

---

# Adaptive Cholesky Gaussian Processes

---

Simon Bartels<sup>1,2</sup> Kristoffer Stensbo-Smidt<sup>2</sup> Pablo Moreno-Muñoz<sup>2</sup> Wouter Boomsma<sup>1</sup> Jes Frelsen<sup>2</sup>  
Søren Hauberg<sup>2</sup>

## Abstract

We present a method to fit exact Gaussian process models to large datasets by considering only a subset of the data. Our approach is novel in that the size of the subset is selected on the fly during exact inference with little computational overhead. From an empirical observation that the log-marginal likelihood often exhibits a linear trend once a sufficient subset of a dataset has been observed, we conclude that many large datasets contain redundant information that only slightly affects the posterior. Based on this, we provide probabilistic bounds on the full model evidence that can identify such subsets. Remarkably, these bounds are largely composed of terms that appear in intermediate steps of the standard Cholesky decomposition, allowing us to modify the algorithm to adaptively stop the decomposition once enough data have been observed. Empirically, we show that our method can be directly plugged into well-known inference schemes to fit exact Gaussian process models to large datasets.

## 1. Introduction

It has been observed (Chalupka et al., 2013) that the subset-of-data approximation can be a hard-to-beat baseline for approximate Gaussian process inference. However, the question of how to choose the subset is non-trivial to answer. Here we make an attempt.

The key computational challenge in Gaussian process regression is to evaluate the log-marginal likelihood of the  $N$  observed data points, which is known to have cubic complexity (Rasmussen & Williams, 2006). In order to arrive at a computationally less expensive approximation of this log-marginal likelihood, we first empirically study its behavior as we increase the number of observations. Figure 1

<sup>1</sup>Datalogisk Institut, University of Copenhagen, København, DENMARK <sup>2</sup>DTU Compute, Technical University of Denmark, Kgs. Lyngby, DENMARK. Correspondence to: Simon Bartels <bartels@di.ku.dk>.

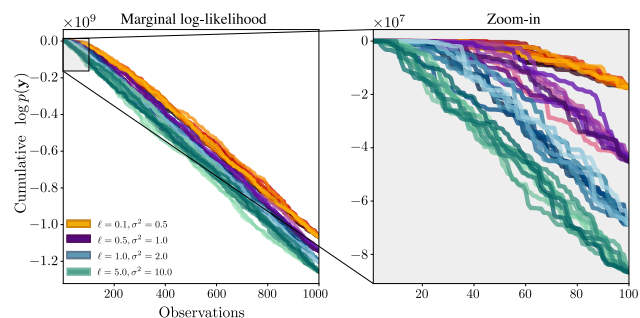


Figure 1: The figure shows total log-marginal likelihood as a function of the size of the training set. The different colors correspond to different Gaussian process models, using the SE kernel, Equation (14), with length scale  $\ell$  and amplitude  $\sigma^2$ . The inputs were sampled from a zero-mean Gaussian with variance 100. The targets have mean  $-2.5$  and variance 25. It can be seen that the log-marginal likelihood exhibits a linear trend after sufficiently many observations have been processed.

show this progression for a variety of models. We elaborate on this figure later, but for now note that after a certain number of observations, which differ between models, the log-marginal likelihood starts to progress with a linear trend. This suggests that we may leverage this near-linearity to estimate the log-marginal likelihood of the full dataset after having seen only a subset of the data. However, as the point-of-linearity differs between models, this must be estimated on-the-fly to keep computations tractable.

In this paper, we approach the problem from a probabilistic numerics perspective (Hennig et al., 2015). By treating the dataset as a collection of independent and identically distributed random variables, we provide expected upper and lower bounds on the log-marginal likelihood, which become tight when the above-mentioned linear trend arises. We provide a particularly efficient algorithm for computing the bounds that leverage the intermediate computations performed by the Cholesky decomposition that is commonly used for evaluating the log-marginal likelihood. The bounds are therefore practically free to evaluate. We further show that these bounds allow us to predict *when* the linear trend determines the full-data log-marginal likelihood, such that we can phrase an *optimal stopping problem* to determine a suitable subset of the data for a particular model.

We call our method *Adaptive Cholesky Gaussian Process* (ACGP). Our approach has a complexity of  $\mathcal{O}(M^3)$ , where  $M$  is the processed subset-size, inducing an overhead of  $\mathcal{O}(M)$  to the Cholesky. The main difference to previous work is that our algorithm does *not necessarily* look at the whole dataset.

## 2. Background

### 2.1. Notation

We use a PYTHON-inspired index notation, abbreviating for example  $[y_1, \dots, y_{n-1}]^\top$  as  $\mathbf{y}_{:n}$ —observe that the indexing starts at 1. With `Diag` we define the operator that sets all off-diagonal entries of a matrix to 0.

### 2.2. Gaussian Process Regression

We start by briefly reviewing Gaussian process (GP) models and how they are trained. A GP is a collection of random variables defined in terms of a mean function,  $m(\mathbf{x})$ , and a covariance function or *kernel*,  $k(\mathbf{x}, \mathbf{x}') = \text{cov}(f(\mathbf{x}), f(\mathbf{x}'))$ . We denote such a GP as

$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

We consider the training dataset  $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$  with inputs  $\mathbf{x}_n \in \mathbb{R}^p$  and outputs  $y_n \in \mathbb{R}$ . The inputs are collected in the matrix  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times p}$  and their corresponding function values are denoted as  $\mathbf{f} = f(\mathbf{X}) \in \mathbb{R}^N$ . In a similar manner, we denote the test inputs  $\mathbf{X}_*$  and their corresponding function values  $\mathbf{f}_*$ . The joint distribution over the training and test outputs, assuming a zero mean function,  $m(\mathbf{X}) = \mathbf{0}$ , is then

$$p(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\begin{array}{c} \mathbf{f} \\ \mathbf{f}_* \end{array} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\text{ff}} & \mathbf{K}_{\text{f}*} \\ \mathbf{K}_{*\text{f}} & \mathbf{K}_{**} \end{bmatrix}\right), \quad (1)$$

where we have used the shorthand notation  $\mathbf{K}_{\text{ff}} = k(\mathbf{X}, \mathbf{X})$  and similarly for the other covariance matrices. As usual, we will consider the outputs  $\mathbf{y}$  as being noise-corrupted versions of the function values  $\mathbf{f}$ , and we shall parameterize this corruption through the likelihood function  $p(\mathbf{y} | \mathbf{f})$ , which for regression tasks is typically assumed to be Gaussian,  $p(\mathbf{y} | \mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I})$ . For such a model, the posterior process can be computed in closed-form:

$$\begin{aligned} p(\mathbf{f}_* | \mathbf{y}) &= \mathcal{N}(\mathbf{m}_*, \mathbf{S}_*) \\ \mathbf{m}_* &= \mathbf{K}_{*\text{f}} \mathbf{K}^{-1} \mathbf{y} \\ \mathbf{S}_* &= \mathbf{K}_{**} - \mathbf{K}_{*\text{f}} \mathbf{K}^{-1} \mathbf{K}_{\text{f}*}. \end{aligned}$$

where  $\mathbf{K} := \mathbf{K}_{\text{ff}} + \sigma^2 \mathbf{I}$ .

By marginalizing over the function values of the likelihood distribution, we obtain the marginal likelihood,  $p(\mathbf{y})$ ,

which is typically used for comparing the performance of models in the Bayesian framework:

$$p(\mathbf{y}) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f}) d\mathbf{f}. \quad (2)$$

While this integral is not tractable in general, it does have a closed-form solution for Gaussian process regression. Given the GP prior,  $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{ff}})$ , and the Gaussian likelihood, the log-marginal likelihood distribution can be found to be

$$\log p(\mathbf{y}) = -\frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} - c, \quad (3)$$

where  $c = (N/2) \log 2\pi$ .

### 2.3. Background on the Cholesky decomposition

Inverting covariance matrices such as  $\mathbf{K}$  is a slow and numerically unstable procedure. Therefore, in practice, one typically leverages the Cholesky decomposition of the covariance matrices to compute the inverses. One contribution of our work is to show that the Cholesky decomposition can be stopped early to produce a highly efficient numerical scheme for Gaussian process regression.

The Cholesky decomposition of a symmetric and positive definite matrix  $\mathbf{K}$  is the unique, lower<sup>1</sup> triangular matrix  $\mathbf{L}$  such that  $\mathbf{K} = \mathbf{L}\mathbf{L}^\top$  (Golub & Van Loan, 2013, Theorem 4.2.7). The advantage of having such a decomposition is that inversion with triangular matrices amounts to Gaussian elimination. There are different options to compute  $\mathbf{L}$ . The Cholesky of a  $1 \times 1$  matrix is the square root of the scalar. For larger matrices,

$$\text{chol}[\mathbf{K}] = \begin{bmatrix} \text{chol}[\mathbf{K}_{:,s;s}] & \mathbf{0} \\ \mathbf{T} & \text{chol}[\mathbf{K}_{s:,s} - \mathbf{T}\mathbf{T}^\top] \end{bmatrix}, \quad (4)$$

where  $\mathbf{T} := \mathbf{K}_{:,s;s} \text{chol}[\mathbf{K}_{:,s;s}]^{-\top}$  and  $s$  is any integer between 1 and the size of  $\mathbf{K}$ . Hence, extending a given Cholesky to a larger matrix requires three steps:

1. solve the linear equation system  $\mathbf{T}$ ,
2. apply the down date  $\mathbf{K}_{s:,s} - \mathbf{T}\mathbf{T}^\top$  and
3. compute the Cholesky of the down-dated matrix.

An important observation is that  $\mathbf{K}_{s:,s} - \mathbf{T}\mathbf{T}^\top$  is the posterior covariance matrix  $\mathbf{S}_* + \sigma^2 \mathbf{I}$  when considering  $\mathbf{X}_s$  as test points. We will make use of this observation in Section 3.5. The log-determinant of  $\mathbf{K}$  can be obtained from the Cholesky using  $\log |\mathbf{K}| = 2 \sum_{n=1}^N \log L_{nn}$ . A similar recursive relationship exists between the quadratic form  $\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}$  and  $\mathbf{L}^{-1} \mathbf{y}$  (see, e.g., Rasmussen & Williams (2006)).

<sup>1</sup>Equivalently, one can define  $\mathbf{L}$  to be upper triangular such that  $\mathbf{K} = \mathbf{L}^\top \mathbf{L}$ .

## 2.4. Related work

For a large number of observations  $N$ , the log-marginal likelihood becomes computationally demanding to evaluate as the computation of the Cholesky requires  $\mathcal{O}(N^3)$  operations and  $\mathcal{O}(N^2)$  storage. Therefore much work has gone into tractable approximations to the log-marginal likelihood, which is also the focus of our work. Arguably, the most popular approximation methods for GPs are inducing point methods (Quiñonero-Candela & Rasmussen, 2005; Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013; 2015; 2017; Shi et al., 2020), where the dataset is approximated through a set of pseudo-data points (inducing points), summarizing information from nearby data. Other approaches involve building approximations to  $\mathbf{K}$  (Fine & Scheinberg, 2001; Harbrecht et al., 2012; Wilson & Nickisch, 2015; Rudi et al., 2017; Wang et al., 2019) or aggregating of distributed local approximations (Gal et al., 2014; Deisenroth & Ng, 2015). One may also consider separately the approximation of the quadratic form via linear solvers such as conjugate gradients (Hestenes & Stiefel, 1952; Cutajar et al., 2016) and the approximation of the log-determinant (Fitzsimons et al., 2017a;b; Dong et al., 2017). Another line of research is scaling the hardware (Nguyen et al., 2019).

The work closest in spirit to the present paper is Artemev et al. (2021), who also propose lower and upper bounds on quadratic form and log-determinant. There are a number of differences, however. Their bound relies on the method of conjugate gradients where we work directly with the Cholesky decomposition. Furthermore, while their bounds are deterministic, ours are probabilistic, which can make them tighter in certain cases, as they do not need to hold for all worst-case scenarios.

All above referenced approaches have computational complexity at least  $\mathcal{O}(N)$ . However, the size of a dataset is seldom a particularly chosen value but rather the ad-hoc end of the sampling procedure. The dependence on the dataset size implies that more available data allows to spend less computational budget even though more data might not be helpful. This is the main motivation for our work to derive an algorithm with sub-linear complexity.

## 3. Methodology

In the following, we will sketch our method. Our main goal is to convey the idea and intuition. To this end, we use suggestive notation. We refer the reader to the appendix for a more thorough and formal treatment.

### 3.1. Intuition on the linear extrapolation

The marginal likelihood is typically presented as a joint distribution, but, using Bayes rule, one can also view it from a

cumulative perspective as the sum of log-conditionals:

$$\log p(\mathbf{y}) = \sum_{n=1}^N \log p(y_n | \mathbf{y}_{:n}). \quad (5)$$

With this equation in hand, the phenomena in Figure 1 becomes much clearer. The figure shows the value of Equation (5) for an increasing number of observations  $N$ . When the plot exhibits a linear trend, it is because the summands  $\log p(y_n | \mathbf{y}_{:n})$  become approximately constant, implying that the model is not gaining additional knowledge. In other words, new outputs are conditionally independent given the output observations seen so far.

The key problem addressed in this paper is how to estimate the full marginal likelihood,  $p(\mathbf{y})$ , from only a subset of  $M$  observations. The cumulative view of the log-marginal likelihood in Equation (5) is our starting point. In particular, we will provide bounds, which are functions of seen observations, on the estimate of the full marginal likelihood. These bounds will allow us to decide, on the fly, when we have seen enough observations to accurately estimate the full marginal likelihood.

### 3.2. Stopping strategy

Suppose that we have processed  $M$  data points with  $(N - M)$  data points still in the queue. We can then decompose Equation (5) into a sum of terms which have already been computed and a remaining sum

$$\log p(\mathbf{y}) = \underbrace{\sum_{n=1}^M \log p(y_n | \mathbf{y}_{:n})}_{A: \text{processed}} + \underbrace{\sum_{n=M+1}^N \log p(y_n | \mathbf{y}_{:n})}_{B: \text{remaining}}.$$

Recall that we consider the  $\mathbf{x}_i, y_i$  as independent and identically distributed random variables. Hence, we could estimate  $B$  as  $A(N - M)/M$ . Yet this estimator is biased, since  $(\mathbf{x}_{M+1}, y_{M+1}), \dots, (\mathbf{x}_N, y_N)$  interact non-linearly through the kernel function. Instead, we will derive unbiased lower and upper bounds,  $\mathcal{L}$  and  $\mathcal{U}$ . To obtain unbiased estimates, we use the last- $m$  processed points, such that conditioned on the points up to  $s := M - m$ ,  $\log p(\mathbf{y})$  can be sandwiched,

$$\mathbb{E}[\mathcal{L} | \mathbf{X}_{:s}, \mathbf{y}_{:s}] \leq A + \mathbb{E}[B | \mathbf{X}_{:s}, \mathbf{y}_{:s}] \leq \mathbb{E}[\mathcal{U} | \mathbf{X}_{:s}, \mathbf{y}_{:s}],$$

and the observations from  $s$  to  $M$  can be used to estimate  $\mathcal{L}$  and  $\mathcal{U}$ . We can then detect when the upper and lower bounds are sufficiently near each other, and stop computations early when the approximation is sufficiently good. More precisely, given a desired relative error  $r$ , we stop when  $\mathcal{U}$  and  $\mathcal{L}$  have the same sign and when

$$\frac{\mathcal{U} - \mathcal{L}}{2 \min(|\mathcal{U}|, |\mathcal{L}|)} < r. \quad (6)$$

If the bounds hold, then the estimator  $\frac{1}{2}\mathcal{L} + \frac{1}{2}\mathcal{U}$  achieves the desired relative error (Lemma 19 in appendix). This is in contrast to other approximations, where one specifies a computational budget, rather than a desired accuracy.

The stopping conditions can be checked before or after Step 3 of the Cholesky decomposition (Section 2.3). Here, we explore the former option since Step 3 is the bottleneck due to being less parallelizable than the other steps. As a consequence, the term  $A$  is available only until index  $s$  and our bounds estimate also the partially processed points.

### 3.3. Bounds

From Equation (3), we see that the log-marginal likelihood decomposes into the log-determinant of the kernel matrix, a quadratic term, and a constant term. In the following we present upper and lower bounds for both the log-determinant ( $\mathcal{U}_D$  and  $\mathcal{L}_D$ , respectively) and the quadratic term ( $\mathcal{U}_Q$  and  $\mathcal{L}_Q$ ).

We will need the posterior equations from Section 2 but for the observations, that is  $p(\mathbf{y}_* | \mathbf{y})$ , and we will need them as functions of  $\mathbf{X}_*$ . With a slight abuse of notation, denote  $\mathbf{m}_*^{(s)}(\cdot)$  for the posterior mean function conditioned on  $\mathbf{y}_{:s}$  and, analogously, define  $\Sigma_*^{(s)}(\cdot, \cdot)$  as the posterior covariance function conditioned on  $\mathbf{y}_{:s}$ . This allows to rewrite Equation (5) as

$$\begin{aligned} \log p(\mathbf{y}) \propto & \sum_{n=1}^N \log \Sigma_*^{(n-1)}(\mathbf{x}_n, \mathbf{x}_n) \\ & + \sum_{n=1}^N \frac{(y_n - \mathbf{m}_*^{(n-1)}(\mathbf{x}_n))^2}{\Sigma_*^{(n-1)}(\mathbf{x}_n, \mathbf{x}_n)} \end{aligned} \quad (7)$$

which reveals that the log-determinant can be written as a sum of posterior variances and the quadratic form has an expression as normalized square errors.

Other key ingredients for our bounds are estimates for average posterior variance and average correlation. Therefore define the short-hands

$$\begin{aligned} \mathbf{V} &:= \text{Diag} \left[ \Sigma_*^{(s)}(\mathbf{X}_{s:M}, \mathbf{X}_{s:M}) \right] \quad \text{and} \\ \mathbf{C} &:= \sum_{i=1}^{\frac{M}{2}} \Sigma_*^{(s)}(\mathbf{x}_{s+2i}, \mathbf{x}_{s+2i-1}) \mathbf{e}_{2i} \mathbf{e}_{2i}^\top, \end{aligned}$$

where  $\mathbf{e}_j \in \mathbb{R}^m$  is the  $j$ -th standard basis vector. The matrix  $\mathbf{V}$  is simply the diagonal of the posterior covariance matrix  $\Sigma_*$ . The matrix  $\mathbf{C}$  consists of every *second* entry of the first off-diagonal of  $\Sigma_*$ . These elements are placed on the diagonal with every second element being 0. The reason for taking every second element is of theoretical nature—see Remark 5 in the appendix. In practice we use the full off-diagonal.

