while

```

---

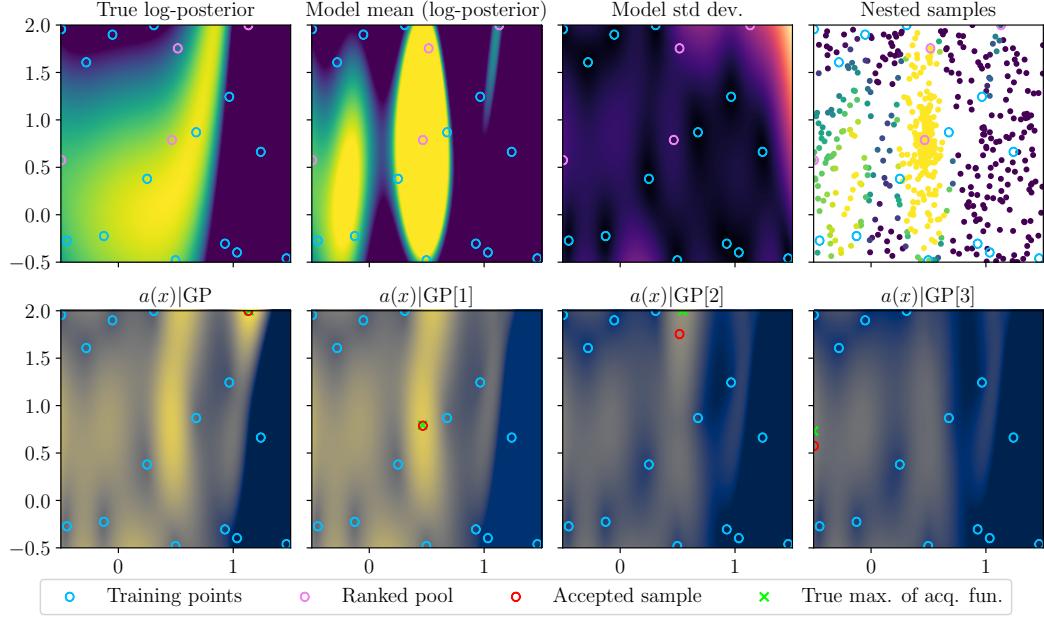


Figure 2: Acquisition procedure with a ranked pool of size  $N = 4$ . The top row shows from left to right: The true function to be emulated, the current GP mean prediction, it's standard deviation, the nested samples (dead points) from POLYCHORD. The bottom row shows the acquisition function for the unconditioned GP on the left, and for the conditioned GPs in the three right panels (each conditioned to all samples added to its left). Blue circles are current training samples, pink circles are samples that have been accepted into the ranked pool (top), and red circles are each respective optimal sample for the conditioned GP (bottom). Note that this example is very early in the active sampling so the mode has not been well mapped. Nevertheless it is visible that even with very few samples the locations of the nested samples still cover the regions of high acquisition function well.

## 4 Results

The combination of the nested sampling approach with the ranked acquisition pool is implemented as NORA (Nested sampling Optimization for Ranked Acquisition), based on the GP treatment from [25, 46] (as well as useful functionality from [47, 48]). A demonstration of the acquisition procedure in NORA can be found in fig. 2.

We tested NORA on a number of synthetic likelihoods to demonstrate both the accuracy and the highly parallel nature of our approach. The likelihoods for accuracy tests include a curved degeneracy, a ring, and the multi-modal Himmelblau function. Further discussion of these synthetic examples can be found in appendix D, while real-world applications to cosmological data can be found in appendix E.

The curved degeneracy (see also [25, 49]) has a tight ridge in the  $x_2 \approx 4x_1^4$  direction, and its log-likelihood is

$$\log L(x_1, x_2) = -(10 \cdot (0.45 - x_1))^2 / 4 - (20 \cdot (x_2 / 4 - x_1^4))^2. \quad (7)$$

The log-likelihood for the ring example is instead

$$\log L(x_1, x_2) = -\frac{1}{2} \left[ \frac{(\sqrt{x_1^2 + x_2^2} - \mu)^2}{\sigma} + \log(2\pi\sigma^2) \right], \quad (8)$$

where  $\mu = 1$  and  $\sigma = 0.05$  in our example. We show in Figure 3 that in both of these cases the accuracy and efficiency is very comparable to the sequential method (while much more parallelizable, see below). Both reach about the same level of agreement between emulation and the true function

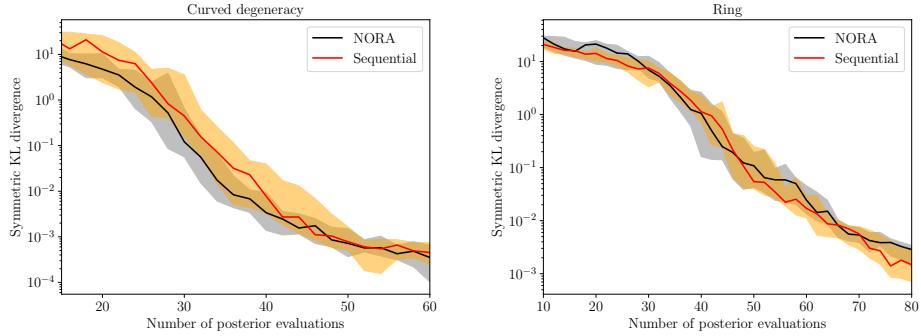


Figure 3: Comparison of the efficiency and accuracy of the acquisition procedure between the naive sequential optimization approach and the NORA approach. We show the agreement between the emulated and the true posterior (specified by the symmetric KL divergence) as a function of the number of samples (posterior evaluations). The solid line is the median, and the shaded region is the 25% to the 75% quantiles of 20 realizations. In this case NORA shows similar performance to the sequential algorithm.

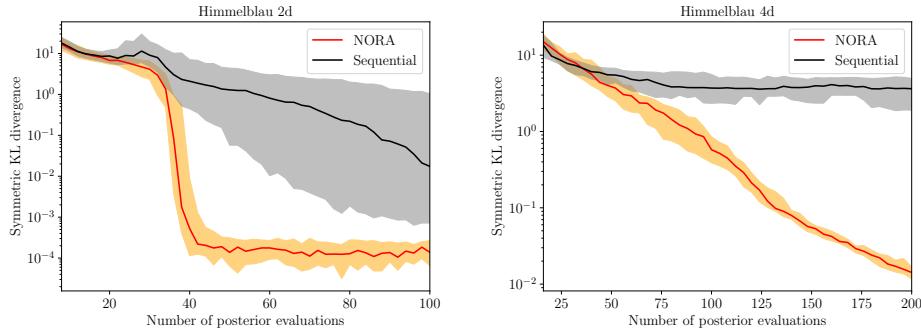


Figure 4: Same as Figure 3 for the Himmelblau function (left) and a four-dimensional extension with 4 modes in two of the dimensions (right). In the other two dimensions it is flat. In these multi-modal cases the NORA algorithm is far more efficient than the sequential sampling algorithm.

(as captured by their symmetric KL divergence, which is further explained in appendix C). We also investigate a multi-modal example like the Himmelblau function with log-likelihood

$$\log L(x_1, x_2) = -\frac{1}{2} [100 \cdot (x_1^2 - x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2] \quad (9)$$

We furthermore construct a four-dimensional version of this function which retains the four maxima in two dimensions but is constant along the other two dimensions. This combines the multimodality with the problem of correctly mapping and exploring the flat dimensions. We show the results in Figure 4. Since in this case the nested sampling has a far higher chance of quickly discovering a mode of the function far from the already known ones, the NORA approach is much more efficient than the Sequential optimization approach in this case (we show examples of explicit modeling for 100 posterior evaluations in Figure 5).

In order to assess that gains in modelling do not come at the cost of overhead in the acquisition step, we have performed a number of tests in Gaussian likelihoods at different dimensionalities. The comparison with the costs of acquisition with sequential optimization and NORA, as well as the scaling with parallelization is shown in Table 1. We see that the overhead of NORA is comparable to that of sequential optimization for the same number of MPI processes. However, sequential optimization will only profit from parallelization up to the number of restarts of the optimizer while nested sampling will parallelize virtually infinitely (up to the large number of live points).

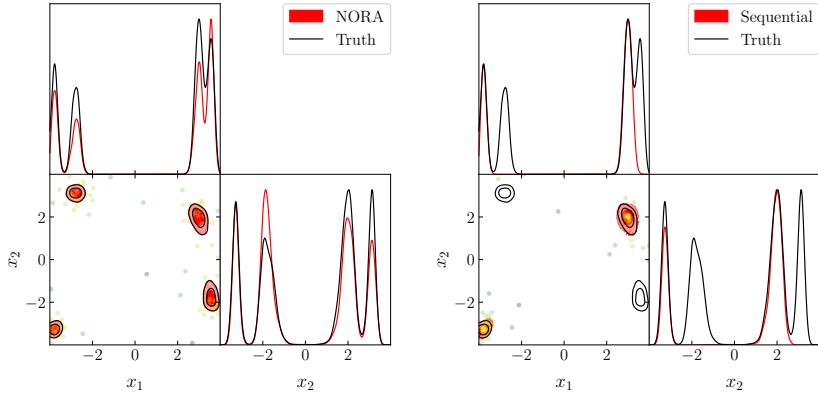


Figure 5: Example of a failure case of the naive sequential optimization compared to the same cases treated with NORA, which due to its nested sampling correctly identifies all modes. The sampling locations are shown with colour denoting how late they were sampled (yellower=later). It is clearly visible that the sequential optimization is sampling more aggressively towards the top of the mode and showing less explorative behaviour. Both are allowed 100 posterior evaluations.

	$d = 2$		$d = 4$		$d = 8$	
SeqOpt	[2]	[2.4, <b>2.55</b> , 2.7]	[4]	[11.2, <b>12.1</b> , 12.7]	[8]	[177, <b>183</b> , 191]
NORA	[2]	[3.67, <b>3.84</b> , 4.34]	—	—	—	—
NORA	[4]	[1.43, <b>1.54</b> , 1.75]	[4]	[9.3, <b>9.9</b> , 10.6]	—	—
NORA	[8]	[0.96, <b>1.05</b> , 1.13]	[8]	[6.29 <b>6.58</b> 6.90]	[8]	[147, <b>160</b> , 220]
NORA	[16]	[0.30, <b>0.33</b> , 0.35]	[16]	[4.52, <b>4.77</b> , 5.16]	[16]	[122, <b>129</b> , 171]

Table 1: Comparison of wall-clock runtimes for the acquisition step between NORA and sequential optimization (dubbed "SeqOpt", with  $5 \cdot d$  restarts of the optimizer). We add in angular brackets the number of MPI processes. In each dimensionality we show the [25, **50**, 75] percent quantiles. We run 50 runs in 2- and 4 dimensions, and 20 runs in 8 dimensions, with respective truth evaluation budgets 20, 60 and 400. Convergence in terms of symmetric KL divergence is similar in all cases and of magnitude  $\mathcal{O}(0.01)$ . We additionally allow multi-threading (useful e.g. for BLAS [50] matrix operations) for each MPI process up to a total of 32 cores.

## 5 Conclusion

Sequential optimization for active learning is facing a variety of challenges, such as difficult parallelization, and a lack of robustness to getting stuck in local maxima, thus requiring many restarts of the optimizer in high dimensions to properly explore the target inference space. To overcome these challenges we propose a new algorithm, called NORA, that substitutes the sequential optimization of the acquisition function by combining Monte Carlo exploration of the GP's mean using Nested Sampling, and ranking of the Monte Carlo samples according to their conditional acquisition function values, to generate a nearly optimal batch of sampling locations. These two steps can be performed in a nearly perfectly-parallelizable way, and the same Monte Carlo sample can be reused in consecutive iterations for lowering computational costs.