#### 3.3.1. BOUNDS ON THE LOG-DETERMINANT

Both bounds, lower and upper, use that  $\log |\mathbf{K}| = \log |\mathbf{K}_{:,s,:s}| + \log |\Sigma_*^{(s)}(\mathbf{X}_{s:})|$  which follows from the matrix-determinant lemma. The first term is available from the already processed datapoints. It is the second addend that needs to be estimated, which we approach from the perspective of Equation (7). It is well-established that, for a fixed input, more observations decrease the posterior variance, and this decrease cannot cross the threshold  $\sigma^2$  (Rasmussen & Williams, 2006, Question 2.9.4). This remains true when taking the expectation over the input. Hence, the average of the posterior variances for inputs  $\mathbf{X}_{s:M}$  is with high probability an overestimate of the average posterior variance for inputs with higher index. This motivates our upper bound on the log-determinant:

$$\begin{aligned} \mathcal{U}_D &= \log |\mathbf{K}_{:,s,:s}| + (N - s)\mu_D \\ \mu_D &:= \frac{1}{m} \mathbf{1}^\top \log(\mathbf{V}) \mathbf{1} \\ &\quad \text{// average log posterior variance} \end{aligned} \quad (8)$$

To arrive at the lower bound on the log-determinant, we need an argument about how fast the average posterior variance could decrease which is governed by the correlation between inputs. The variable  $\rho_D$  measures the average correlation, and we show in Theorem 9 in the appendix that this overestimates the decrease per step with high probability. Since the decrease cannot exceed  $\sigma^2$ , we introduce  $\psi_D$  to denote the step which would cross this threshold.

$$\begin{aligned} \mathcal{L}_D &= \log |\mathbf{K}_{:,s,:s}| + (\psi_D - s) \left( \mu_D - \frac{\psi_D - s - 1}{2\sigma^4} \rho_D \right) \\ &\quad + (N - \psi_D) \log \sigma^2. \end{aligned} \quad (9)$$

$$\begin{aligned} \rho_D &:= \frac{2}{m} \mathbf{1}^\top \mathbf{C} \mathbf{C} \mathbf{1} \\ &\quad \text{// average square cross-correlation} \\ \psi_D &:= \max(N, \lfloor s - 1 + \frac{2}{\rho_D} (\mu_D - \log \sigma^2) \rfloor) \\ &\quad \text{// number of steps } \mu_D \text{ can decrease by } \rho \text{ before hitting } \log \sigma^2. \end{aligned} \quad (10)$$

Both bounds collapse to the exact solution when  $s = N$ . The bounds are close when the average correlation between inputs,  $\rho_D$ , is small. This occurs for example when the average variance is close to  $\sigma^2$  since the variance is an upper bound to the correlation, an example being shown in Figure 2. Another case where  $\rho_D$  is small is when points are not correlated to begin with.

#### 3.3.2. BOUNDS ON THE QUADRATIC TERM

Denote with  $\mathbf{e}_* := \mathbf{y}_s - \mathbf{m}_*^{(s)}(\mathbf{X}_{s:})$  the prediction error, when considering the first  $s$  points as training set

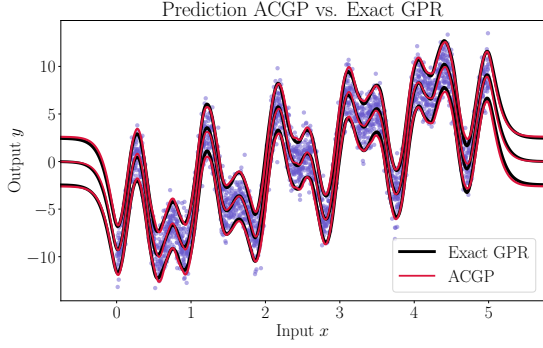


Figure 2: Prediction performance of ACGP vs. exact GP regression. The 5000 inputs are distributed standard normal multiplied by a factor 5. The targets stem from a periodic function with linear trend corrupted by independent zero-mean Gaussian noise with variance 2.25. The kernel is a Matérn $\frac{5}{2}$  with parameters determined by marginal likelihood maximization. ACGP stopped *early* using only a subset of 1536 data points, saving more than 50% of the computational cost but getting similar results as in the exact case.

and the remaining inputs as test set. Analogous to the bounds on the log-determinant, one can show with the matrix inversion lemma that  $\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} = \mathbf{y}_{:s}^\top \mathbf{K}_{:,s}^{-1} \mathbf{y}_{:s} + \mathbf{e}_*^\top \left( \boldsymbol{\Sigma}_*^{(s)}(\mathbf{X}_{s:M}) \right)^{-1} \mathbf{e}_*$ . Again, the first term will turn out to be already computed. With a slight abuse of notation let  $\mathbf{e}_* := \mathbf{y}_{s:M} - \mathbf{m}_*^{(s)}(\mathbf{X}_{s:M})$ , that is, we consider only the first  $m$  entries. Our lower bound arises from another well-known lower bound:  $\mathbf{a}^\top \mathbf{A}^{-1} \mathbf{a} \geq 2\mathbf{a}^\top \mathbf{b} - \mathbf{b}^\top \mathbf{A} \mathbf{b}$  for all  $\mathbf{b}$  (see for example Kim & Teh (2018); Artemev et al. (2021)). We choose  $\mathbf{b} := \alpha \mathbf{1}$  where  $\alpha$  is chosen to maximize the bound. The result, after some cancellations, is the following lower bound on the quadratic term:

$$\begin{aligned} \mathcal{L}_Q &= \mathbf{y}_{:s}^\top \mathbf{K}_{:,s}^{-1} \mathbf{y}_{:s} + (N-s)\alpha(2\mu_Q - \alpha\rho_Q) \quad (11) \\ \mu_Q &:= \frac{1}{m} \mathbf{e}_*^\top \mathbf{e}_* \\ &\quad // \text{ average square error} \\ \rho_Q &:= \frac{1}{m} \mathbf{e}_*^\top \mathbf{e}_* + \frac{N-s-1}{m} \sum_{j=\frac{s+2}{2}}^{\frac{M}{2}} \mathbf{e}_{*,2j} \mathbf{e}_{*,2j-1} \mathbf{C}_{2j,2j} \end{aligned}$$

The  $\alpha$  maximizing above bound is  $\frac{\mu_Q}{\rho_Q}$ , which is the value we chose in our implementation. Note however, that  $\mathcal{L}_Q$  is an expected lower bound only if  $\alpha$  depends on variables with index smaller than  $s$ .

Our upper bound arises from the element-wise perspective of Equation (7). We assume that the expected mean square error  $(y_n - \mathbf{m}_*^{(n-1)}(\mathbf{x}_n))^2$  decreases with more observa-

tions. However, though mean square error and variance decrease, their expected ratio may increase or decrease depending on the choice of kernel, dataset and number of processed points. Using the average error calibration with a correction for the decreasing variance, we arrive at our upper bound on the quadratic term:

$$\begin{aligned} \mathcal{U}_Q &= \mathbf{y}_{:s}^\top \mathbf{K}_{:,s}^{-1} \mathbf{y}_{:s} + (N-s)(\mu'_Q + \rho'_Q) \quad (12) \\ \mu'_Q &:= \frac{1}{m} \mathbf{e}_*^\top \mathbf{V}^{-1} \mathbf{e}_* \\ &\quad // \text{ average error calibration} \\ \rho'_Q &:= \frac{N-s-1}{m} \frac{1}{\sigma^4} \mathbf{e}_*^\top \mathbf{C} \mathbf{V}^{-1} \mathbf{C} \mathbf{e}_* \\ &\quad // \text{ average increase in error calibration} \end{aligned}$$

In our implementation we use a slightly different upper bound. The estimate of the possible decrease of the variance uses the same technique as the lower bound for the log-determinant. Therefore we can define an analogue to Equation (10) determining the step when the variance estimate falls below  $\sigma^2$ . In our implementation, addends of the quadratic after this step are estimated by the more conservative  $\sigma^{-2}\mu_Q$ .

Again, the bounds collapse to the true quantity when  $s = N$ . The bounds will give good estimates when the average correlation between inputs, represented by the matrix  $\mathbf{C}$ , is low or when the model can predict new data well, that is, when  $\mathbf{e}_*$  is close to 0.

### 3.4. Result

**Assumption 1.** Assume that

$$\begin{aligned} &\mathbb{E} \left[ f(\mathbf{x}, \mathbf{x}') (y_j - \mathbf{m}_*^{(j-1)}(\mathbf{x}))^2 \mid \mathbf{X}_{:s}, \mathbf{y}_{:s} \right] \\ &\leq \mathbb{E} \left[ f(\mathbf{x}, \mathbf{x}') (y_j - \mathbf{m}_*^{(s)}(\mathbf{x}))^2 \mid \mathbf{X}_{:s}, \mathbf{y}_{:s} \right] \end{aligned}$$

for all  $s \in \{1, \dots, N\}$  and for all  $s < j \leq N$ , where  $f(\mathbf{x}, \mathbf{x}')$  is either  $\frac{1}{\boldsymbol{\Sigma}_*^{(s)}(\mathbf{x}, \mathbf{x})}$  or  $\frac{\boldsymbol{\Sigma}_*^{(s)}(\mathbf{x}, \mathbf{x}')^2}{\sigma^4 \boldsymbol{\Sigma}_*^{(s)}(\mathbf{x}, \mathbf{x})}$ .

This assumption is required for the upper bound on the quadratic form, and it expresses the intuition that the mean square error should decrease with more data, which, empirically, appears to be true.

**Theorem 2.** Assume that  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$  are independent and identically distributed, assume that Assumption 1 holds, and assume that  $\alpha$  in the definition of  $\mathcal{L}_Q$  depends only on  $\mathbf{x}_1, y_1, \dots, \mathbf{x}_s, y_s$ .

For any  $s \in \{1, \dots, N\}$ , the bounds defined in Equations (8), (9), (11) and (12) hold in expectation:

$$\begin{aligned} \mathbb{E}[\mathcal{L}_D \mid \mathcal{F}_s] &\leq \mathbb{E}[\log(\det[\mathbf{K}]) \mid \mathcal{F}_s] \leq \mathbb{E}[\mathcal{U}_D \mid \mathcal{F}_s] \text{ and} \\ \mathbb{E}[\mathcal{L}_Q \mid \mathcal{F}_s] &\leq \mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] \leq \mathbb{E}[\mathcal{U}_Q \mid \mathcal{F}_s], \end{aligned}$$

where  $\mathcal{F}_s$  is the  $\sigma$ -algebra generated by  $\mathbf{x}_1, y_1, \dots, \mathbf{x}_s, y_s$ .

Proof and a proof sketch can be found in the appendix.

**Theorem 3.** *Let  $r > 0$  be a desired relative error and set  $\mathcal{U} := \mathcal{U}_D + \mathcal{U}_Q$  and  $\mathcal{L} := \mathcal{L}_D + \mathcal{L}_Q$ . If the stopping conditions hold, that is,  $\text{sign}(\mathcal{U}) = \text{sign}(\mathcal{L})$  and Equation (6) is true, then  $\log p(\mathbf{y})$  can be estimated from  $\frac{1}{2}(\mathcal{U} + \mathcal{L})$  such that, under the condition  $\mathcal{L}_D \leq \log(\det[\mathbf{K}]) \leq \mathcal{U}_D$  and  $\mathcal{L}_Q \leq \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \leq \mathcal{U}_Q$ , the relative error is smaller than  $r$ , formally:*

$$\left| \frac{\log p(\mathbf{y}) - \frac{1}{2}(\mathcal{U} + \mathcal{L})}{\log p(\mathbf{y})} \right| \leq r. \quad (13)$$

The proof follows from Lemma 19 in the appendix.

Theorem 2 is a first step to obtain a probabilistic statement for Equation (13), that is, a statement of the form  $\mathbb{P}\left(\left|\frac{\log p(\mathbf{y}) - \frac{1}{2}(\mathcal{U} + \epsilon_{\mathcal{U}, \delta} + \mathcal{L} - \epsilon_{\mathcal{L}, \delta})}{\log p(\mathbf{y})}\right| > r\right) \leq \delta$ . Theoretically, we can obtain such a statement using standard concentration inequalities and a union bound over  $s$ . In practice, the error guarding constants  $\epsilon$  would render the result trivial. A union bound can be avoided using Hoeffding’s inequality for martingales (Fan et al., 2012). However, this requires to replace  $s := M - m$  by a stopping time independent of  $M$ , which we regard as future work.

### 3.5. Practical implementation

The proposed bounds turn out to be surprisingly cheap to compute. If we set the block-size of the Cholesky decomposition to be  $m$ , the matrix  $\Sigma_*^{(s)}$  is exactly the down-dated matrix in step 2 of the Cholesky decomposition algorithm outlined in Section 2.3. Similarly, the expressions for the bounds on the quadratic form appear while solving the linear equation system  $\mathbf{L}^{-1} \mathbf{y}$ . A slight modification to the Cholesky algorithm is enough to compute these bounds on the fly during the decomposition with little overhead.

Note that the definition of the bounds does not involve variables  $\mathbf{x}, y$  which have not been processed. This allows an on-the-fly construction of the kernel matrix, avoiding potentially expensive kernel function evaluations. Furthermore, it is *not* necessary to allocate  $\mathcal{O}(N^2)$  memory in advance—a user can specify a maximal amount of processed datapoints, hoping that stopping occurs before hitting that limit. We provide the pseudo-code for this modified algorithm, our key algorithmic contribution, in supplementary. Additionally, we provide a PYTHON implementation of our modified Cholesky decomposition and scripts to replicate the experiments of this paper.<sup>2</sup>

<sup>2</sup>The code is available at the following repository: [anonymized](https://github.com/anonymized)

## 4. Experiments

In this section we examine bounds and stopping strategy for ACGP. When running experiments without GPU support, all linear algebra operations are substituted for direct calls to the OPENBLAS library (Wang et al., 2013), for efficient realization of *in-place* operations. To still benefit from automatic differentiation, we used PYTORCH (Paszke et al., 2019) with a custom backward function for  $\log p(\mathbf{y})$ . We found that evaluating gradients with PYTORCH costs roughly a factor six of the standard evaluation of the log-marginal. We were able to reduce this computation time to a factor of two.

### 4.1. Performance on synthetic data

Figure 2 shows an evaluation of the performance of ACGP in an illustrative manner with synthetic data. The dataset consists of 5000 inputs distributed standard normal multiplied by a factor 5. The targets stem from a periodic function with linear trend corrupted by independent zero-mean Gaussian noise with variance 2.25. The kernel is a Matérn  $\frac{5}{2}$  with parameters determined by marginal likelihood maximization. With  $m := 256$  and  $r := 0.1$ , ACGP uses only 1536 data points, saving more than 50% of the computational cost but getting similar results as in the exact case.

### 4.2. Bound quality

In this section we examine the bounds presented in Section 3 and compare them to those proposed by Artemev et al. (2021, Lemma 2 and Lemma 3). Specifically, for the determinant we compare to their  $\mathcal{O}(N)$  upper bound (Artemev et al., 2021, Eq. 11) and  $\log(\det[\mathbf{Q}])$  as lower bound.

We set the number of inducing inputs  $M$  for CGLB to 512, 1024 and 2048. For ACGP, we define  $m := 40 \times 256 = 10240$  which is the number of cores times the default OPENBLAS block size for our machines. We compare both methods using squared exponential kernel (SE) and the Ornstein-Uhlenbeck kernel (OU)

$$k_{\text{SE}}(\mathbf{x}, \mathbf{z}) := \theta \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\ell^2}\right) \quad (14)$$

$$k_{\text{OU}}(\mathbf{x}, \mathbf{z}) := \theta \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|}{\ell}\right). \quad (15)$$

where we fix  $\sigma^2 := 10^{-3}$  and  $\theta := 1$ , and we vary  $\ell$  as  $\log \ell \in \{-1, 0, 1, 2, 3\}$ . As benchmarking datasets we use the two datasets consisting of more than 20 000 instances used by Artemev et al. (2021): `kin40k` and `protein`. We further consider two additional datasets from the UCI repository (Dua & Graff, 2019): `metro`<sup>3</sup>

<sup>3</sup><http://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume> no citation request

and `pm25`<sup>4</sup> (Liang et al., 2015). We chose these datasets in addition as they are of similar size, they are marked as regression tasks and no data points are missing.

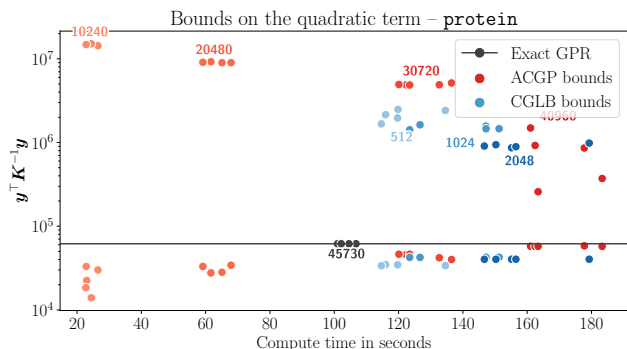


Figure 3: Upper and lower bounds on the quadratic term for ACGP and CGLB on the `protein` dataset using the OU kernel, Equation (15), with a length scale of  $\log \ell = -1$ . The black line indicates the result obtained using exact GP regression with points above it marking the upper bounds and points below marking the lower bounds. The experiment was repeated a number of times with different seeds to illustrate the variability in the computation time, shown here as multiple points. For ACGP the number near the points shows  $M$ , the size of the used subset; for CGLB it is the number of inducing inputs. The color of the points reflects these numbers to help discern the size of the subset or number of inducing inputs from the repeated experiments.

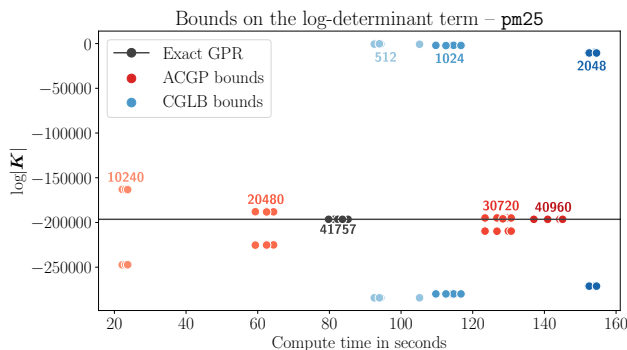


Figure 4: Upper and lower bounds on the log-determinant for ACGP and CGLB on the `pm25` dataset using the SE kernel, Equation (14), with a length scale of  $\log \ell = 0$ .

The typical picture we observe is that CGLB provides better estimates for the quadratic term, whereas ACGP is faster to identify the log-determinant. Figures 3 and 4 show typical examples. Note that Figure 3 is on a log-scale: for the quadratic form, the upper bounds tend to be less tight than the lower lower bounds. Generally, there is no clear

<sup>4</sup><http://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data>

winner—sometimes ACGP estimates both quantities faster and sometimes CGLB. For other results, see the appendix.

The reason why CGLB has more difficulties to approximate the log-determinant is that the bound involves  $\text{trace}[\mathbf{K} - \mathbf{Q}]$  where  $\mathbf{Q}$  is a low rank approximation to  $\mathbf{K}$ . If  $\mathbf{K}_{\text{ff}}$  is of high rank, the gap in trace can be large. For CGLB the time to compute the bounds is dominated by the pivoted Cholesky decomposition to select the inducing inputs. This overhead becomes irrelevant for the following hyper-parameter tuning experiments, since the selection is computed only once in the beginning. One conclusion from these experiments is to keep in mind that when high precision is required, simply computing the exact solution can be a hard to beat baseline.

### 4.3. Application in hyper-parameter tuning

We repeat the hyper-parameter tuning experiments performed by Artemev et al. (2021) using the same set-up. We use the same kernel function, a Matérn $_{\frac{3}{2}}$ , and the same optimizer: L-BFGS-B (Liu & Nocedal, 1989) with SCIPY (Virtanen et al., 2020) default parameters if not specified otherwise. Each configuration has been measured five times with a different shuffle of the dataset. All algorithms are stopped the latest after 2000 optimization steps, after 12 hours of compute time or when optimization has failed three times. For CGLB and the exact Cholesky, the L-BFGS-B convergence criterion “relative change in function value” (`ftol`) is set to 0. For ACGP, we successively decrease this tolerance as  $(2/3)^{\text{restart}+1}$  and we set the same value for  $r$ . Furthermore, we use a block size of  $m := 40 \cdot 256 = 10240$  where 256 is the standard OPENBLAS blocksize on our machines and 40 is the number of cores. Artemev et al. (2021) report their best results using  $M = 2048$  inducing inputs. We refer to *op. cit.* for a comparison to more baselines.

We explore two different computing environments. For datasets smaller than 20 000 data points, we ran our experiments on a single GPU. This is the same setup as in Artemev et al. (2021) with the difference that we use a TITAN RTX whereas they have used a TESLA V100. The results can be summarized in one paragraph: all methods converge the latest after two minutes. The time difference between methods is less than twenty seconds. Exact Gaussian process regression is fastest, more often than not. Figure 5 shows exemplary the result for the `elevators` dataset. All other results can be found in the appendix. We conclude that in an environment with significantly more processing resources than memory, approximation may just cause overhead.

For datasets larger than 20 000 datapoints, our setup differs from Artemev et al. (2021). We use only CPUs on machines where the kernel matrix still fits fully into memory.

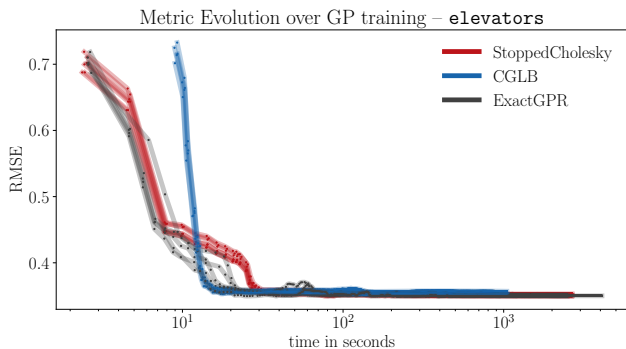


Figure 5: The evolution of the root mean square error while tuning parameters on `elevators`. When training on a GPU and the kernel matrix fits fully into memory, exact inference can be a hard-to-beat baseline.

Specifically, we used machines running Ubuntu 18.04 with 50 Gigabytes of RAM and two INTEL XEON E5-2670 v2 CPUs. The datasets are the same as in Section 4.2. On all datasets, ACGP is essentially exhibiting the same optimization behavior as the exact Gaussian process regressor, just stretched out. ACGP can provide results faster than exact optimization but may be slower in convergence as Figure 6 shows for the `protein` data-set. This observation is as expected. However, approximation can also hinder fast convergence as Figure 7 reveals on for the `metro` dataset. CGLB benefits from caching the chosen inducing inputs and reusing the solution from the last solved linear equation system. The algorithm is faster, though it often plateaus at higher objective function values. The results for `kin40k` are similar to `protein` and the results for `pm25` are similar to `metro`. These results and the evolution of the root mean square error over time can be found in the appendix. Again, when the available memory permits, the exact computation is a hard-to-beat baseline. However, the Cholesky as a standard numerical routine has been engineered over decades, whereas for the implementations of CGLB and ACGP there is opportunity for improvement.

## 5. Conclusions

In this paper we have revisited the use of Cholesky decompositions in Gaussian process regression. We have shown that the Cholesky decomposition almost computes expected lower and upper bounds on the marginal log-likelihood associated with GP regression. With fairly limited modifications to this classic matrix decomposition, we can use these bounds to stop the decomposition before all observations have been processed. This has the practical benefit that the kernel matrix  $\mathbf{K}$  does not have to be computed prior to performing the decomposition, but can rather be computed on-the-fly.

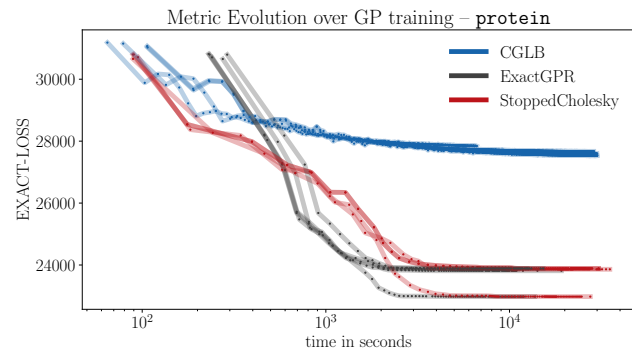


Figure 6: The evolution of the exact log marginal likelihood  $p(\mathbf{y})$  while tuning parameters for the `protein` dataset. The iteratively increasing precision may allow ACGP to provide better solutions faster than exact optimization at the price of later convergence.

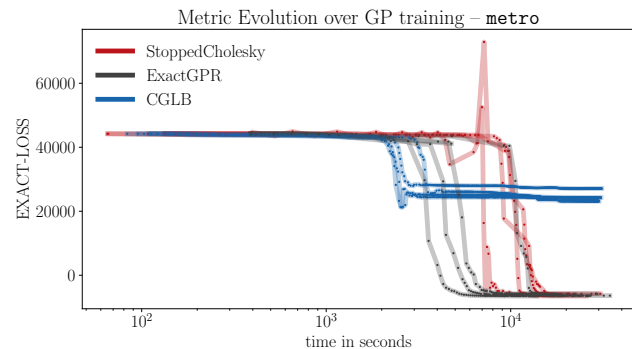


Figure 7: The evolution of the exact log marginal likelihood  $p(\mathbf{y})$  while tuning parameters for the `metro` dataset. Function evaluations with CGLB are generally the fastest at the cost of plateauing at higher objective function values.

Empirical results indicate that the approach carries significant promise, but no clear ‘winner’ can be crowned from our experiments. In general, we find that exact GP inference leads to better behaved optimization than approximations such as CGLB and inducing point methods, and that a well-optimized Cholesky implementation is surprisingly competitive in terms of performance. An advantage of our approach is that it is essentially parameter-free. The user has to specify a requested numerical accuracy and the computational demands will be scaled accordingly. Finally, we note that ACGP is complementary to much existing work, and should be seen as an addition to the GP toolbox, rather than a substitute for all existing tools.

## ACKNOWLEDGEMENTS

We are grateful for the valuable feedback of all anonymous reviewers who saw the different iterations of this article.

This work was funded in part by the Novo Nordisk Foun-



dation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606). It also received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research, innovation programme (757360), from a research grant (15334) from VILLUM FONDEN, from the Danish Ministry of Education and Science, and from Digital Pilot Hub and Skylab Digital.

## References

- Artemev, A., Burt, D. R., and van der Wilk, M. Tighter bounds on the log marginal likelihood of gaussian process regression using conjugate gradients. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 362–372, 2021.
- Bartels, S. *Probabilistic Linear Algebra*. PhD thesis, University of Tübingen, 2020.
- Bartels, S., Boomsma, W., Frellsen, J., and Garreau, D. Kernel-Matrix Determinant Estimates from stopped Cholesky Decomposition. *ArXiv e-prints*, 2107.10587, 2021. 2107.10587.
- Chalupka, K., Williams, C. K. I., and Murray, I. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14(1):333–350, 2013.
- Cutajar, K., Osborne, M., Cunningham, J., and Filippone, M. Preconditioning kernel matrices. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2529–2538, 2016.
- Deisenroth, M. and Ng, J. W. Distributed Gaussian processes. In *International Conference on Machine Learning (ICML)*, pp. 1481–1490, 2015.
- Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. G. Scalable log determinants for gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, pp. 6330–6340, 2017.
- Dua, D. and Graff, C. UCI machine learning repository, 2019. URL <http://archive.ics.uci.edu/ml>.
- Fan, X., Grama, I., and Liu, Q. Hoeffding’s inequality for supermartingales. *Stochastic Processes and their Applications*, 122(10):3545–3559, 2012.
- Fine, S. and Scheinberg, K. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- Fitzsimons, J., Cutajar, K., Osborne, M., Roberts, S., and Filippone, M. Bayesian inference of log determinants. In Elidan, G., Kersting, K., and Ihler, A. T. (eds.), *Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, August 11-15, 2017, Sydney, Australia, 2017a*.
- Fitzsimons, J., Granziol, D., Cutajar, K., Osborne, M., Filippone, M., and Roberts, S. Entropic trace estimates for log determinants. In Ceci, M., Hollmén, J., Todorovski, L., Vens, C., and Džeroski, S. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 323–338, 2017b.
- Gal, Y., Van Der Wilk, M., and Rasmussen, C. E. Distributed variational inference in sparse gaussian process regression and latent variable models. *arXiv preprint arXiv:1402.1389*, 2014.
- George, A., Heath, M. T., and Liu, J. Parallel cholesky factorization on a shared-memory multiprocessor. *Linear Algebra and its Applications*, 77:165–187, 1986.
- Golub, G. and Van Loan, C. *Matrix computations*. Johns Hopkins Univ Pr, 4 edition, 2013.
- Harbrecht, H., Peters, M., and Schneider, R. On the low-rank approximation by the pivoted Cholesky decomposition. *Applied Numerical Mathematics*, 62(4):428–440, 2012.
- Hennig, P., Osborne, M., and Girolami, M. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.
- Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 282–290, 2013.
- Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 351–360, 2015.
- Hensman, J., Durrande, N., Solin, A., et al. Variational fourier features for gaussian processes. *J. Mach. Learn. Res.*, 18(1):5537–5588, 2017.
- Hestenes, M. and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.
- Kim, H. and Teh, Y. W. Scaling up the automatic statistician: Scalable structure discovery using gaussian processes. In Storkey, A. and Perez-Cruz, F. (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 575–584, 2018.

- Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H., and Chen, S. X. Assessing beijing's  $pm_{2.5}$  pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2182): 20150257, 2015.
- Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.
- Mnih, V., Szepesvári, C., and Audibert, J.-Y. Empirical Bernstein stopping. pp. 672–679, 2008.
- Nguyen, D.-T., Filippone, M., and Michiardi, P. Exact gaussian process regression with distributed computations. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1286–1295, 2019.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.
- Quiñonero-Candela, J. and Rasmussen, C. A unifying view of sparse approximate Gaussian process regression. *J of Machine Learning Research*, 6:1939–1959, 2005.
- Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. MIT, 2006.
- Rudi, A., Carratino, L., and Rosasco, L. Falkon: An optimal large scale kernel method. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017.
- Shi, J., Titsias, M., and Mnih, A. Sparse orthogonal variational inference for gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1932–1942. PMLR, 2020.
- Snelson, E. and Ghahramani, Z. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257, 2006.
- Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32:14648–14659, 2019.
- Wang, Q., Zhang, X., Zhang, Y., and Yi, Q. AUGEM: Automatically generate high performance Dense Linear Algebra kernels on x86 CPUs. In *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2013.
- Wilson, A. and Nickisch, H. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pp. 1775–1784. PMLR, 2015.

## A. ADDITIONAL RESULTS

In this section, we report additional results for both the hyper-parameter tuning experiments (section A.1) as well as plots to show the quality of the bounds on both the log-determinant term and the quadratic term (sections A.2–A.4).

**Hyper-parameter tuning.** We repeat the experiments performed by Artemev et al. (2021). They randomly split each dataset into a training set consisting of 2/3 of examples, and a test set consisting of the remaining third. Inputs and targets are normalized to have mean 0 and variance 1 within the training set. We use the same kernel function, a Matérn $_{\frac{3}{2}}$ , and the same optimizer. For ACGP, we successively decrease the optimizer’s tolerance as  $(2/3)^{\text{restart}+1}$  and we set the same value for  $r$ . The block size is set to  $40 \cdot 256 = 10192$ . Artemev et al. (2021) report their best results using  $M = 2048$  inducing inputs.

We explore two different computing environments. For datasets smaller than 20 000 data points, we ran our experiments on a single GPU. This is the same setup as in Artemev et al. (2021) with the difference that we use a TITAN RTX whereas they have used a TESLA V100. For datasets larger than 20 000 datapoints, our setup differs from Artemev et al. (2021). We use only CPUs on machines where the kernel matrix still fits fully into memory. Specifically, we used machines running Ubuntu 18.04 with 50 Gigabytes of RAM and two INTEL XEON E5-2670 v2 CPUs.

The plots for the hyper-parameter optimization are shown in figures 8–19. Each point in the plots corresponds to one accepted optimization step for the given methods. Each point thus corresponds to a particular set of hyper-parameters during the optimization. In figures 12–19, we show the root-mean-square error, RMSE, that each methods obtains on the test set at each optimisation step. In figures 8–11, we show the log-marginal likelihood,  $\log p(\mathbf{y})$ , that an exact GP would have achieved with the specific set of hyper-parameters at each optimization step for each method.

**Bound quality plots.** For the bound quality plots, we compare ACGP to CGLB proposed by Artemev et al. (2021). For CGLB, we compute the bounds with varying number of inducing inputs  $M := \{512, 1024, 2048\}$  and measure the time it takes to compute the bounds. The three values for  $M$  are connected with a line in the plots. For ACGP, we define  $m := 10192$ . We measure the elapsed time every time a block of data points is added to the processed dataset and the bounds are recomputed.

We compare both methods using squared exponential kernel (SE) and the Ornstein-Uhlenbeck kernel (OU).

$$k_{SE}(\mathbf{x}, \mathbf{z}) := \theta \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\ell^2}\right) \tag{16}$$

$$k_{OU}(\mathbf{x}, \mathbf{z}) := \theta \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|}{\ell}\right). \tag{17}$$

where we fix  $\sigma^2 := 10^{-3}$  and  $\theta := 1$ , and we vary  $\ell$  as  $\log \ell \in \{-1, 0, 1, 2, 3\}$ .

### A.1. Additional results for hyper-parameter tuning

Table 1 summarizes the results reporting for each data-set the averages over the last points.

DATASET	ALGORITHM	RMSE	EXACT NEGATIVE MLL
protein	CGLB	$0.57 \pm 0.00$	$27620.75 \pm 35.57$
	StoppedCholesky	$0.60 \pm 0.03$	$28399.84 \pm 3299.38$
	ExactGPR	$0.56 \pm 0.00$	$23735.92 \pm 152.59$
wilson-kin40k	CGLB	$0.09 \pm 0.00$	$-16235.72 \pm 66.26$
	StoppedCholesky	$0.07 \pm 0.00$	$-20843.01 \pm 59.27$
	ExactGPR	$0.07 \pm 0.00$	$-20837.51 \pm 41.83$
metro	CGLB	$0.37 \pm 0.06$	$23974.65 \pm 1467.86$
	StoppedCholesky	$0.46 \pm 0.31$	$10402.82 \pm 23328.20$
	ExactGPR	$0.30 \pm 0.13$	$1747.71 \pm 17614.45$
pm25	CGLB	$0.44 \pm 0.04$	$22605.36 \pm 1878.74$
	StoppedCholesky	$0.55 \pm 0.09$	$24752.24 \pm 4790.72$
	ExactGPR	$0.44 \pm 0.01$	$19349.50 \pm 31.93$

Table 1: Result overview table.

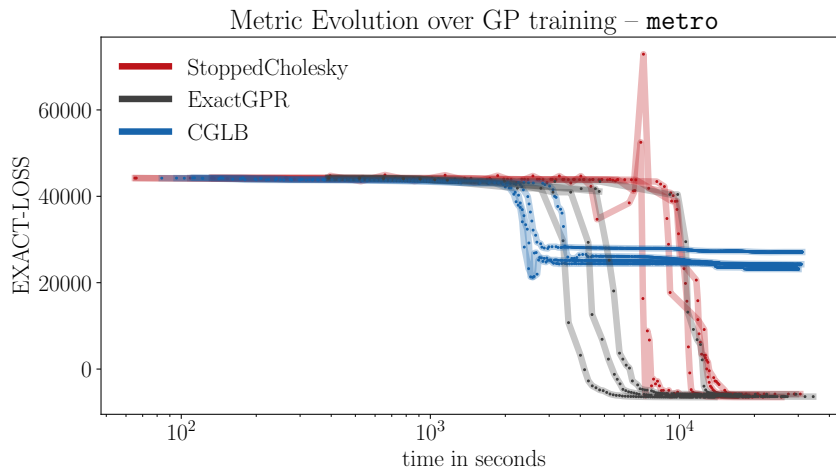


Figure 8: Log-marginal likelihood over time while optimizing hyper-parameters for the metro dataset.