We apply NORA to a number of synthetic Bayesian inference problems to assess its performance, and compare it to a reasonably good implementation of sequential optimisation of the acquisition function.

We find that NORA and sequential optimization perform equally well at comparable computational costs for simple unimodal likelihoods for  $d < 10$ , and for highly non-Gaussian likelihoods in small dimensionalities. NORA greatly outperforms sequential optimization for multi-modal likelihoods, due to the more exploratory approach to acquisition, despite producing less precise acquisition batches than sequential optimization.

The limitations of the NORA algorithm are similar to those of other approaches to Bayesian inference based on surrogate GP models: Their strong divergence with dimensionality due to the increasingly large number of training points needed for good posterior modelling, and the  $\mathcal{O}(n^3)$  scaling when fitting of the hyperparameters of the GP (see also [51]). Furthermore, one particular shortcoming of NORA compared to sequential optimization is that due to its less aggressive acquisition it will converge later in simple problems, e.g. Gaussian likelihoods. Our methodology also does not address the problem of stochastic likelihood evaluations (see [27]).

### Acknowledgments and Disclosure of Funding

J. T. acknowledges support from the STARS@UNIPD2021 project *GWCross*. 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.

## A Description of the surrogate model

In this section we describe details of the GP surrogate model (see also [25] for a detailed description).

### A.1 Choice of kernel function

On top of the choice of the kernel function itself, as defined in Equation (3), some knowledge of the target function is also incorporated in the priors for the hyperparameters. Our assumption is that the length scales should be of an order of magnitude close to that of the posterior modes, while 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 by setting the prior of the length scales to being uniform between 0.01 and 1 in units of the prior length in each direction. 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. The prior of the output scale  $C$  is chosen to be very broad and allows for values between 0.001 and 10000. The  $d + 1$  free hyperparameters  $\theta \equiv \{C, l_i\}$  are then chosen such that they maximize

$$-\log p(\mathbf{y}|\mathbf{X}, \theta) = \frac{1}{2} \mathbf{y}^T (\mathbf{k}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} + \frac{1}{2} \log |\mathbf{k}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}| - \frac{N_s}{2} \log 2\pi. \quad (10)$$

where  $\sigma_n$  is a small noise parameter that typically improves numerical stability of the matrix inversion.

### A.2 Parameter space transformations

To ensure numerical stability we use a number of transformations during the modelling with the GP:

Firstly, we sample the log-posterior distribution to reduce the scale of the function that the GP interpolates. Furthermore, the characteristic length scale of isotropic kernels tends to be larger when sampling the log-posterior, which implies that the GP surrogate generalizes better to distant parts of the function, making the GP more predictive.

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

$$\log \tilde{p}(\mathbf{X}) = \frac{\log p(\mathbf{X}) - \bar{y}}{s_y}, \quad (11)$$

where  $\bar{y}$  and  $s_y$  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.

As for the space of parameters  $\mathbf{x}$ , we transform the samples such that the prior boundary becomes a unit-length hypercube. 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 [52], used to optimize the GP hyperparameters.

## B Some details of the algorithm

Active-sampling Bayesian inference algorithms based on surrogate models in the literature [19–28] usually follow a fixed procedure: after an initial batch of training samples is either provided or drawn from the prior, the algorithm iterates on a cycle of (1) optimising an acquisition function to obtain candidates for evaluation of the true posterior, (2) evaluation of the true posterior at the proposed locations, (3) refitting of the surrogate model, and (4) convergence checks. In this study we do not concern ourselves with the initial proposal (in our case sampled from the prior) or the convergence checks (in most examples we have fixed budgets of how many true posterior points are sampled), since the focus of this study is on the acquisition step.

As discussed in the main text, our acquisition procedure has two steps: first the mean GP is explored using nested sampling, and, second, the resulting MC samples are ranked according to their conditioned acquisition function value. In this short appendix, we discuss some particularities of these procedures.

## B.1 Scaling of Nested Sampling precision parameters

The two fundamental parameters of a nested sampler are the number of *live points*, and the fraction of the total posterior mass (evidence) contained in the final set of live points. Additional parameters depend on the particular implementation of NS. In POLYCHORD, our sampler of choice, the aforementioned parameters are called respectively `nlive` and `precision_criterion`. In addition, and among others, POLYCHORD has two more important parameters: the length of the slice-sampling chains (`num_repeats`), and the size of the initial prior sample from which the live points are extracted (`nprior`). It is a natural choice that in the early stages of learning, where limited precision when optimising the acquisition function is enough, this would translate in our procedure into lower precision settings for POLYCHORD. To reflect this, we scale the number of live points to be proportional to the number of points in the training set (by a factor of 3 by default), with a cap equal to the default precision criterion of POLYCHORD, which is 25 times the dimensionality of the problem. On the other hand, we have found that the accuracy of our algorithm benefits from more accuracy than the default for the length of slice chains (`num_repeats`, 5 times the dimensionality instead of 2), whereas the evidence fraction contained in the live points (`precision_criterion`) can be relaxed with respect to the default by a factor of 5, since we are not interested in an accurate calculation of the model evidence.

## B.2 Byproducts of Nested Sampling

The nested sampling step produces both a MC sample and a calculation of the evidence of the model. The first one is a useful by-product, which can be used for inference once the run has converged, or to implement a global convergence criterion, such as one based on the calculation of KL divergences between iterations. The value of the evidence is also a useful output, in particular to define a further convergence criterion, but it needs to be taken into account that the resulting NS uncertainty does not include the uncertainty due to the probabilistic nature of the GP, or the uncertainty over the choice of hyperparamenters values, as Bayesian Quadrature approaches do.