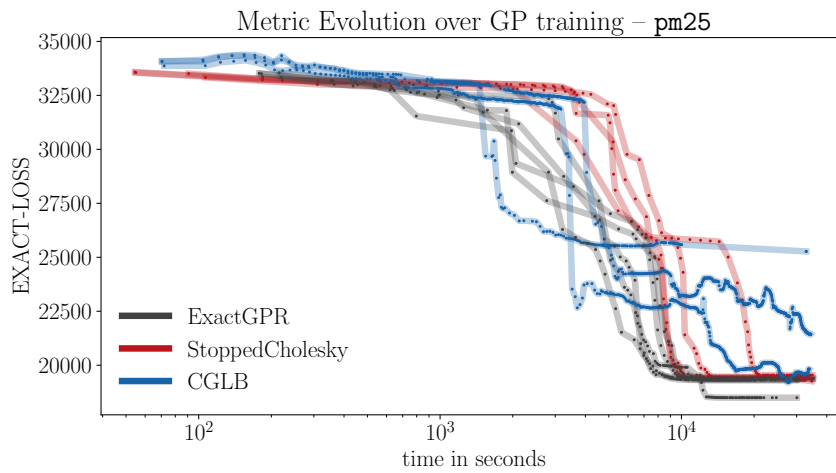


Figure 9: Log-marginal likelihood over time while optimizing hyper-parameters for the pm25 dataset.

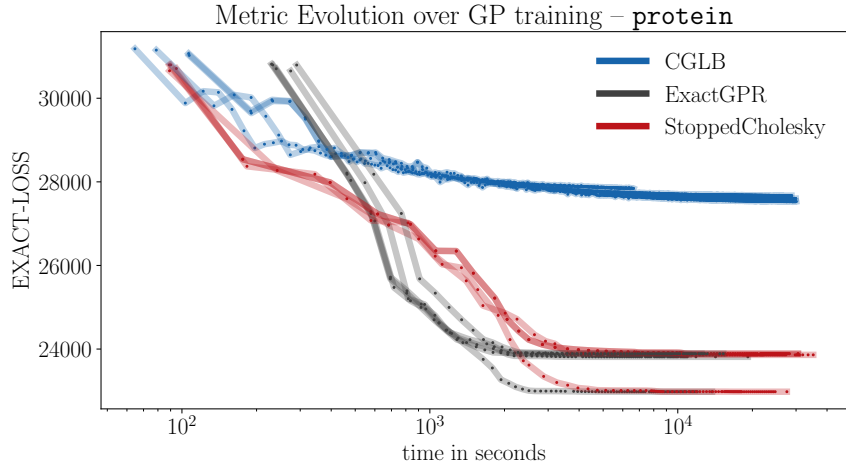


Figure 10: Log-marginal likelihood over time while optimizing hyper-parameters for the `protein` dataset.

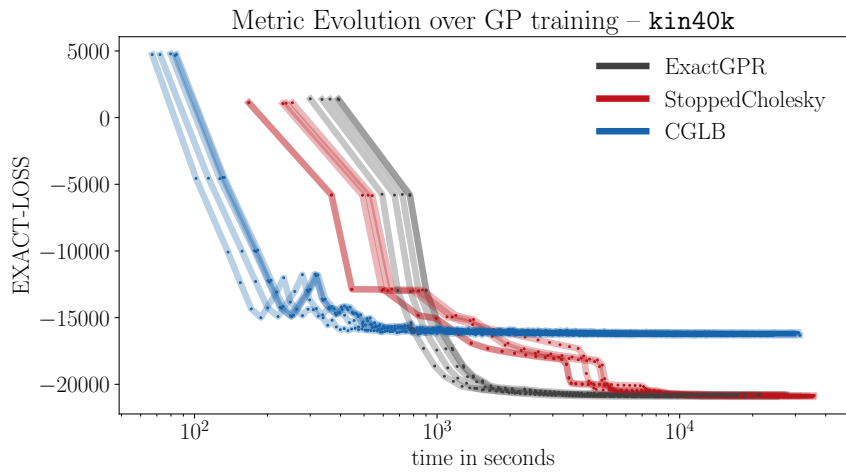


Figure 11: Log-marginal likelihood over time while optimizing hyper-parameters for the `kin40k` dataset.

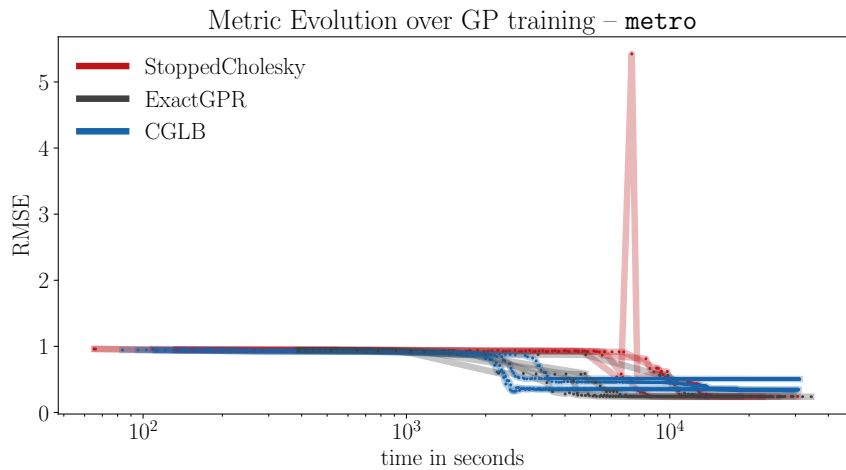


Figure 12: RMSE over time while optimizing hyper-parameters for the `metro` dataset.

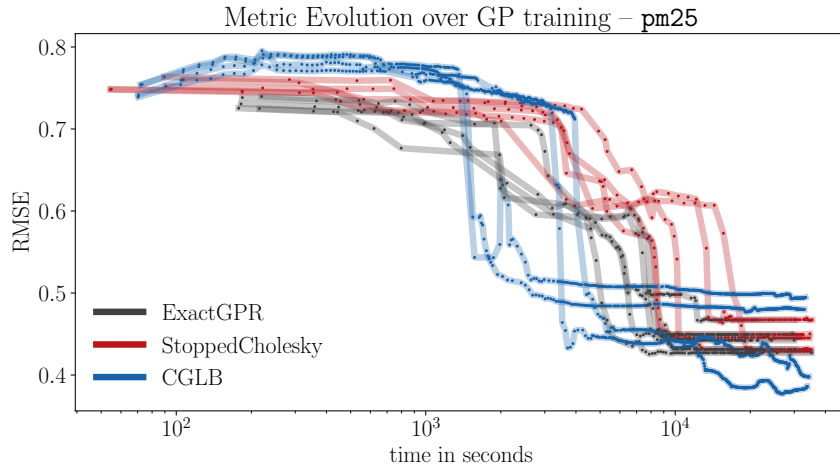


Figure 13: RMSE over time while optimizing hyper-parameters for the pm25 dataset.

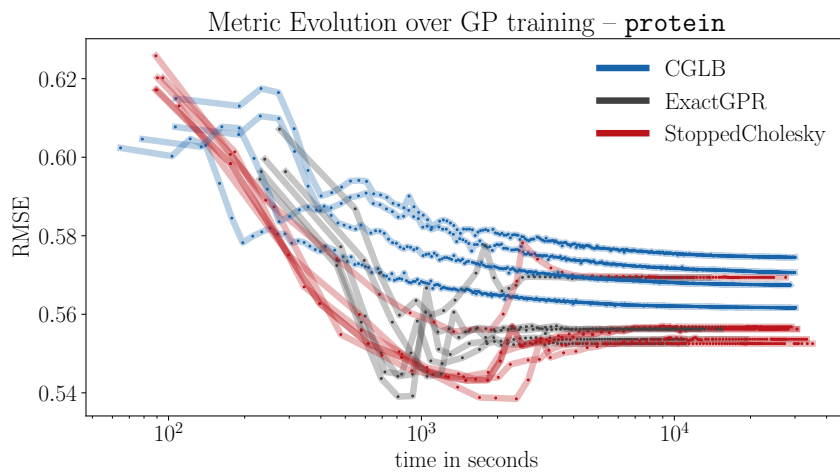


Figure 14: RMSE over time while optimizing hyper-parameters for the protein dataset.

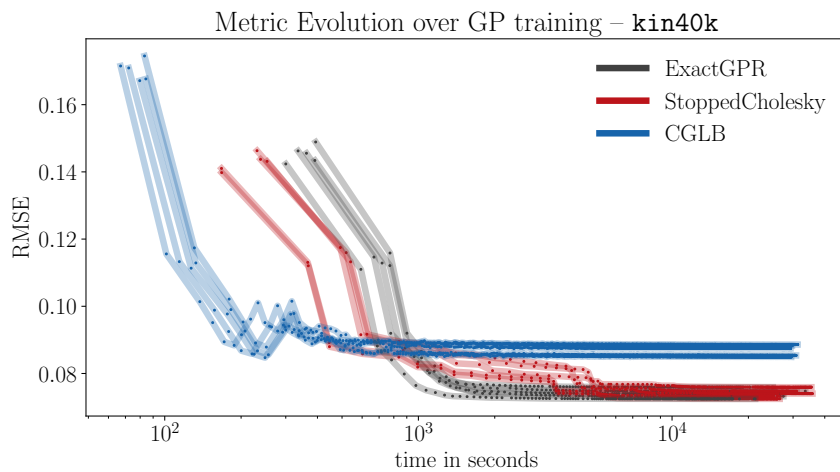


Figure 15: RMSE over time while optimizing hyper-parameters for the kin40k dataset.

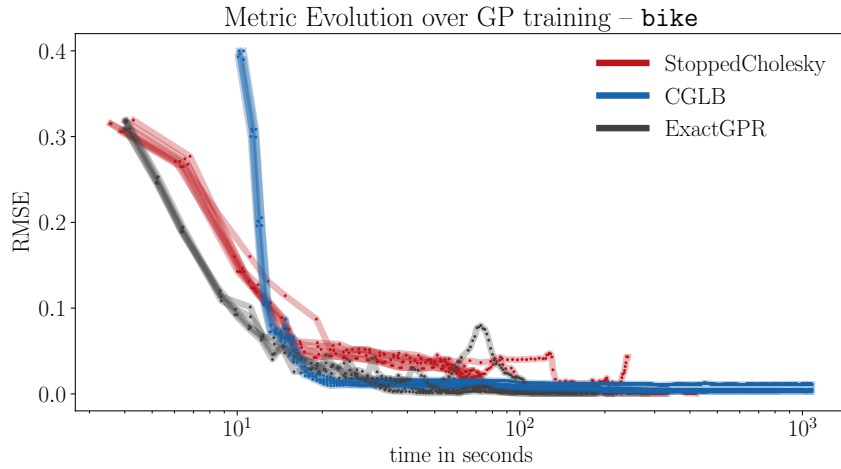


Figure 16: RMSE over time while optimizing hyper-parameters for the `bike` dataset.

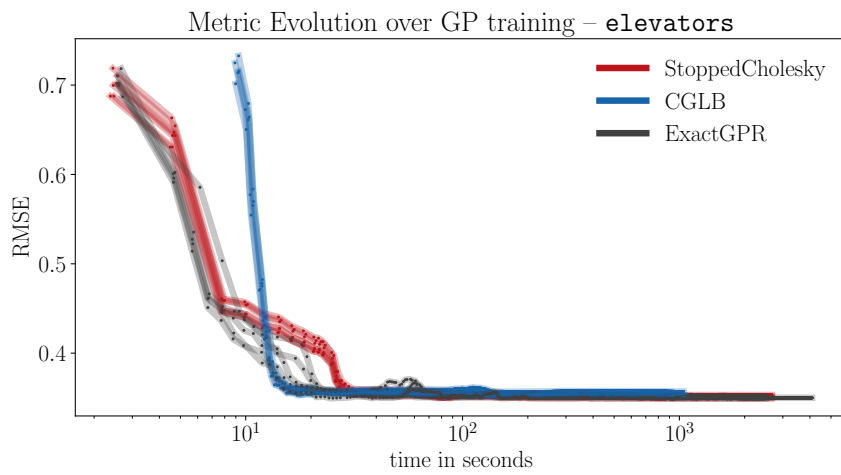


Figure 17: RMSE over time while optimizing hyper-parameters for the `elevators` dataset.

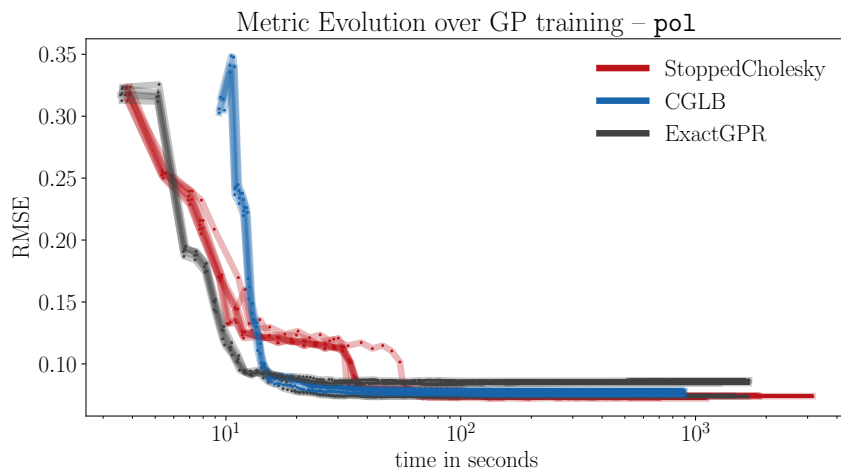


Figure 18: RMSE over time while optimizing hyper-parameters for the `pole` dataset.

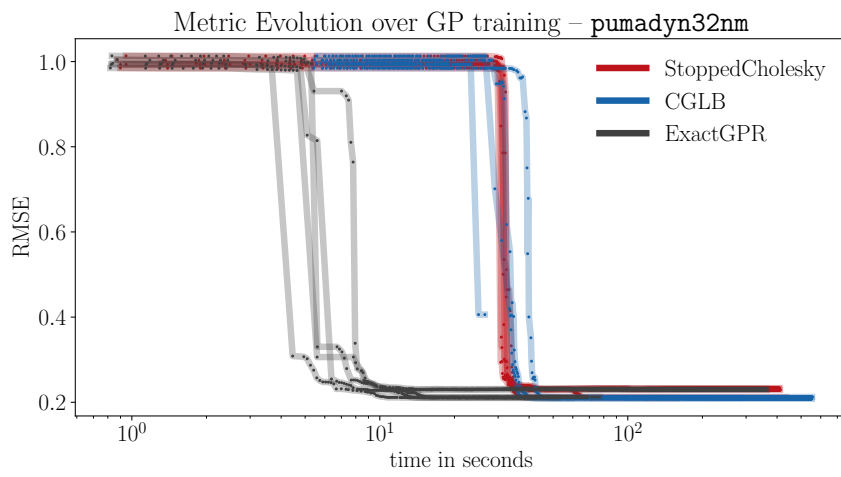


Figure 19: RMSE over time while optimizing hyper-parameters for the pumadyn32nm dataset.



A.2. Bounds for experiments on metro

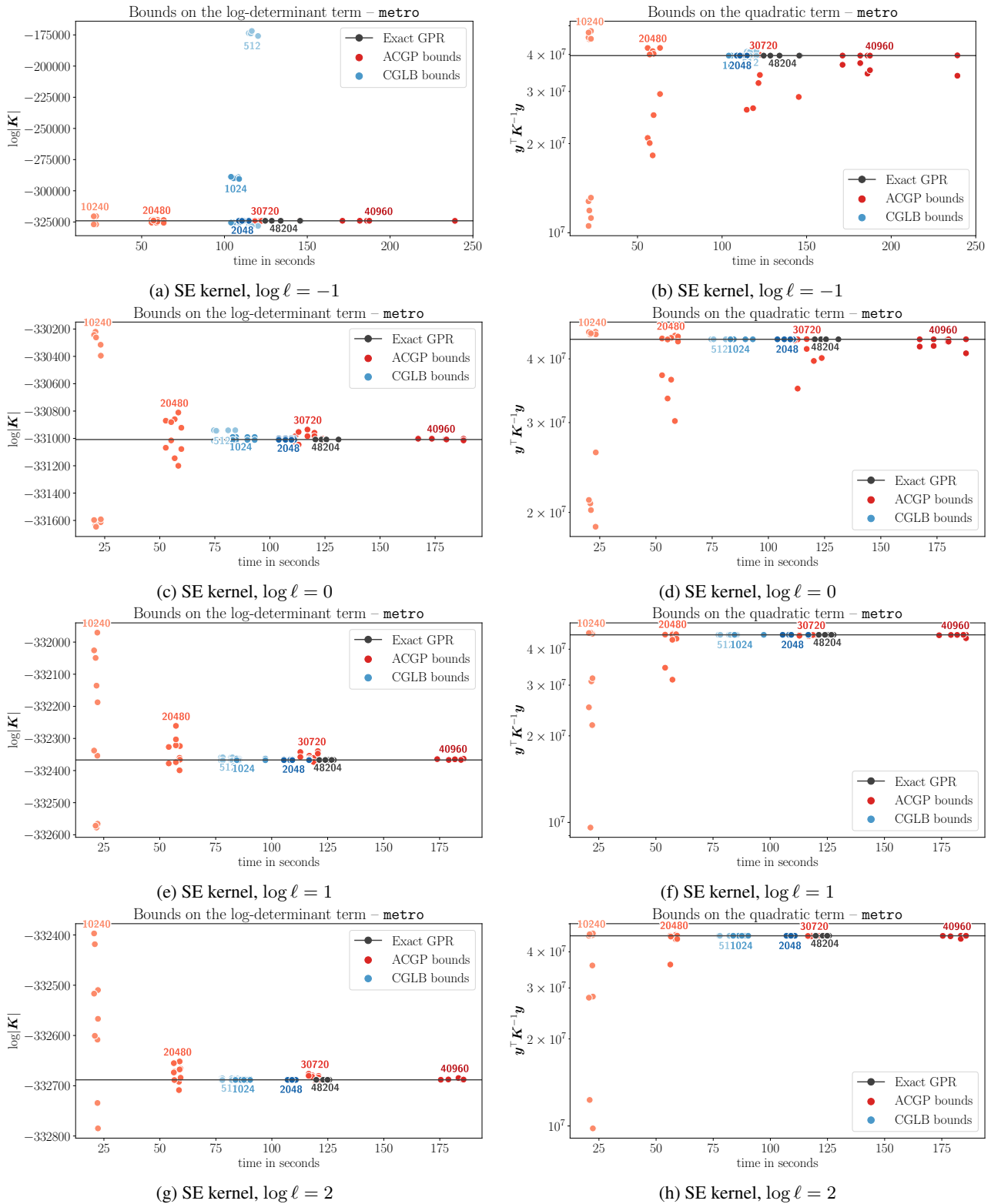


Figure 20: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the metro dataset when using a squared exponential (SE) kernel..

## Adaptive Cholesky Gaussian Processes

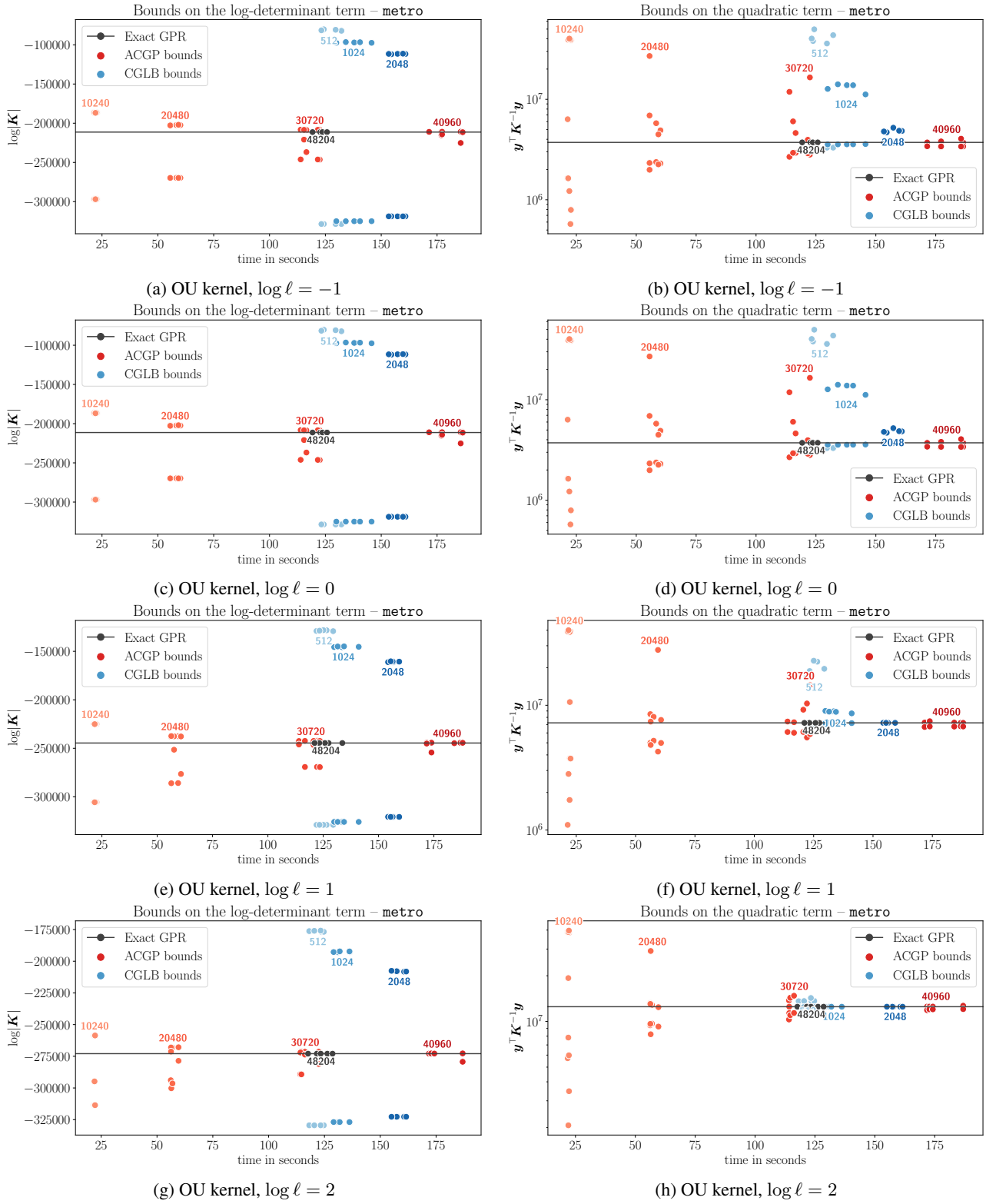


Figure 21: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the `metro` dataset using an Ornstein-Uhlenbeck (OU) kernel.

A.3. Bounds for experiments on pm25

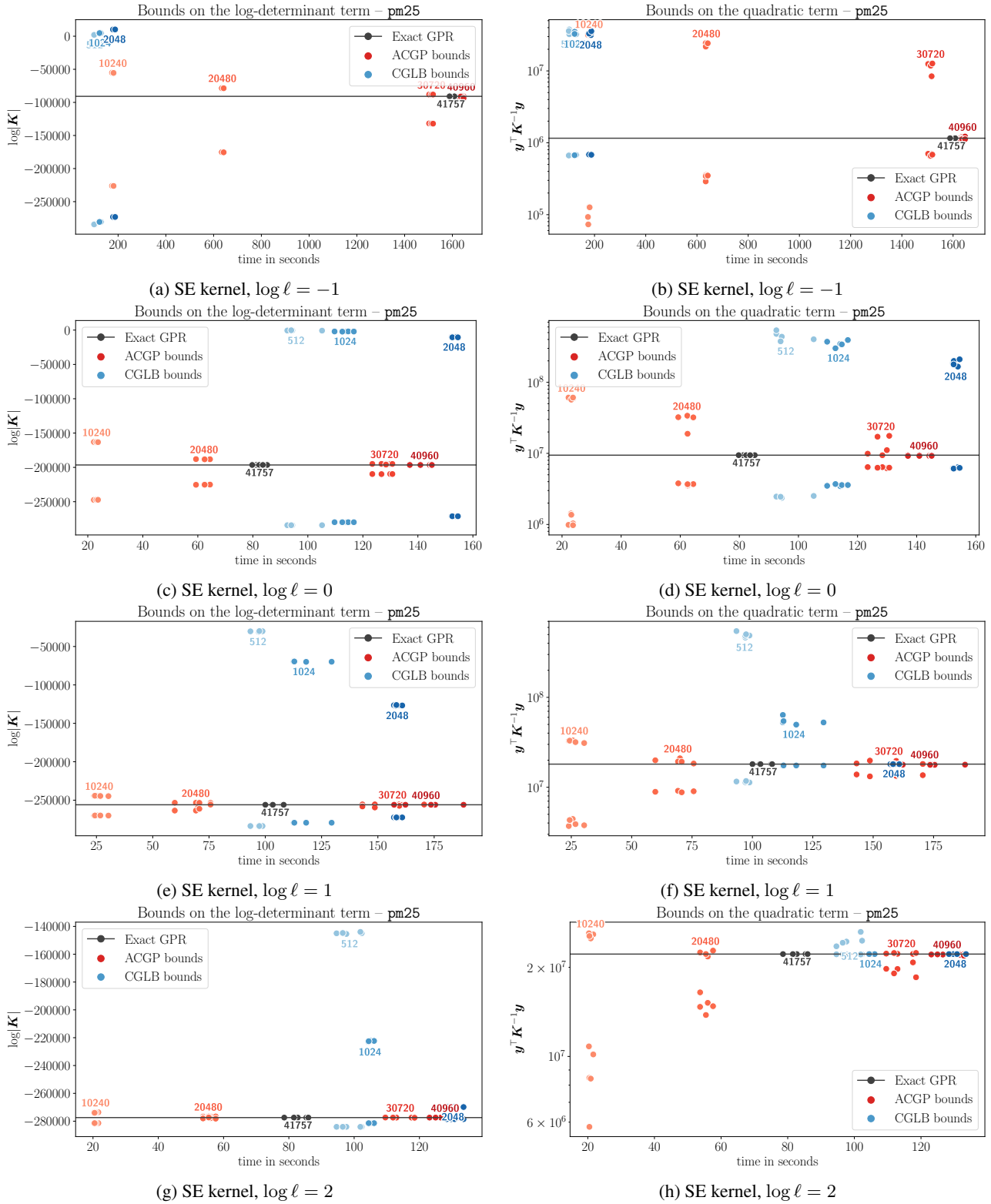


Figure 22: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the pm25 dataset.

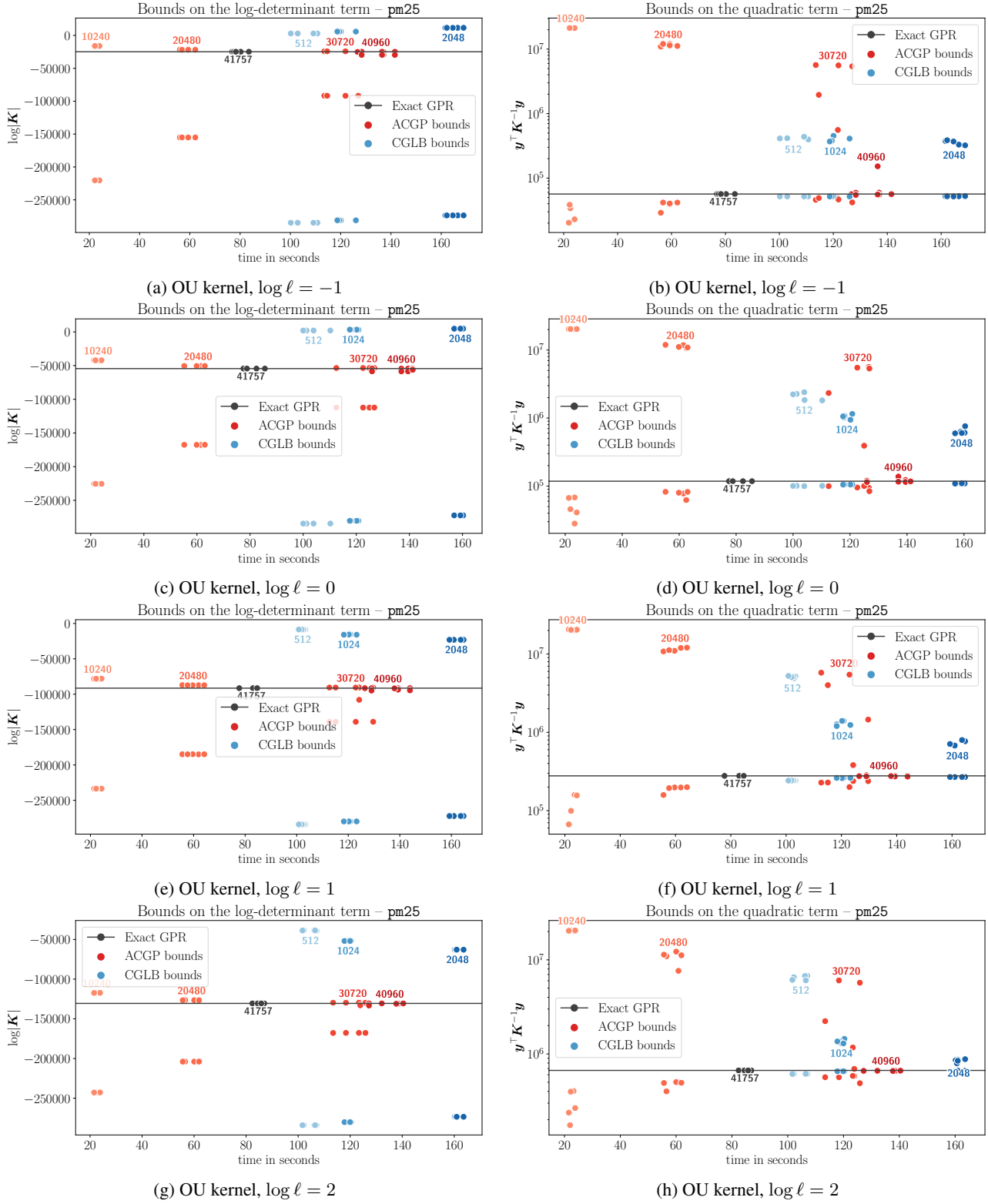
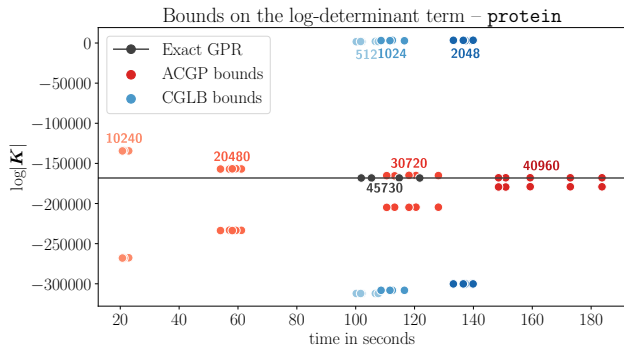
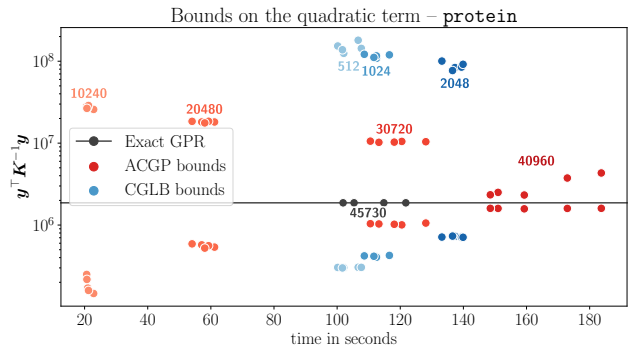


Figure 23: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the pm25 dataset using an Ornstein-Uhlenbeck (OU) kernel.

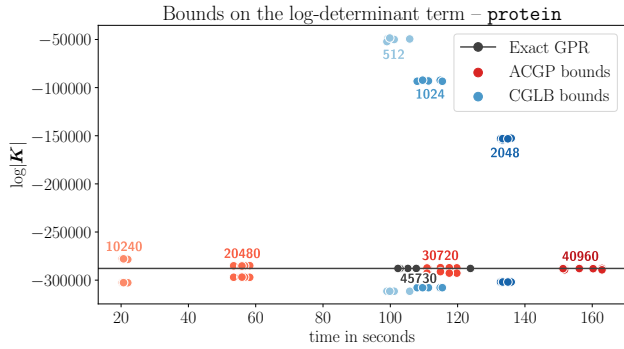
A.4. Bounds for experiments on protein



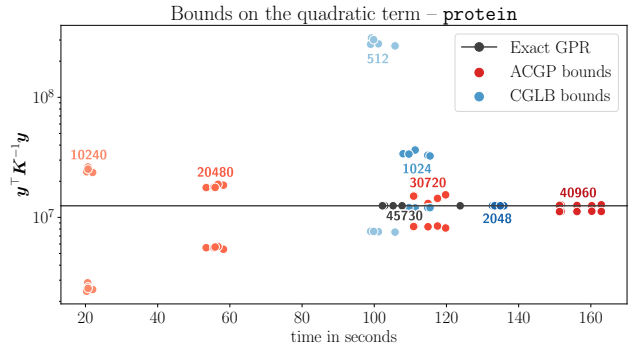
(a) SE kernel,  $\log \ell = -1$



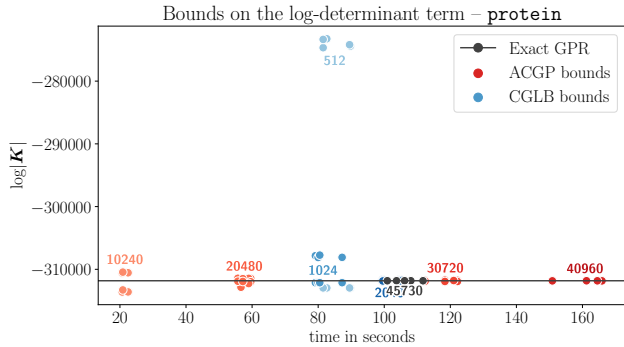
(b) SE kernel,  $\log \ell = -1$



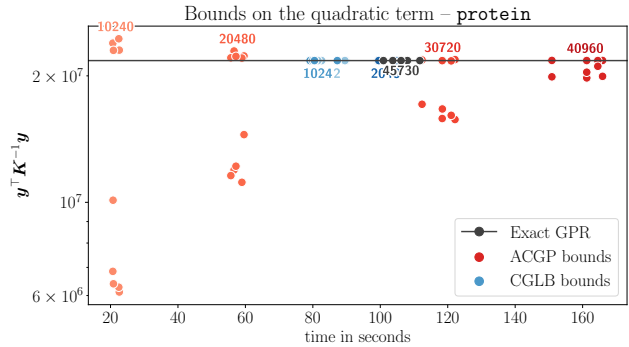
(c) SE kernel,  $\log \ell = 0$



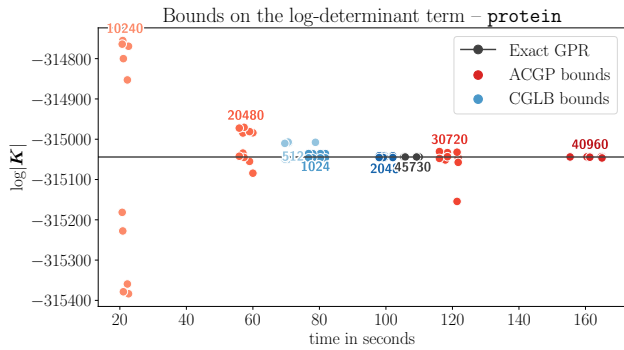
(d) SE kernel,  $\log \ell = 0$



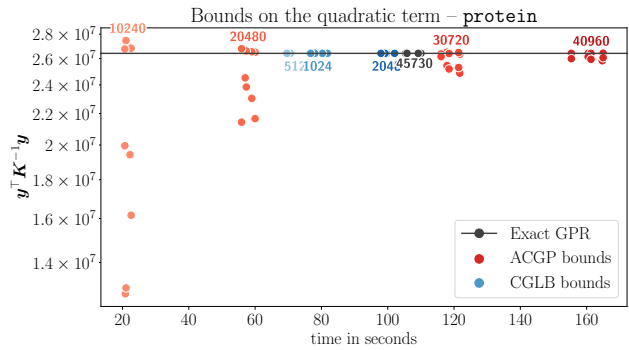
(e) SE kernel,  $\log \ell = 1$



(f) SE kernel,  $\log \ell = 1$



(g) SE kernel,  $\log \ell = 2$



(h) SE kernel,  $\log \ell = 2$

Figure 24: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the protein dataset.