### B.2.1 Parallelization

Nested samplers parallelize effectively up to the number of live points (`nlive`), since parallel evaluation of the target function increases the chance that at every iteration an acceptable sample will be found at the cost of a single evaluation. Since this number is usually a few tens of times the dimensionality, this step of our algorithm will effectively parallelize linearly with the number of simultaneous processes. POLYCHORD does not do vectorized evaluation of the target function, i.e. the target function is always called with a single argument. Hence for this step we prefer to invest CPU cores into separate MPI processes, as opposed to multiple threads.

The ranking step of the algorithm when running NORA in parallel occurs in two steps: first the MC sample is split in as many equal parts as running processes, for evaluation of the GP standard deviation and the acquisition function value, and the individual ranking of each subset into ranked pools with as many points as the desired Kriging believer steps; and later all the ranked pools are combined and re-ranked in a single process. The first of these two steps can be effectively parallelized, but the second one is not parallelizable by definition, and may at most benefit for multi-threading. In most situations, unless the size of the training set is very large, the first step is costlier and thus a larger number of MPI processes is more beneficial than a larger number of threads per process.

Finally, the evaluation step occurs always in parallel when MPI processes are available, but its parallelization is limited by the number of Kriging believer steps we have decided to take. This number must be kept in check because the quality of the batch of proposals decreases when conditioning on increasingly bad information, making our model larger and more computationally expensive. We have found that a number of Kriging believer steps equal to the dimensionality is a good choice in most cases. Highly multimodal posterior can benefit from larger number of Kriging believer steps, since their acquisition functions have more local maxima, but it would not be wise to go beyond a few times the number of dimensions. Thus, the evaluation step benefits from the number of MPI processes in a limited way, and may be faster if more cores are left available for multi-threading, thus accelerating the evaluation of the true posterior.

The difference between the acquisition step benefiting from a large number of MPI processes, and the evaluation step potentially benefiting more from a large number of threads, makes the choice of the

ratio of MPI processes to threads per process dependent on the speed of the evaluation step and the overhead costs, which scale with dimensionality: fast true posteriors in high dimensionality call for larger number of MPI threads, and very slow posterior with an implementation that benefits from multi-threading would call for a larger amount of threads and a smaller amount of MPI processes. In the future, we will look at substituting POLYCHORD by a nested sampler that can perform vectorized calls to the target function, in order to make multi-threading an overall better choice, beyond the small necessary MPI parallelization for Kriging believer.

## C Kullback-Leibler Divergences

We define the Kullback-Leibler (KL) divergence of the continuous probability distribution  $P$  with respect to  $Q$  with probability density functions  $p(\mathbf{x})$  and  $q(\mathbf{x})$  as

$$D_{\text{KL}}(P||Q) = \int p(\mathbf{x}) \log \left( \frac{p(\mathbf{x})}{q(\mathbf{x})} \right) d\mathbf{x}. \quad (12)$$

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

$$D_{\text{KL}}^{\text{sym}}(P, Q) = \frac{1}{2} (D_{\text{KL}}(P||Q) + D_{\text{KL}}(Q||P)). \quad (13)$$

A smaller value means that the two posteriors are in better agreement, and one typically wants  $D_{\text{KL}}^{\text{sym}}(P||Q) \ll 1$  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  $p(\mathbf{x})d\mathbf{x}$ . One can thus approximate the integral as a sum of the quantity  $\log p(\mathbf{x}_i) - \log q(\mathbf{x}_i)$  over all points in the MC sample (multiplied by their respective weights/multiplicities). This can be done by evaluating either the real model or the GP emulated posterior for the given points.

There also exists a Gaussian approximation for the KL divergence which is particularly useful when computing the true log-posteriors at each point of the MC sample is computationally undesirable (such as the cosmological examples below). It is defined as

$$D_{\text{KL}}(P||Q) \approx \frac{1}{2} \left( \text{tr} \left( \mathbf{C}_Q^{-1} \mathbf{C}_P \right) - d + (\mathbf{m}_Q - \mathbf{m}_P)^T \mathbf{C}_Q^{-1} (\mathbf{m}_Q - \mathbf{m}_P) + \log \left( \frac{\det \mathbf{C}_Q}{\det \mathbf{C}_P} \right) \right). \quad (14)$$

with  $\mathbf{C}_Q$  and  $\mathbf{C}_P$  being the respective covariance matrices of the two probability distributions, while  $\mathbf{m}_Q$  and  $\mathbf{m}_P$  are the respective means. While the approximation of the individual distribution as multivariate Gaussian is certainly incorrect in non-Gaussian cases, it is typically the case that a good agreement of the Gaussian KL signals a good compatibility of the true KL as well. We always compute the true symmetric KL unless explicitly stated otherwise.

## D Test functions

Here we comment further on the test functions presented in Section 4 of the main text.

Figure 6 and Figure 7 show exemplary corner plots of the examples used in Section 4. In all three multi-modal cases presented in that section, NORA correctly recovers the contours.

In Section 4 we also presented a study of the parallelization of the overhead costs of NORA. In this context, in Figure 8 we show comparisons in convergence between NORA and sequential optimization for Gaussians drawn with random correlations in 2, 4 and 8 dimensions. For these very easy-to-model functions NORA converges as fast as sequential optimization. The slightly slower convergence in  $d = 8$  is likely due to the somewhat more exploratory behaviour of NORA compared to sequential optimization.

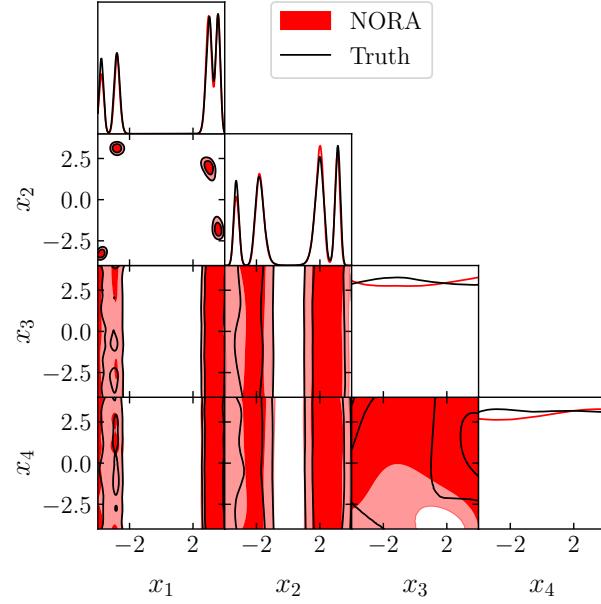


Figure 6: Example of inference on the 4d Himmelblau example. The four modes are in the  $x_1$ - $x_2$  direction while the other two directions are flat. The example shows NORA sampling with a budget of 200 posterior evaluations. Both contours are in good agreement with each other.

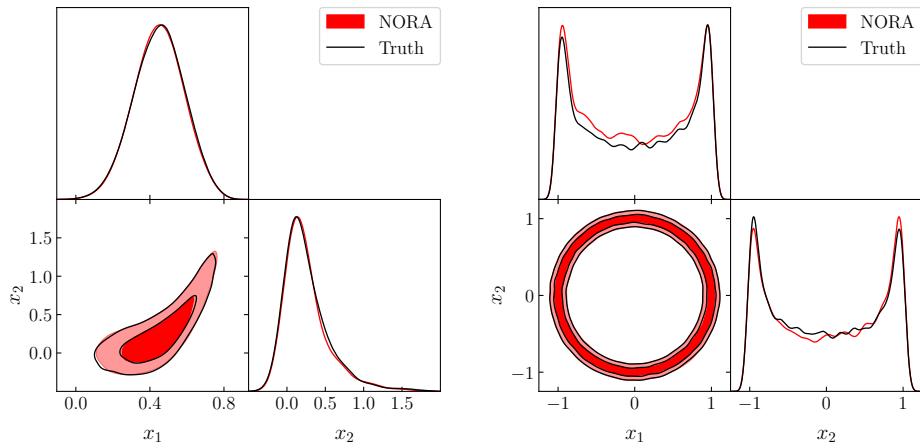


Figure 7: Example of inference on the curved degeneracy example (left) and the ring (right). NORA correctly recovers both contours. Both runs have been performed with a budget of 80 posterior evaluations.

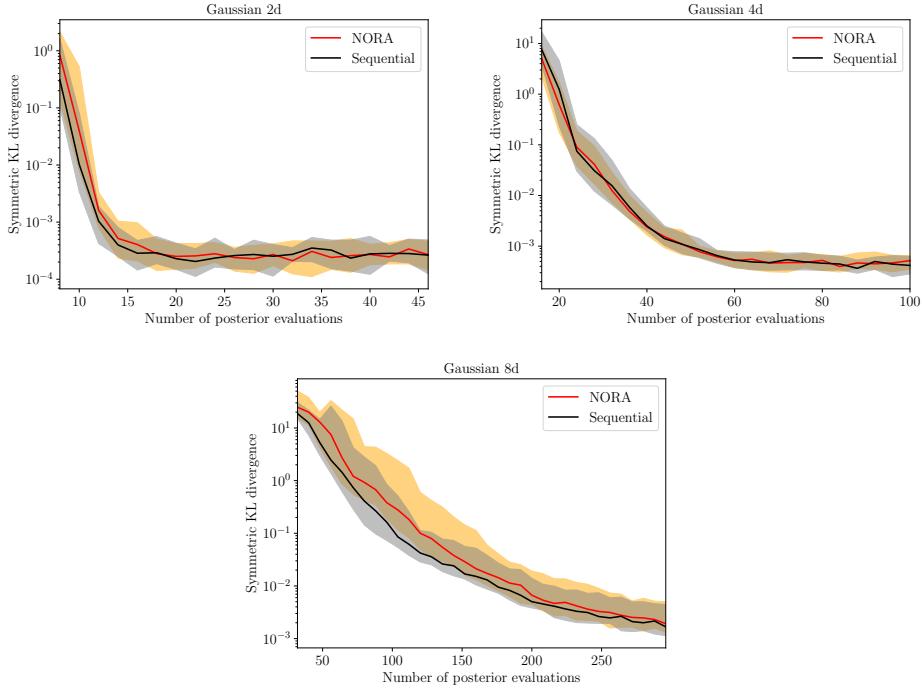


Figure 8: Comparing convergence of NORA vs. Sequential optimization for randomly drawn Gaussians in 2, 4, 8 dimensions. NORA and sequential optimization perform nearly equally well for these easy-to-model likelihoods.

## E Cosmological examples

In order to test the applicability and robustness of the NORA implementation to real-world examples, we also apply it to a number of inference runs commonly used in cosmology. In particular, we use the Planck 2018 temperature, polarization, and lensing data (using the nuisance-marginalized ‘lite’ version, as described in [53, 54]), and consider either a model of a curved universe ( $\Lambda$ CDM +  $\Omega_k$ ) or a model with sinusoidal variations of the primordial power spectrum, similar to [55, Sec 7.1.1]. For the  $\Lambda$ CDM baseline model in both cases we adopt the common 6 cosmological parameters  $\{\ln(10^{10} A_s), n_s, H_0, \Omega_b h^2, \Omega_{\text{cdm}} h^2, \tau_{\text{reio}}\}$  and adopt a single massive neutrino with mass 0.06eV (see [56] for a more detailed description of this baseline model).

We show in Figure 9 the results for a model of a curved universe, an extension of the  $\Lambda$ CDM model described above with an additional seventh parameter  $\Omega_k$  representing the energy density-equivalent of the curvature. It presents a particularly strong degeneracy between the curvature parameter  $\Omega_k$  and the Hubble constant  $H_0$ . We observe that the contours are in good agreement ( $D_{\text{KL}}^{\text{sym, Gaussian}} = 0.08$  using the Gaussian approximation of Equation (14)).