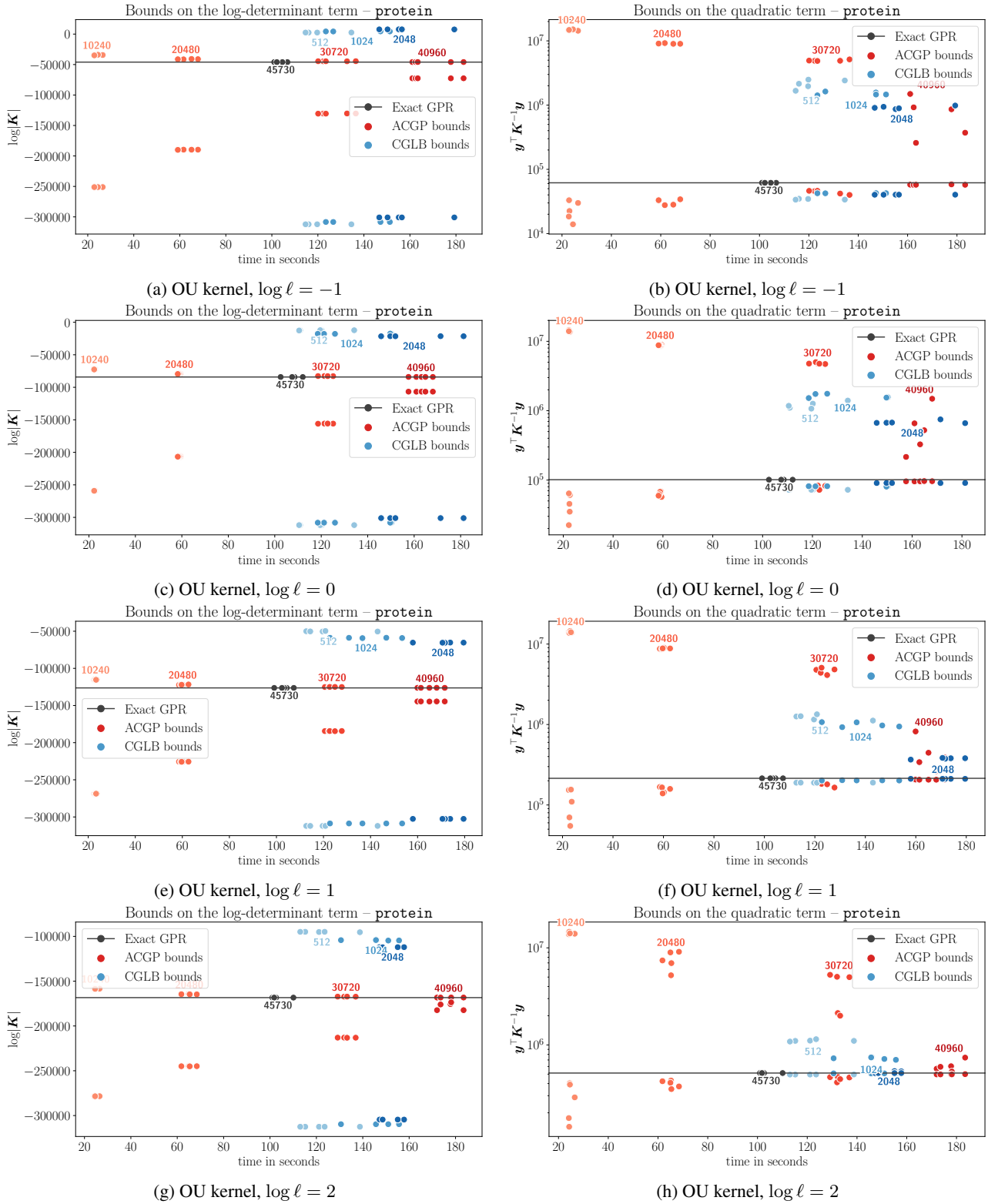


Figure 25: Upper and lower bounds on the log-determinant term (left column) and the quadratic term (right column) for the protein dataset using an Ornstein-Uhlenbeck (OU) kernel.

## B. Notation

We use a PYTHON-inspired index notation, abbreviating for example  $[y_1, \dots, y_n]^\top$  as  $\mathbf{y}_{:n}$ —observe that the indexing starts at 1. INDEXING binds before any other operation such that  $\mathbf{K}_{:,s:s}^{-1}$  is the inverse of  $\mathbf{K}_{:,s:s}$  and *not* all elements up to  $s$  of  $\mathbf{K}^{-1}$ . Differing from the main paper, we assume a heteroskedastic noise model such that we exchange  $\sigma^2 \in \mathbb{R}^+$  for a function of the inputs,  $\sigma^2 : \mathbb{X} \rightarrow \mathbb{R}^+$ . With  $\sigma^2$  we will refer to  $\inf_{\mathbf{x} \in \mathbb{X}} \sigma^2(\mathbf{x})$ , which we assume to be strictly larger than 0. For  $s \in \{1, \dots, N\}$  define  $\mathcal{F}_s := \sigma(\mathbf{x}_1, y_1, \dots, \mathbf{x}_s, y_s)$  to be the  $\sigma$ -algebra generated by  $\mathbf{x}_1, y_1, \dots, \mathbf{x}_s, y_s$ . With respect to the main article, we change the letter  $M$  to  $t$ . The motivation for the former notation is to highlight the role of the variable as a subset size, whereas in this part, the focus is on  $M$  as a stopping time.

## C. Proof Sketch

In this section of the appendix, we provide additional details, proofs and theorems on the proposed formulation. The principal equations included in Sec. 3 of the main manuscript are also included here for a better comprehension.

### C.1. The cumulative perspective

The key issue this paper is concerned with is how to estimate the full marginal likelihood,  $p(\mathbf{y})$ , given only a subset of  $n$  observations and their combined marginal likelihood,  $p(\mathbf{y}_{:n})$ . In particular, we will derive bounds, which are functions of seen observations, on this estimate. These bounds will allow us to decide, on the fly, when we have seen enough observations to accurately estimate the full marginal likelihood.

We can write  $\log p(\mathbf{y})$  equivalently as

$$\log p(\mathbf{y}) = \sum_{n=1}^N \log p(y_n | \mathbf{y}_{:n-1}). \quad (18)$$

With this equation in hand, the phenomena shown in Fig. 1 of the main manuscript becomes much clearer: The figure shows the value of Equation (18) for an increasing number of observations  $N$ . When the plot exhibits a linear trend it is because the summands  $\log p(y_n | \mathbf{y}_{:n-1})$  become approximately constant, implying that the model is not gaining additional knowledge after the  $n$ th observation.<sup>5</sup> From this perspective, we can craft an approximation by an *optimal stopping problem*: after processing observation  $n$ , we may decide whether to continue processing the sum or whether to stop and to estimate the remaining  $N-n$  terms.

### C.2. Extrapolation

For each potential stopping point  $t$  we can decompose Equation (18) into a sum of terms which have already been computed and a remaining sum

$$\log p(\mathbf{y}) = \underbrace{\sum_{n=1}^t \log p(y_n | \mathbf{y}_{:n-1})}_{A: \text{processed}} + \underbrace{\sum_{n=t+1}^N \log p(y_n | \mathbf{y}_{:n-1})}_{B: \text{remaining}}.$$

It is tempting to estimate  $B$  as  $\frac{N-t}{t}A$ , yet this estimator is biased. In the following, we will derive lower and upper bounds,  $\mathcal{L}_t$  and  $\mathcal{U}_t$ , such that conditioned on the points already processed,  $B$  can be sandwiched,

$$\mathbb{E}[\mathcal{L}_t | \mathbf{x}_1, y_1, \dots, \mathbf{x}_t, y_t] \leq \mathbb{E}[B | \mathbf{x}_1, y_1, \dots, \mathbf{x}_t, y_t] \leq \mathbb{E}[\mathcal{U}_t | \mathbf{x}_1, y_1, \dots, \mathbf{x}_t, y_t]. \quad (19)$$

These bounds tighten as we increase the number of observations, which allow us to monitor convergence of the approximation. We can then detect when the upper and lower bounds are sufficiently near each other, and stop computations early when the approximation is sufficiently good. This is in contrast to other approximations, where one specifies a computational budget, rather than a desired accuracy.

<sup>5</sup>An alternative way of understanding the linear trend is that the spectrum of the covariance matrix  $\mathbf{K}$  typically drop to  $\sigma^2$  at some point; since the log-determinant is the sum of the log-eigenvalues then the linear trend comes from additional  $2 \log \sigma$  terms in the sum.

### C.3. General bounds

A more practical-minded reader may safely skip this section and continue in Appendix C.6 where we show how to use the bounds to obtain a stopped Cholesky decomposition. Recall that we want to detect the case when the log-marginal likelihood starts to behave linearly with more processed datapoints as shown in Fig. 1 in the main manuscript. That is to say, the bounds presented in the following are valid in general but useful only in the linear setting.

The posterior of the  $n$ th observation conditioned on the previous is Gaussian with

$$\begin{aligned} p(y_n | \mathbf{y}_{:n-1}) &= \mathcal{N}(m_{n-1}(\mathbf{x}_n), k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)) \\ m_{n-1}(\mathbf{x}_n) &:= k(\mathbf{x}_n, \mathbf{X}_{:n-1}) \mathbf{K}_{n-1}^{-1} \mathbf{y}_{:n-1} \\ k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) &:= k(\mathbf{x}_n, \mathbf{x}_n) - k(\mathbf{x}_n, \mathbf{X}_{:n-1}) \mathbf{K}_{n-1}^{-1} k(\mathbf{X}_{:n-1}, \mathbf{x}_n), \end{aligned}$$

where we assumed (w.l.o.g) that  $\mu_0(\mathbf{x}) := 0$ . Inspecting these expressions one finds that

$$\log \det \mathbf{K}_N = \sum_{n=1}^N \log (k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)), \quad (20)$$

$$\mathbf{y}^\top \mathbf{K}_N^{-1} \mathbf{y} = \sum_{n=1}^N \frac{(y_n - m_{n-1}(\mathbf{x}_n))^2}{k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)}. \quad (21)$$

These expressions are permutation invariant. This allows us to prove that these terms cannot be too far from their expected values using a *Hoeffding's inequality for super-martingales* by Fan et al. (2012). This observation also holds for the conditional case, that is, after having observed  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_t, y_t)$ . Taking for granted that we can bound  $\mathbb{P}(|\log p(\mathbf{y}) - \mathbb{E}[\log p(\mathbf{y}) | \mathbf{x}_1, y_1, \dots, \mathbf{x}_\tau, y_\tau]| > \epsilon)$  for stopping times  $\tau$ , we proceed with the estimation of the expectation.

Recall that  $\frac{N-t}{t} \log p(\mathbf{y}_{:t})$  is *not* an unbiased estimator for  $\mathbb{E}[\log p(\mathbf{y}) | \mathbf{x}_1, y_1, \dots, \mathbf{x}_t, y_t]$ , due to the interaction of  $\mathbf{x}_{t+1}, y_{t+1}, \dots, \mathbf{x}_N, y_N$ . Our strategy is to find function families  $u$  (and  $l$ ) which upper (and lower) bound the expectation

$$\begin{aligned} l_{n,t}^d &\leq_E \log k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) \leq_E u_{n,t}^d \\ l_{n,t}^q &\leq_E \frac{(y_n - m_{n-1}(\mathbf{x}_n))^2}{k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)} \leq_E u_{n,t}^q, \end{aligned}$$

where  $\leq_E$  denotes that the inequality holds in expectation. We will choose the function families such that the unseen variables interact only in a *controlled* manner. More specifically,

$$f_{n,t}^x(\mathbf{x}_n, y_n, \dots, \mathbf{x}_1, y_1) = \sum_{j=s+1}^n g_t^{f,x}(\mathbf{z}_n, \mathbf{z}_j; \mathbf{z}_1, \dots, \mathbf{z}_s),$$

with  $f \in \{u, l\}$  and  $x \in \{d, q\}$ . The effect of this restriction becomes apparent when taking the expectation. The sum over the bounds becomes the sum of only two terms: variance and covariance, formally:

$$\mathbb{E} \left[ \sum_{n=s+1}^N f_{n,t}^x(\mathbf{z}_n, \dots, \mathbf{z}_1) | \sigma(\mathbf{z}_1, \dots, \mathbf{z}_s) \right] \quad (22)$$

$$\begin{aligned} &= (N-s) \mathbb{E} [g(\mathbf{z}_{s+1}, \mathbf{z}_{s+1}, \mathbf{z}_1, \dots, \mathbf{z}_n) | \sigma(\mathbf{z}_1, \dots, \mathbf{z}_s)] \\ &+ (N-s) \frac{N-s+1}{2} \mathbb{E} [g(\mathbf{z}_{s+1}, \mathbf{z}_{s+2}, \mathbf{z}_1, \dots, \mathbf{z}_n) | \sigma(\mathbf{z}_1, \dots, \mathbf{z}_s)]. \end{aligned} \quad (23)$$

We can estimate this expectation from the observations we obtained between  $s$  and  $t$ .

$$\begin{aligned} &\approx \frac{N-t}{t-s} \sum_{n=s+1}^t g(\mathbf{z}_n, \mathbf{z}_n, \mathbf{z}_1, \dots, \mathbf{z}_s) \\ &+ \frac{2(N-t)}{t-s} \frac{N-s+1}{2} \sum_{i=1}^{\frac{t-s}{2}} g(\mathbf{z}_{s+2i}, \mathbf{z}_{s+2i-1}, \mathbf{z}_1, \dots, \mathbf{z}_s). \end{aligned} \quad (24)$$

As mentioned before, we will present a way of choosing  $t$  in Appendix C.6.



#### C.4. Bounds on the log-determinant

Since the posterior variance of a Gaussian process can never increase with more data, the average of the (log) posterior variances is an estimator for an upper bound on the log-determinant (Bartels et al., 2021). Hence in this case, we simply ignore the interaction between the remaining variables. We set  $g(\mathbf{x}_n, \mathbf{x}_i) := \delta_{ni} \log(k_s(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n))$  where  $\delta_{ni}$  denotes Kronecker's  $\delta$ .

To obtain a lower bound we use that for  $c > 0$  and  $a \geq b \geq 0$ , one can show that  $\log(c + a - b) \geq \log(c + a) - \frac{b}{c}$  where the smaller  $b$  the better the bound. In our case  $c = \sigma^2(\mathbf{x}_n)$ ,  $a = k_s(\mathbf{x}_n, \mathbf{x}_n)$  and  $b = k_s(\mathbf{x}_n, \mathbf{X}_{s+1:n-1}) (k_s(\mathbf{X}_{s+1:n-1}, \mathbf{X}_{s+1:n-1}) + \sigma^2(\mathbf{X}_{s+1:n-1}))^{-1} k_s(\mathbf{X}_{s+1:n-1}, \mathbf{x}_n)$ . Underestimating the eigenvalues of  $k_s(\mathbf{X}_{s+1:n-1}, \mathbf{X}_{s+1:n-1})$  by 0 we obtain a lower bound, where each quantity can be estimated. Formally, for any  $s \leq t$ ,

$$\log(k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)) \geq \left( \log(k_s(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)) - \sum_{i=s+1}^{n-1} \frac{k_s(\mathbf{x}_n, \mathbf{x}_i)^2}{\sigma^2(\mathbf{x}_n)\sigma^2(\mathbf{x}_i)} \right). \quad (25)$$

This bound can be worse than the deterministic lower bound  $\min_{n'} \log \sigma^2(\mathbf{x}_{n'})$ . It depends on how large  $n$  is, how large the average correlation is and how small  $\sigma^2(\cdot)$  is. We can determine the number of steps  $n - s$  that this bound is better by solving a quadratic equation. Denote with  $\mu$  the estimator for the left addend and with  $\rho$  the estimator for the second addend. The tipping point  $\psi$  is the solution of  $(\psi - s) \left( \mu - \frac{\psi - s + 1}{2} \rho \right) \leq (\psi - s) \min \log \sigma^2(\cdot)$ . One solution is  $\psi = s$ , the other is

$$\psi := \lfloor s - 1 + \frac{2}{\rho} (\mu - \min \log \sigma^2(\cdot)) \rfloor. \quad (26)$$

Hence, for  $n > \psi$  we set  $u_n^d := \min \log \sigma^2(\cdot)$ .

Observe that, the smaller  $k_s(j, j+1)^2$  the closer the bounds. This term represents the correlation of datapoints conditioned on the  $s$  datapoints observed before. Thus, our bounds come together, when incoming observations become independent conditioned on what was already observed. Essentially, that  $k_s(\mathbf{x}_j, \mathbf{x}_{j+1})^2 = 0$  is the basic assumption of inducing input approximations (Quiñonero-Candela & Rasmussen, 2005).