Next we try to fit a sinusoidal oscillation with three parameters (amplitude, wavelength and phase) to the primordial power spectrum of Planck 2018, fixing the parameters of the  $\Lambda$ CDM model. This is a low-dimensional problem, but with a highly multi-modal behavior in the frequency and phase of the oscillation, since we are effectively fitting experimental noise. The result can be seen in Figure 10: most of the distribution is well recovered, despite its complexity.

## F Reproducibility

The NORA implementation and all scripts required to reproduce the tests will be released after review of this manuscript.

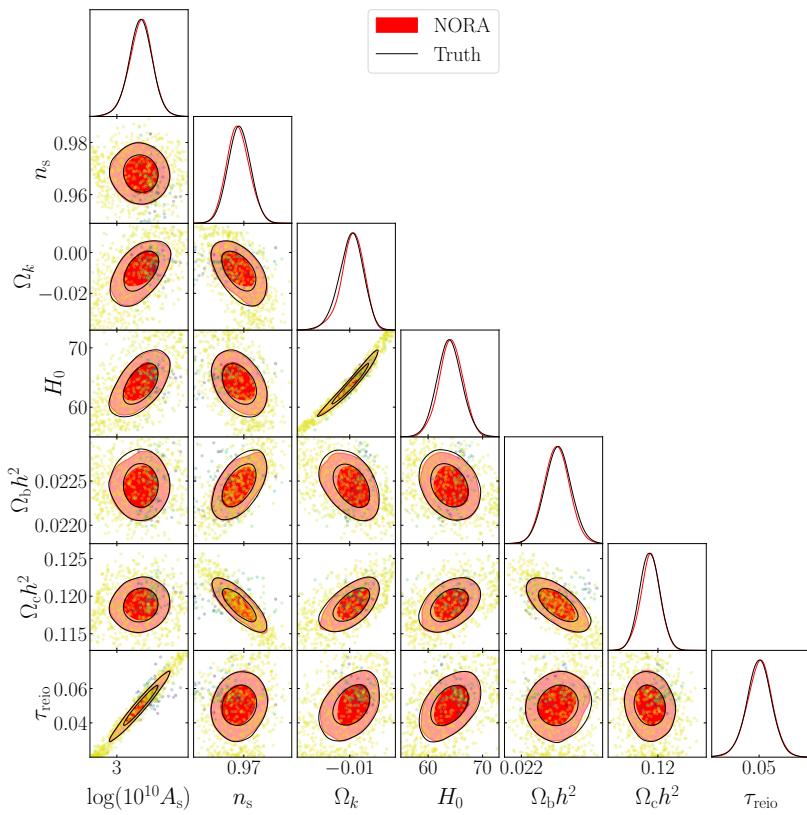


Figure 9: Inference of the cosmological parameters of the Planck 2018 likelihood (Planck lite) with curvature  $\Omega_k$  sampled in addition. NORA correctly recovers the contours with only 903 evaluations of the likelihood function.

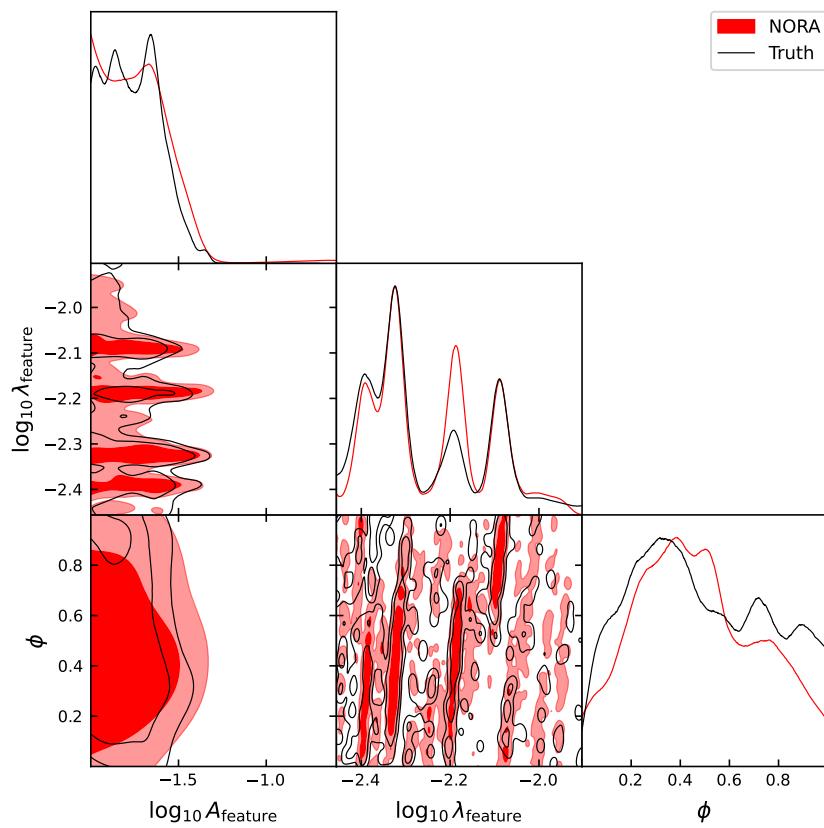


Figure 10: A three dimensional model of an oscillation in the primordial power spectrum of the Planck 2018 CMB sky constrained by NORA, and the reference Nested Sampling results with PolyChord. NORA accurately captures the constraints with a budget of 2000 evaluations, performed in parallel batches of 6 evaluations.

## References

- [1] Philip Graff, Farhan Feroz, Michael P. Hobson, and Anthony Lasenby. BAMBI: blind accelerated multimodal bayesian inference. *Monthly Notices of the Royal Astronomical Society*, jan 2012. doi: 10.1111/j.1365-2966.2011.20288.x. URL <https://doi.org/10.1111%2Fj.1365-2966.2011.20288.x>.
- [2] Adam Moss. Accelerated Bayesian inference using deep learning. *Mon. Not. Roy. Astron. Soc.*, 496(1):328–338, 2020. doi: 10.1093/mnras/staa1469.
- [3] Hector J. Hortua, Riccardo Volpi, Dimitri Marinelli, and Luigi Malago. Accelerating MCMC algorithms through Bayesian Deep Networks. In *34th Conference on Neural Information Processing Systems*, 11 2020.
- [4] Michael J. Williams, John Veitch, and Chris Messenger. Nested sampling with normalizing flows for gravitational-wave inference. *Phys. Rev. D*, 103(10):103006, 2021. doi: 10.1103/PhysRevD.103.103006.
- [5] Minas Karamanis, David Nabergoj, Florian Beutler, John A. Peacock, and Uros Seljak. pocoMC: A Python package for accelerated Bayesian inference in astronomy and cosmology. *J. Open Source Softw.*, 7(79):4634, 2022. doi: 10.21105/joss.04634.
- [6] Florent Leclercq. Bayesian optimization for likelihood-free cosmological inference. *Phys. Rev. D*, 98(6):063511, 2018. doi: 10.1103/PhysRevD.98.063511.
- [7] Justin Alsing, Tom Charnock, Stephen Feeney, and Benjamin Wandelt. Fast likelihood-free cosmology with neural density estimators and active learning. *Mon. Not. Roy. Astron. Soc.*, 488(3):4440–4458, 2019. doi: 10.1093/mnras/stz1960.
- [8] Benjamin Kurt Miller, Alex Cole, Gilles Louppe, and Christoph Weniger. Simulation-efficient marginal posterior estimation with swyft: stop wasting your precious time. 11 2020.
- [9] T. Lucas Makinen, Tom Charnock, Justin Alsing, and Benjamin D. Wandelt. Lossless, scalable implicit likelihood inference for cosmological fields. *JCAP*, 11(11):049, 2021. doi: 10.1088/1475-7516/2021/11/049. [Erratum: JCAP 04, E02 (2023)].
- [10] Biwei Dai and Uros Seljak. Translation and rotation equivariant normalizing flow (TRENF) for optimal cosmological analysis. *Mon. Not. Roy. Astron. Soc.*, 516(2):2363–2373, 2022. doi: 10.1093/mnras/stac2010.
- [11] Justine Zeghal, Francois Lanusse, Alexandre Boucaud, Benjamin Remy, and Eric Aubourg. Neural Posterior Estimation with Differentiable Simulator. In *Machine Learning for Astrophysics*, page 52, July 2022. doi: 10.48550/arXiv.2207.05636.
- [12] Pablo Lemos, Miles Cranmer, Muntazir Abidi, ChangHoon Hahn, Michael Eickenberg, Elena Massara, David Yallup, and Shirley Ho. Robust simulation-based inference in cosmology with Bayesian neural networks. *Mach. Learn. Sci. Tech.*, 4(1):01LT01, 2023. doi: 10.1088/2632-2153/acbb53.
- [13] Andrea Manrique-Yus and Elena Sellentin. Euclid-era cosmology for everyone: neural net assisted MCMC sampling for the joint  $3 \times 2$  likelihood. *Mon. Not. Roy. Astron. Soc.*, 491(2):2655–2663, 2020. doi: 10.1093/mnras/stz3059.
- [14] Arrykrishna Mootoovaloo, Alan F. Heavens, Andrew H. Jaffe, and Florent Leclercq. Parameter Inference for Weak Lensing using Gaussian Processes and MOPED. *Mon. Not. Roy. Astron. Soc.*, 497(2):2213–2226, 2020. doi: 10.1093/mnras/staa2102.
- [15] Alessio Spurio Mancini, Davide Piras, Justin Alsing, Benjamin Joachimi, and Michael P. Hobson. CosmoPower: emulating cosmological power spectra for accelerated Bayesian inference from next-generation surveys. *Mon. Not. Roy. Astron. Soc.*, 511(2):1771–1788, 2022. doi: 10.1093/mnras/stac064.
- [16] Chun-Hao To, Eduardo Rozo, Elisabeth Krause, Hao-Yi Wu, Risa H. Wechsler, and Andrés N. Salcedo. LINNA: Likelihood Inference Neural Network Accelerator. *JCAP*, 01:016, 2023. doi: 10.1088/1475-7516/2023/01/016.

- [17] Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference. *JCAP*, 05:025, 2023. doi: 10.1088/1475-7516/2023/05/025.
- [18] Sven Günther, Julien Lesgourgues, Georgios Samaras, Nils Schöneberg, Florian Stadtmann, Christian Fidler, and Jesús Torrado. CosmicNet II: emulating extended cosmologies with efficient and accurate neural networks. *JCAP*, 11:035, 2022. doi: 10.1088/1475-7516/2022/11/035.
- [19] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl Rasmussen. Active learning of model evidence using bayesian quadrature. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL [https://proceedings.neurips.cc/paper\\_files/paper/2012/file/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2012/file/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Paper.pdf).
- [20] Tom Gunter, Michael A Osborne, Roman Garnett, Philipp Hennig, and Stephen J Roberts. Sampling for inference in probabilistic models with fast bayesian quadrature. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL [https://proceedings.neurips.cc/paper\\_files/paper/2014/file/e94f63f579e05cb49c05c2d050ead9c0-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2014/file/e94f63f579e05cb49c05c2d050ead9c0-Paper.pdf).
- [21] K. Kandasamy, J. Schneider, and B. Póczos. Bayesian active learning for posterior estimation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3605–3611, 2015. ISBN 978-1-577-35738-4.
- [22] Henry Chai and Roman Garnett. Improving Quadrature for Constrained Integrands. *arXiv e-prints*, art. arXiv:1802.04782, February 2018. doi: 10.48550/arXiv.1802.04782.
- [23] Hongqiao Wang and Jinglai Li. Adaptive Gaussian Process Approximation for Bayesian Inference with Expensive Likelihood Functions. *Neural Computation*, 30(11):3072–3094, 11 2018. ISSN 0899-7667. doi: 10.1162/neco\_a\_01127. URL [https://doi.org/10.1162/neco\\_a\\_01127](https://doi.org/10.1162/neco_a_01127).