#### C.5. Bounds on the quadratic form

For an upper bound on the quadratic form we apply a similar trick:

$$\frac{x}{c + a - b} \leq \frac{x(c + b)}{c(c + a)}, \quad (27)$$

where  $x \geq 0$ . Further we assume that in expectation the mean square error improves with more data. Formally,

$$\frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \leq E \frac{(y_j - m_s(\mathbf{x}_j))^2}{\sigma^2(\mathbf{x}_j) (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j))} \left( \sigma^2(\mathbf{x}_j) + \sum_{i=s+1}^{j-1} \frac{(k_s(\mathbf{x}_j, \mathbf{x}_i))^2}{\sigma^2(\mathbf{x}_i)} \right) \quad (28)$$

For a lower bound observe that

$$\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} = \mathbf{y}_{1:s}^\top \mathbf{K}_{s+1}^{-1} \mathbf{y}_{s+1} + (\mathbf{y}_{s+1:N} - m_s(\mathbf{X}_{t+1:N}))^\top \mathbf{Q}_{s+1:N}^{-1} (\mathbf{y}_{s+1:N} - m_s(\mathbf{X}_{t+1:N})) \quad (29)$$

where  $\mathbf{Q}_{s+1:j} := k_s(\mathbf{X}_{s+1:j}, \mathbf{X}_{s+1:j}) + \sigma^2(\mathbf{X}_{s+1:j})$  with  $j \geq s + 1$  for the posterior covariance matrix of  $\mathbf{X}_{s+1:j}$  conditioned on  $\mathbf{X}_{1:s}$ . We use a trick we first encountered in Kim & Teh (2018):  $\mathbf{y}^\top \mathbf{A}^{-1} \mathbf{y} \geq 2\mathbf{y}^\top \mathbf{b} - \mathbf{b}^\top \mathbf{A} \mathbf{b}$ , for any  $\mathbf{b}$ . Applying this inequality directly would result in a poor lower bound. For brevity introduce  $\mathbf{e} := \mathbf{y}_{s+1:N} - m_s(\mathbf{X}_{t+1:N})$ . We rewrite the second term as

$$\mathbf{e}^\top \text{diag}(\mathbf{e}) (\text{diag}(\mathbf{e}) \mathbf{Q}_{s+1:N} \text{diag}(\mathbf{e}))^{-1} \text{diag}(\mathbf{e}) \mathbf{e} \quad (30)$$

Now applying the inequality with  $\mathbf{b} := \alpha \mathbf{1}$ , we obtain

$$2\alpha \sum_{n=s+1}^N (y_n - m_s(\mathbf{x}_n))^2 - \alpha^2 \sum_{n, n'=s+1}^N (y_n - m_s(\mathbf{x}_n)) [\mathbf{Q}_{s+1:N}]_{nn'} (y_{n'} - m_s(\mathbf{x}_{n'})) \quad (31)$$

which is now in the form of Equation (22). After taking the expectation, the optimal value of  $\alpha$  is

$$\frac{\mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 | \mathcal{F}_s]}{\alpha_{\text{den}}} \quad (32)$$

where

$$\begin{aligned} \alpha_{\text{den}} = \mathbb{E} & \left[ (y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1}) \right] \\ & + \mathbb{E} [(N - s - 1) (y_{s+1} - m_s(\mathbf{x}_{s+1})) (y_{s+2} - m_s(\mathbf{x}_{s+2})) (k_s(s+1, s+2)) | \mathcal{F}_s], \end{aligned} \quad (33)$$

which is  $\mathcal{F}_s$ -measurable and we can estimate it.

Observe that, the smaller the square error  $(y_j - m_s(\mathbf{x}_j))^2$ , the closer the bounds. That is, if the model fit is good, the quadratic form can be easily identified.

### C.6. Using the Bounds for Stopping the Cholesky

We will use the same stopping strategy as (Mnih et al., 2008; Bartels, 2020): when the difference between bounds becomes sufficiently small and their absolute value is far away from zero. More precisely, when having deterministic bounds  $\mathcal{L} \leq x \leq \mathcal{U}$  on a number  $x$ , with

$$\frac{\mathcal{U} - \mathcal{L}}{2 \min(|\mathcal{U}|, |\mathcal{L}|)} \leq r \text{ and} \quad (34)$$

$$\text{sign } \mathcal{U} = \text{sign } \mathcal{L}, \quad (35)$$

then the relative error of the estimate  $\frac{1}{2}(\mathcal{U} + \mathcal{L})$  is less than  $r$ , that is  $|\frac{\frac{1}{2}(\mathcal{U} + \mathcal{L}) - x}{x}| \leq r$ .

**Remark 4.** In our experiments, we do **not** use  $\frac{1}{2}(\mathcal{U} + \mathcal{L})$  as estimator. When computing gradients for kernel parameters, the points from  $s$  to  $\tau$  enter the equation with a different weighting than the first  $s$  datapoints for which there is no justification. In exploratory experiments this was **not problematic**. Nevertheless, we instead use the **biased** estimator  $(N - \tau) \frac{1}{\tau} \log p(\mathbf{y}_{:\tau})$ , since we use our own, faster auto-differentiation implementation for this term.

**Remark 5.** Another ingredient where we deviate from the theory before is the estimation of the average correlation. In theory, the estimator is allowed to take only every second entry of the off-diagonal. Yet, we could use the same estimator using entries with an offset of 1 and still would get a valid estimator. Hence, we the average of the two should also be a decent estimator. We believe that it should be possible to prove that the average of the two estimators is also a decent estimator. Therefore, in practice, take the average over all indices.

The question remains how to use bounds and stopping strategy to derive an approximation algorithm. We transform the exact Cholesky decomposition for that purpose. For brevity denote  $\mathbf{L}_s := \text{chol}[k(\mathbf{X}_{:,s}, \mathbf{X}_{:,s}) + \sigma^2(\mathbf{X}_{:,s})]$  and  $\mathbf{T}_s := k(\mathbf{X}_{s+1:, \mathbf{X}}) \mathbf{L}_s^{-\top}$ . For any  $s \in \{1, \dots, N\}$ :

$$\mathbf{L}_N = \begin{bmatrix} \mathbf{L}_s & \mathbf{0} \\ \mathbf{T} & \text{chol}[k(\mathbf{X}_{s+1:, s+1:}) - \mathbf{T} \mathbf{T}^\top] \end{bmatrix} \quad (36)$$

One can verify that  $\mathbf{L}_N$  is indeed the Cholesky of  $\mathbf{K}_N$  by evaluating  $\mathbf{L}_N \mathbf{L}_N^\top$ . Observe that  $k(\mathbf{X}_{s+1:, s+1:}) - \mathbf{T} \mathbf{T}^\top$  is the posterior covariance matrix of the  $\mathbf{y}_{s+1:}$  conditioned on  $\mathbf{y}_{s:}$ . Hence, in the step before the Cholesky of the posterior covariance matrix is computed, we can estimate our log-determinant bounds.

Similar reasoning applies for solving the linear equation system. We can write

$$\boldsymbol{\alpha}_N = \begin{bmatrix} \boldsymbol{\alpha}_s \\ \text{chol}[k(\mathbf{X}_{s+1:, s+1:}) - \mathbf{T} \mathbf{T}^\top]^{-1} (\mathbf{y}_{s+1:} - \mathbf{T}_s \boldsymbol{\alpha}_s) \end{bmatrix} \quad (37)$$

Now observe that  $\mathbf{T}_s \boldsymbol{\alpha}_s = m_s(\mathbf{X}_{s+1:})$ . Hence, before the solving the lower equation system (and before computing the posterior Cholesky), we can compute our bounds for the quadratic form. There are different options to implement the Cholesky decomposition. We use a blocked, row-wise implementation (George et al., 1986). For a practical implementation see Algorithm 1 and Algorithm 2.

**Algorithm 1** blocked and recursive formulation of Cholesky decomposition and Gaussian elimination, augmented with our stopping conditions.

```

1  procedure ACGP( $k(\cdot, \cdot), \mu(\cdot), \sigma^2(\cdot), \mathbf{X}, \mathbf{y}, m, N_{\max}$ )
2       $\mathbf{A} \leftarrow \mathbf{0}^{N_{\max} \times N_{\max}}, \boldsymbol{\alpha} \leftarrow \mathbf{0}^{N_{\max}}$  // allocate memory
3       $\mathbf{A}_{1:m, 1:m} \leftarrow k(\mathbf{X}_{1:m}) + \sigma^2(\mathbf{X}_{1:m})$  // initialize kernel matrix
4       $\boldsymbol{\alpha}_{1:m} \leftarrow \mathbf{y}_{1:m} - \mu(\mathbf{X}_{1:m})$  // evaluate mean function for the same datapoints
5       $\mathbf{A}_{1:m, 1:m} \leftarrow \text{chol}(\mathbf{A}_{1:m, 1:m})$  // call to low-level Cholesky
6       $\boldsymbol{\alpha}_{1:m} \leftarrow \mathbf{A}_{1:m, 1:m}^{-1} \boldsymbol{\alpha}_{1:m}$  // second back-substitution step
7       $i \leftarrow m + 1, j \leftarrow \min(i + m, N)$ 
8      while  $i < N_{\max}$  do
9           $\mathbf{A}_{i:j, 1:i} \leftarrow k(\mathbf{X}_{i:j}, \mathbf{X}_{1:i})$  // evaluate required block-off-diagonal part of the kernel matrix
10          $\mathbf{A}_{i:j, 1:i} \leftarrow \mathbf{A}_{i:j, 1:i} \mathbf{A}_{1:i, 1:i}^{-\top}$  // solve triangular linear equation system
11          $\mathbf{A}_{i:j, i:j} \leftarrow k(\mathbf{X}_{i:j}) + \sigma^2(\mathbf{X}_{i:j})$  // evaluate required block-diagonal part of the kernel matrix
12          $\boldsymbol{\alpha}_{i:j} \leftarrow \mathbf{y}_{i:j} - \mu(\mathbf{X}_{i:j})$  // evaluate mean function for the same datapoints
13          $\mathbf{A}_{i:j, i:j} \leftarrow \mathbf{A}_{i:j, i:j} - \mathbf{A}_{i:j, 1:i} \mathbf{A}_{i:j, 1:i}^{\top}$  // down-date
14         // now  $\mathbf{A}_{i:j, i:j} = \mathbf{Q}_{s+1:j}$ 
15          $\boldsymbol{\alpha}_{i:j} \leftarrow \boldsymbol{\alpha}_{i:j} - \mathbf{A}_{i:j, 1:i} \boldsymbol{\alpha}_{1:i}$  // now  $\boldsymbol{\alpha}_{i:j}$  contains  $\mathbf{y}_{i:j} - m_i(\mathbf{X}_{i:j})$ 
16          $\mathcal{L}, \mathcal{U} \leftarrow \text{EvaluateBounds}(i, j)$  // costs  $\mathcal{O}(j - i)$ 
17         if Equations (34) and (35) fulfilled then
18             return estimator
19         end if
20          $\mathbf{A}_{i:j, i:j} \leftarrow \text{chol}(\mathbf{A}_{i:j, i:j})$  // finish computing Cholesky for data-points up to index  $j$ 
21          $\boldsymbol{\alpha}_{i:j} \leftarrow \mathbf{A}_{i:j, i:j}^{-1} \boldsymbol{\alpha}_{i:j}$  // finish solving linear equation system for index up to  $j$ 
22          $i \leftarrow i + m, j \leftarrow \min(i + m, N_{\max})$ 
23     end while // now  $\mathbf{A} = \mathbf{L}$  and  $\boldsymbol{\alpha} = \mathbf{L}^{-1}(\mathbf{y} - \mu(\mathbf{X}))$ 
24     return estimator
25 end procedure

```

**Algorithm 2** bound algorithm as used in our experiments. The algorithm deviates slightly from our theory. We use Equation (64) for the upper bound in the quadratic form, and we use all off-diagonal entries (instead of only every second).

```

1  procedure EVALUATEBOUNDS( $s, t$ )
2       $D \leftarrow \sum_{j=1}^s 2 \log \mathbf{A}_{jj}$  // in practice we reuse the sum from the last iteration
3       $Q \leftarrow \sum_{j=1}^s \boldsymbol{\alpha}_j^2$ 
4       $\mu \leftarrow \frac{1}{t-s} \sum_{j=s+1}^t 2 \log \mathbf{A}_{jj}$  // average variance of the new points conditioned on all points processed until  $s$ 
5       $\mathcal{U}_D \leftarrow D + (N - s)\mu$ 
6       $\rho \leftarrow \frac{1}{t-s-1} \sum_{j=s+1}^{t-1} \mathbf{A}_{j, j+1}^2$  // average square correlation (deviating from theory!)
7       $\psi \leftarrow \lfloor s - 1 + \frac{2}{\rho} (\mu - \log \sigma^2) \rfloor$  // number of steps the probabilistic bound is better than the deterministic
8       $\mathcal{L}_D \leftarrow D + (\psi - s) \left( \mu - \frac{\psi-s-1}{2} \rho \right) + (N - \psi) \log \sigma^2$ 
9       $\mathcal{U}_Q \leftarrow Q + \frac{N-s}{t-s-1} \sum_{j=s+1}^{t-1} \frac{\boldsymbol{\alpha}_j^2}{\mathbf{A}_{j,j} \sigma^2(\mathbf{x}_j)} \left( \sigma^2(\mathbf{x}_j) + \frac{\mathbf{A}_{j, j+1}^2}{\sigma^2(\mathbf{x}_{j+1})} \right)$ 
10      $m \leftarrow \frac{1}{t-s} \sum_{j=s+1}^t \boldsymbol{\alpha}_j^2$  // mean square error
11      $\mu \leftarrow \frac{1}{t-s} \sum_{j=s+1}^t \boldsymbol{\alpha}_j^2 \mathbf{A}_{j,j}$ 
12      $\rho \leftarrow \frac{1}{t-s-1} \sum_{j=s+1}^{t-1} \boldsymbol{\alpha}_j \boldsymbol{\alpha}_{j+1} \mathbf{A}_{j, j+1}$ 
13      $\alpha \leftarrow \frac{m}{\mu + (N-s-1)\rho}$ 
14      $\mathcal{L}_Q \leftarrow Q + \alpha(N-s)(2m - \alpha(\mu + \rho(N-s-1)))$ 
15     return  $\mathcal{L}_D + \mathcal{L}_Q, \mathcal{U}_D + \mathcal{U}_Q$ 
16 end procedure

```

## D. Assumptions

**Assumption 6.** Let  $(\Omega, \mathcal{F}, \mathbb{P})$  be a probability space and let  $(\mathbf{x}_j, y_j)_{j=1}^N$  be a sequence of independent and identically distributed random vectors with  $\mathbf{x} : \Omega \rightarrow \mathbb{R}^D$  and  $y : \Omega \rightarrow \mathbb{R}$ .

**Assumption 7.** For all  $s, i, j, t$  with  $s < i \leq j \leq N$  and functions  $f(\mathbf{x}_j, \mathbf{x}_i; \mathbf{x}_1, \dots, \mathbf{x}_s) \geq 0$

$$\mathbb{E} \left[ f(\mathbf{x}_j, \mathbf{x}_i) (y_j - m_{j-1}(\mathbf{x}_j))^2 \mid \mathcal{F}_s \right] \leq \mathbb{E} \left[ f(\mathbf{x}_j, \mathbf{x}_i) (y_j - m_s(\mathbf{x}_j))^2 \mid \mathcal{F}_s \right] \quad (38)$$

where  $f(\mathbf{x}_j, \mathbf{x}_i) \in \left\{ \frac{1}{k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)}, \frac{k_s(\mathbf{x}_j, \mathbf{x}_i)^2}{(k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j))\sigma^2(\mathbf{x}_j)\sigma^2(\mathbf{x}_i)} \right\}$ .

That is, we assume that in expectation the estimator improves with more data. Note that,  $f$  can not depend on any entries of  $\mathbf{y}$ .

## E. Main Theorem

**Theorem 8.** Assume that Assumption 6 and Assumption 7 hold. For any even  $m \in \{2, 4, \dots, N - 2\}$  and any  $s \in \{1, \dots, N - m\}$ , the bounds defined in Equations (8), (9), (11) and (12) hold in expectation:

$$\begin{aligned} \mathbb{E}[\mathcal{L}_D \mid \mathcal{F}_s] &\leq \mathbb{E}[\log(\det[\mathbf{K}]) \mid \mathcal{F}_s] \leq \mathbb{E}[\mathcal{U}_D \mid \mathcal{F}_s] \text{ and} \\ \mathbb{E}[\mathcal{L}_Q \mid \mathcal{F}_s] &\leq \mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] \leq \mathbb{E}[\mathcal{U}_Q \mid \mathcal{F}_s]. \end{aligned}$$

*Proof.* Follows from Theorems 9, 12 and 16, and Theorem 28 in Bartels (2020). □

## F. Proof for the Lower Bound on the Determinant

**Theorem 9.** Assume that Assumption 6 holds, and that  $m \in \{2, 4, \dots, N\}$  is an even number. Set  $t := s + m$ , then, for all  $s \in \{1, \dots, N - m\}$

$$\mathbb{E}[\mathcal{L}_D \mid \mathcal{F}_s] \leq \mathbb{E}[D_N \mid \mathcal{F}_s].$$

$$\mathcal{L}_D := \log \det \mathbf{K}_{:,s,:s} + (\psi_t - s) \left( \log \underline{\mu}_t - \frac{\psi_t - s - 1}{2} \tilde{\rho}_t \right) + (N - \psi_t) \log \sigma^2 \quad (39)$$

// the lower bound

$$\log \underline{\mu}_t := \frac{1}{m} \sum_{j=s+1}^t \log (\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j)) \quad (40)$$

// (under-)estimate of the posterior variance conditioned on  $s$  points

$$\tilde{\rho}_t := \frac{2}{m} \sum_{j=\frac{s+1}{2}}^{\frac{t-1}{2}} \frac{k_s(\mathbf{x}_{2j+1}, \mathbf{x}_{2j})^2}{\sigma^2(\mathbf{x}_{2j+1})\sigma^2(\mathbf{x}_{2j})} \quad (41)$$

// (over-)estimate of the correlation conditioned on  $s$  points

$$\psi_t := s + \max p \text{ where } p \text{ is such that} \quad (42)$$

$$p \left( \log \underline{\mu}_t - \frac{p-1}{2} \tilde{\rho}_t \right) \geq p \log \sigma^2 \quad (43)$$

// number of steps that we suspect the decrease in variance to be controllable

$$= \max(N, \lfloor s - 1 + \frac{2}{\rho_D} (\mu_D - \log \sigma^2) \rfloor) \quad (44)$$

*Proof.*

$$\begin{aligned}
 & \mathbb{E}[\mathcal{L}_D \mid \mathcal{F}_s] - \mathbb{E}[\log \det \mathbf{K} \mid \mathcal{F}_s] = \mathbb{E}[\mathcal{L}_D - \log \det \mathbf{K} \mid \mathcal{F}_s] \\
 & = \mathbb{E} \left[ (\psi_t - s) \left( \log \underline{\mu}_t - \frac{\psi_t - s - 1}{2} \tilde{\rho}_t \right) + (N - \psi_t) \log \sigma^2 - \sum_{j=s+1}^N \log (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \mid \mathcal{F}_s \right] \\
 & \quad // \text{ using the definition of } \mathcal{L}_t \text{ and slightly simplifying using Lemma 20} \\
 & \leq \mathbb{E} \left[ (\psi_t - s) \left( \log \underline{\mu}_t - \frac{\psi_t - s - 1}{2} \tilde{\rho}_t \right) - \sum_{j=s+1}^{\psi_t} \log (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \mid \mathcal{F}_s \right] \\
 & \quad // \text{ using that } \log \sigma^2 \leq \log (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \text{ for all } j \\
 & = (\psi_t - s) \left( \mathbb{E} [\log (\sigma^2 + k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \mid \mathcal{F}_s)] - \frac{\psi_t - s - 1}{2} \mathbb{E} \left[ \frac{k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+2})^2}{\sigma^2(\mathbf{x}_{t+1}) \sigma^2(\mathbf{x}_{t+2})} \mid \mathcal{F}_s \right] \right) \\
 & \quad - \mathbb{E} \left[ \sum_{j=s+1}^{\psi_t} \log (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \mid \mathcal{F}_s \right] \\
 & \quad // \text{ using Assumption 6} \\
 & \leq 0 \\
 & \quad // \text{ Lemma 11}
 \end{aligned}$$