- [24] Marcos Pellejero-Ibañez, Raul E. Angulo, Giovanni Aricó, Matteo Zennaro, Sergio Contreras, and Jens Stücker. Cosmological parameter estimation via iterative emulation of likelihoods. *Mon. Not. Roy. Astron. Soc.*, 499(4):5257–5268, 2020. doi: 10.1093/mnras/staa3075.
- [25] Jonas El Gammal, Nils Schöneberg, Jesús Torrado, and Christian Fidler. Fast and robust Bayesian Inference using Gaussian Processes with GPry. 11 2022.
- [26] Luigi Acerbi. Variational bayesian monte carlo. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/747c1bcc6109a4ef936bc70cfe67de-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/747c1bcc6109a4ef936bc70cfe67de-Paper.pdf).
- [27] Luigi Acerbi. Variational bayesian monte carlo with noisy likelihoods. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8211–8222. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/5d40954183d62a82257835477ccad3d2-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/5d40954183d62a82257835477ccad3d2-Paper.pdf).
- [28] Bobby Huggins, Chengkun Li, Marlon Tobaben, Mikko J. Aarnos, and Luigi Acerbi. Pyvbmc: Efficient bayesian inference in python, 2023.
- [29] Zoubin Ghahramani and Carl Rasmussen. Bayesian monte carlo. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL [https://proceedings.neurips.cc/paper\\_files/paper/2002/file/24917db15c4e37e421866448c9ab23d8-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2002/file/24917db15c4e37e421866448c9ab23d8-Paper.pdf).

- [30] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass. [u.a.], 2006. ISBN 0-262-18253-X and 978-0-262-18253-9.
- [31] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(119):4053–4103, 2014. URL <http://jmlr.org/papers/v15/desautels14a.html>.
- [32] Clément Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In Giuseppe Nicosia and Panos Pardalos, editors, *Learning and Intelligent Optimization*, pages 59–69, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-44973-4.
- [33] 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)*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 648–657, May 2016. URL <http://jmlr.org/proceedings/p51/gonzalez16a.pdf>.
- [34] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Technical report, March 2008. URL <https://hal.archives-ouvertes.fr/hal-00260579>.
- [35] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. *Kriging Is Well-Suited to Parallelize Optimization*, volume 2, pages 131–162. 01 2010. doi: 10.1007/978-3-642-10701-6\_6.
- [36] R. J. Barnes and A. G. Watson. Efficient updating of kriging estimates and variances. *Mathematical Geology*, 24(1):129–133, Jan 1992. ISSN 1573-8868. doi: 10.1007/BF00890091.
- [37] John Skilling. Nested sampling. *AIP Conference Proceedings*, 735(1):395–405, 2004. doi: 10.1063/1.1835238.
- [38] F. Feroz and M. P. Hobson. Multimodal nested sampling: an efficient and robust alternative to markov chain monte carlo methods for astronomical data analyses. *Monthly Notices of the Royal Astronomical Society*, 384(2):449–463, Jan 2008. ISSN 1365-2966. doi: 10.1111/j.1365-2966.2007.12353.x.
- [39] F. Feroz, M. P. Hobson, and M. Bridges. Multinest: an efficient and robust bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, Oct 2009. ISSN 1365-2966. doi: 10.1111/j.1365-2966.2009.14548.x.
- [40] Farhan Feroz, Michael P. Hobson, Ewan Cameron, and Anthony N. Pettitt. Importance nested sampling and the multinest algorithm. *The Open Journal of Astrophysics*, 2(1), Nov 2019. ISSN 2565-6120. doi: 10.21105/astro.1306.2144.
- [41] W. J. Handley, M. P. Hobson, and A. N. Lasenby. PolyChord: nested sampling for cosmology. *Mon. Not. Roy. Astron. Soc.*, 450(1):L61–L65, 2015. doi: 10.1093/mnrasl/slv047.
- [42] W. J. Handley, M. P. Hobson, and A. N. Lasenby. polychord: next-generation nested sampling. *Mon. Not. Roy. Astron. Soc.*, 453(4):4385–4399, 2015. doi: 10.1093/mnras/stv1911.
- [43] Edward Higson, Will Handley, Michael Hobson, and Anthony Lasenby. Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Statistics and Computing*, 29(5):891–913, dec 2018. doi: 10.1007/s11222-018-9844-0. URL <https://doi.org/10.1007%2Fs11222-018-9844-0>.
- [44] Joshua S Speagle. dynesty: a dynamic nested sampling package for estimating bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493(3):3132–3158, feb 2020. doi: 10.1093/mnras/staa278. URL <https://doi.org/10.1093%2Fmnras%2Fstaa278>.
- [45] Greg Ashton et al. Nested sampling for physical scientists. *Nature*, 2, 2022. doi: 10.1038/s43586-022-00121-x.

- [46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.
- [48] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [49] Ewan Cameron and Anthony Pettitt. Recursive pathways to marginal likelihood estimation with prior-sensitivity analysis. *Statistical Science*, 29(3):397–419, 2014. ISSN 08834237, 21688745. URL <http://www.jstor.org/stable/43288518>.
- [50] L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.
- [51] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O’Neil. Fast Direct Methods for Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:252, June 2015. doi: 10.1109/TPAMI.2015.2448083.
- [52] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, dec 1997. ISSN 0098-3500. doi: 10.1145/279232.279236.
- [53] N. Aghanim et al. Planck 2018 results. V. CMB power spectra and likelihoods. 2019.
- [54] N. Aghanim et al. Planck 2018 results. VIII. Gravitational lensing. 2018.
- [55] Y. Akrami et al. Planck 2018 results. X. Constraints on inflation. *Astron. Astrophys.*, 641:A10, 2020. doi: 10.1051/0004-6361/201833887.
- [56] N. Aghanim et al. Planck 2018 results. VI. Cosmological parameters. *Astron. Astrophys.*, 641:A6, 2020. doi: 10.1051/0004-6361/201833910. [Erratum: Astron.Astrophys. 652, C4 (2021)].