□

**Lemma 10.** For  $c > 0$  and  $b \geq a \geq 0$ :

$$\log(c + b - a) \geq \log(c + b) - \frac{a}{c}$$

*Proof.* For  $a = 0$ , the statement is true with equality. We rewrite the inequality as

$$\frac{a}{c} \geq \log \left( \frac{c + b}{c + b - a} \right) = \log \left( 1 + \frac{a}{c + b - a} \right).$$

For the case  $a = b$ , apply the exponential function on both sides, and the statement follows from  $e^x \geq x + 1$  for all  $x$ . For  $a \in (0, b)$ , consider  $f(a) := \log(c + b - a) + \frac{a}{c} - \log(c + b)$ . The first derivative of this function is  $f'(a) = -\frac{1}{c + b - a} + \frac{1}{c}$ , which is always positive for  $a \in (0, b)$ . Since  $f(0) = 0$ , we must have  $f(a) \geq 0$  for all  $a \in (a, b)$ . □

**Lemma 11.** For all  $n \geq t \geq s$ :

$$\begin{aligned}
 \mathbb{E} \left[ \sum_{j=t+1}^n \log (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \mid \mathcal{F}_s \right] & \geq (n - t) \left( \mathbb{E} [\log (\sigma^2(\mathbf{x}_{t+1}) + k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})) \mid \mathcal{F}_s] \right. \\
 & \quad \left. - \frac{n - t - 1}{2\sigma^4} \mathbb{E} [k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+2})^2 \mid \mathcal{F}_s] \right)
 \end{aligned}$$

*Proof.* Introduce  $\bar{\mathbf{X}}_j := [\mathbf{x}_{s+1}, \dots, \mathbf{x}_{j-1}]$  with the convention  $k_s(\mathbf{x}_{s+1}, \bar{\mathbf{X}}_{s+1}) := 0$ .

$$\mathbb{E} \left[ \sum_{j=t+1}^n \log k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j) \mid \mathcal{F}_s \right] \quad (45)$$

$$= \mathbb{E} \left[ \sum_{j=t+1}^n \log \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j) - k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (k_s(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_j) + \sigma^2(\bar{\mathbf{X}}_j))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j) \right) \mid \mathcal{F}_s \right] \quad (46)$$

// Lemma 22

$$\geq \mathbb{E} \left[ \sum_{j=t+1}^n \left( \log \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j) \right) - \frac{1}{\sigma^2(\mathbf{x}_j)} k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (k_s(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_j) + \sigma^2(\bar{\mathbf{X}}_j))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j) \right) \mid \mathcal{F}_s \right] \quad (47)$$

// Lemma 10

$$\geq \mathbb{E} \left[ \sum_{j=t+1}^n \left( \log \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j) \right) - \frac{1}{\sigma^2(\mathbf{x}_j)} k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (\sigma^2(\bar{\mathbf{X}}_j))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j) \right) \mid \mathcal{F}_s \right] \quad (48)$$

// underestimating  $k_s(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_j)$  by  $\mathbf{0}$

$$\geq \mathbb{E} \left[ \sum_{j=t+1}^n \left( \log \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j) \right) - \frac{1}{\sigma^2(\mathbf{x}_j)} \sum_{i=t+1}^{j-1} \frac{k_s(\mathbf{x}_j, \mathbf{x}_i)^2}{\sigma^2(\mathbf{x}_i)} \right) \mid \mathcal{F}_s \right] \quad (49)$$

// writing the vector multiplication as sum

$$= (n-t) \mathbb{E} \left[ \log \left( \sigma^2 + k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \right) \mid \mathcal{F}_s \right] \quad (50)$$

$$+ \frac{(n-t)(n-t-1)}{2} \mathbb{E} \left[ \frac{k_s(\mathbf{x}_{t+1}, \mathbf{x}_{t+2})^2}{\sigma^2(\mathbf{x}_{t+1}) \sigma^2(\mathbf{x}_{t+2})} \mid \mathcal{F}_s \right]$$

// using Assumption 6 and then applying Lemma 23

□

## G. Proof for the Upper Bound on the Quadratic Form

**Theorem 12.** Assume that Assumption 6 and Assumption 7 hold. Let  $m \in \mathbb{N}$  be even, then for all  $s \in \{1, \dots, N-m\}$

$$\mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] \leq \mathbb{E}[\mathcal{U}_Q \mid \mathcal{F}_s],$$

where

$$\mathcal{U}_Q := \mathbf{y}_{:s}^\top \mathbf{K}_{:,s}^{-1} \mathbf{y}_{:s} + (N-s) \left( \mu_t + \frac{N-s-1}{2} \tilde{\rho}_t \right) \quad (51)$$

// the upper bound

$$\mu_t := \frac{1}{t-s} \sum_{j=s+1}^t \frac{(y_j - m_s(\mathbf{x}_j))^2}{k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \quad (52)$$

$$\tilde{\rho}_t := \frac{2}{t-s} \sum_{j=\frac{s+2}{2}}^{\frac{t}{2}} \frac{(y_{2j} - m_s(\mathbf{x}_{2j}))^2 k_s(\mathbf{x}_{2j}, \mathbf{x}_{2j-1})^2}{(k_s(\mathbf{x}_{2j}, \mathbf{x}_{2j}) + \sigma^2(\mathbf{x}_{2j})) \sigma^2(\mathbf{x}_{2j}) \sigma^2(\mathbf{x}_{2j-1})}. \quad (53)$$

*Proof.*

$$\begin{aligned}
 & \mathbb{E} [\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] - \mathbb{E} [\mathcal{U}_Q \mid \mathcal{F}_s] \\
 &= \mathbb{E} \left[ \sum_{j=s+1}^N \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} - (N-s) \left( \mu_t + \frac{N-s-1}{2} \tilde{\rho}_t \right) \mid \mathcal{F}_s \right] \\
 & \quad // \text{ using the definition of } \mathcal{U}_Q \text{ and slightly simplifying with Lemma 21} \\
 &= \mathbb{E} \left[ \sum_{j=s+1}^{\psi_t} \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \\
 & \quad - (N-s) \left( \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2}{k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] \right) \\
 & \quad - (N-s) \frac{N-s-1}{2} \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2})^2}{(k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \sigma^2(\mathbf{x}_{s+2})} \mid \mathcal{F}_s \right] \\
 & \quad // \text{ using Assumption 6} \\
 & \leq 0 \\
 & \quad // \text{ Lemma 14}
 \end{aligned}$$

□

### G.1. Tools for this proof

**Lemma 13.** For  $c > 0$ ,  $b, x \geq 0$  and  $a \geq b$ :

$$\frac{x}{c+a-b} \leq \frac{x}{c} \left( 1 - \frac{a-b}{c+a} \right) = \frac{x(c+b)}{c(c+a)}$$

*Proof.*

$$\begin{aligned}
 \frac{x}{c+a-b} &= \frac{x}{c} \left( \frac{c}{c+a-b} \right) \\
 &= \frac{x}{c} \left( 1 - \frac{a-b}{c+a-b} \right) \\
 &\leq \frac{x}{c} \left( 1 - \frac{c+a-b}{c+a} \frac{a-b}{c+a-b} \right) \\
 & \quad // \text{ since } \frac{c+a-b}{c+a} \leq 1 \\
 &= \frac{x}{c} \left( 1 - \frac{a-b}{c+a} \right) \\
 & \quad // \text{ cancelling terms}
 \end{aligned}$$

□

**Lemma 14.** For all  $s, t$  and  $n$  with  $n \geq t \geq s$ :

$$\begin{aligned}
 & \mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \\
 & \leq (n-t) \left( \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2}{k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] \right) \\
 & \quad + (n-t) \left( \left( \frac{n+t+1}{2} - s \right) \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2})^2}{(k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \sigma^2(\mathbf{x}_{s+2})} \mid \mathcal{F}_s \right] \right)
 \end{aligned}$$

*Proof.* Introduce  $\bar{\mathbf{X}}_j := [\mathbf{x}_{s+1}, \dots, \mathbf{x}_{j-1}]$  with the convention  $k_s(\mathbf{x}_{s+1}, \bar{\mathbf{X}}_{s+1}) := 0$ .

$$\mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \quad (54)$$

$$= \mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j) - k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (k_s(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_j) + \sigma^2(\bar{\mathbf{X}}))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j)} \mid \mathcal{F}_s \right] \quad (55)$$

// Lemma 22

$$\leq \mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2 \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (k_s(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_j) + \sigma^2(\bar{\mathbf{X}}))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j) \right)}{\sigma^2(\mathbf{x}_j) (\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j))} \mid \mathcal{F}_s \right] \quad (56)$$

// Lemma 13

$$\leq \mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2 \left( \sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \bar{\mathbf{X}}_j) (\sigma^2(\bar{\mathbf{X}}))^{-1} k_s(\bar{\mathbf{X}}_j, \mathbf{x}_j) \right)}{\sigma^2(\mathbf{x}_j) (\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j))} \mid \mathcal{F}_s \right] \quad (57)$$

// underestimating the eigenvalues of  $k_s(\bar{\mathbf{X}}, \bar{\mathbf{X}})$  by 0

$$= \mathbb{E} \left[ \sum_{j=t+1}^n \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2 \left( \sigma^2(\mathbf{x}_j) + \sum_{i=s+1}^{j-1} \frac{k_s(\mathbf{x}_j, \mathbf{x}_i)^2}{\sigma^2(\mathbf{x}_i)} \right)}{\sigma^2(\mathbf{x}_j) (\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j))} \mid \mathcal{F}_s \right] \quad (58)$$

// writing the vector-product explicitly as a sum

$$= \sum_{j=t+1}^n \left( \mathbb{E} \left[ \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j)} \mid \mathcal{F}_s \right] + \sum_{i=s+1}^{j-1} \mathbb{E} \left[ \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2 k_s(\mathbf{x}_j, \mathbf{x}_i)^2}{(k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \sigma^2(\mathbf{x}_i) \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \right) \quad (59)$$

// linearity of expectation

$$= \sum_{j=t+1}^n \left( \mathbb{E} \left[ \frac{(y_j - m_s(\mathbf{x}_j))^2}{\sigma^2(\mathbf{x}_j) + k_s(\mathbf{x}_j, \mathbf{x}_j)} \mid \mathcal{F}_s \right] + \sum_{i=s+1}^{j-1} \mathbb{E} \left[ \frac{(y_j - m_s(\mathbf{x}_j))^2 k_s(\mathbf{x}_j, \mathbf{x}_i)^2}{(k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \sigma^2(\mathbf{x}_i) \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \right) \quad (60)$$

// by assumption Equation (38)

$$= \sum_{j=t+1}^n \left( \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2}{\sigma^2(\mathbf{x}_{s+1}) + k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] + \sum_{i=s+1}^{j-1} \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2})^2}{(k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \sigma^2(\mathbf{x}_{s+2}) \sigma^2(\mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] \right) \quad (61)$$

// using Assumption 6

$$= (n-t) \left( \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2}{\sigma^2(\mathbf{x}_{s+1}) + k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] \right) \quad (62)$$

$$+ (n-t) \left( \left( \frac{n+t+1}{2} - s \right) \mathbb{E} \left[ \frac{(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2})^2}{(k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \sigma^2(\mathbf{x}_{s+2}) \sigma^2(\mathbf{x}_{s+1})} \mid \mathcal{F}_s \right] \right) \quad (63)$$

// by Lemma 23

□

**Remark 15.** Similar to the proof of Theorem 9, we can improve the bound by monitoring how many steps the sum of average correlations is below the average variance. More precisely, we solve for the largest  $\psi \leq N$  such that

$$\mu_t + \frac{N - \psi - 1}{2} \tilde{\rho}_t \leq \frac{1}{t-s} \sum_{j=s+1}^t \frac{(y_j - m_s(\mathbf{x}_j))^2}{\sigma^2(\mathbf{x}_j)},$$



and replace the upper bound by

$$\mathcal{U}_t := \mathbf{y}_{:s}^\top \mathbf{K}_{:s,:s}^{-1} \mathbf{y}_{:s} + (\psi - s) \left( \mu_t + \frac{\psi - s - 1}{2} \tilde{\rho}_t \right) + \frac{N - \psi}{t - s} \sum_{j=s+1}^t \frac{(y_j - m_s(\mathbf{x}_{\mathbf{x}_j}))^2}{\sigma^2(\mathbf{x}_j)}. \quad (64)$$

## H. Proof for the Lower Bound on the Quadratic Form

**Theorem 16.** Assume that Assumption 6 and Assumption 7 hold. Let  $m \in \{2, \dots, N - 2\}$  be an even number less than  $N$ . For  $s \in \{1, \dots, N - m\}$ , let  $\alpha$  be  $\mathcal{F}_s$ -measurable, then

$$\mathbb{E}[\mathcal{L}_Q \mid \mathcal{F}_s] \leq \mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s]$$

where

$$\mathcal{L}_Q := \mathbf{y}_{:s}^\top \mathbf{K}_{:s,:s}^{-1} \mathbf{y}_{:s} + \alpha(N - s) \left( 2\mu_t - \alpha\mu'_t - \alpha \frac{N - s}{2} \rho_t \right) \quad (65)$$

$$\mu_t := \frac{1}{t - s} \sum_{j=s+1}^t (y_j - m_s(\mathbf{x}_j))^2 \quad (66)$$

$$\mu'_t := \frac{1}{t - s} \sum_{j=s+1}^t (y_j - m_s(\mathbf{x}_j))^2 (k_s(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \quad (67)$$

$$\rho_t := \frac{2}{t - s} \sum_{j=\frac{s+2}{2}}^{\frac{t}{2}} (y_{2j} - m_s(\mathbf{x}_{2j}))(y_{2j-1} - m_s(\mathbf{x}_{2j-1}))k_s(\mathbf{x}_{2j}, \mathbf{x}_{2j-1}) \quad (68)$$

**Remark 17.** In our implementation we choose  $\alpha := \frac{\mu_t}{\mu'_t + (N-s)\rho_t}$  which maximizes the lower bound, though violates the assumption that  $\alpha$  is  $\mathcal{F}_s$ -measurable.

*Proof.*

$$\begin{aligned} & \mathbb{E}[\mathcal{L}_Q \mid \mathcal{F}_s] - \mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] \\ &= \mathbb{E} \left[ \alpha(N - s) \left( 2\mu_t - \alpha\mu'_t - \alpha \frac{N - s}{2} \rho_t \right) - \sum_{j=s+1}^N \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \\ & \quad // \text{ using the definition of } \mathcal{L}_Q \text{ and slightly simplifying} \\ &= 2(N - s)\alpha \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 \mid \mathcal{F}_s] \\ & \quad - (N - s)\alpha^2 \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 (k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \mid \mathcal{F}_s] \\ & \quad - \frac{(N - s)^2}{2} \alpha^2 \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))(y_{s+2} - m_s(\mathbf{x}_{s+2}))k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2}) \mid \mathcal{F}_s] \\ & \quad - \sum_{j=s+1}^N \mathbb{E} \left[ \frac{(y_j - m_{j-1}(\mathbf{x}_j))^2}{k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)} \mid \mathcal{F}_s \right] \\ & \quad // \text{ using Assumption 6} \\ & \leq 0 \\ & \quad // \text{ using Lemma 18} \end{aligned}$$

□

**Lemma 18.** For all  $\mathcal{F}_s$ -measurable  $\alpha \in \mathbb{R}$  :

$$\begin{aligned} \mathbb{E}[\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] & \geq \mathbf{y}_{:s}^\top \mathbf{K}_{:s,:s}^{-1} \mathbf{y}_{:s} + 2\alpha(N - s)\mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 \mid \mathcal{F}_s] \\ & \quad - \alpha^2(N - s)\mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 (k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \mid \mathcal{F}_s] \\ & \quad - \alpha^2 \frac{(N - s)^2}{2} \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))(y_{s+2} - m_s(\mathbf{x}_{s+2}))k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2}) \mid \mathcal{F}_s] \end{aligned} \quad (69)$$

*Proof.* Using Lemma 21, we can write the quadratic form as a sum of two quadratic forms:

$$\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} = \mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s} + (\mathbf{y}_{s+1:} - m_s(\mathbf{X}_{s+1:}))^\top (k_s(\mathbf{X}_{s+1:}, \mathbf{X}_{s+1:}) + \sigma^2(\mathbf{X}_{s+1:}))^{-1} (\mathbf{y}_{s+1:} - m_s(\mathbf{X}_{s+1:})). \quad (70)$$

Define  $\mathbf{e} := (\mathbf{y}_{s+1:} - m_s(\mathbf{X}_{s+1:}))$  to rewrite the second addend as

$$\mathbf{e}^\top \text{Diag}(\mathbf{e}) (\text{Diag}(\mathbf{e}) (k_s(\mathbf{X}_{s+1:}, \mathbf{X}_{s+1:}) + \sigma^2(\mathbf{X}_{s+1:})) \text{Diag}(\mathbf{e}))^{-1} \text{Diag}(\mathbf{e}) \mathbf{e} \quad (71)$$

We use a trick we first encountered in Kim & Teh (2018):  $\mathbf{y}^\top \mathbf{A}^{-1} \mathbf{y} \geq 2\mathbf{y}^\top \mathbf{b} - \mathbf{b}^\top \mathbf{A} \mathbf{b}$ , for any  $\mathbf{b}$ . Choose  $\mathbf{b} := \alpha \mathbf{1}$  and observe that  $\mathbf{b}$  is  $\mathcal{F}_s$ -measurable.

$$\mathbb{E} [\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \mid \mathcal{F}_s] \quad (72)$$

$$= \mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s} + \mathbb{E} \left[ (\mathbf{y}_{s+1:} - m_s(\mathbf{X}_{s+1:}))^\top (k_s(\mathbf{X}_{s+1:}, \mathbf{X}_{s+1:}) + \sigma^2(\mathbf{X}_{s+1:}))^{-1} (\mathbf{y}_{s+1:} - m_s(\mathbf{X}_{s+1:})) \mid \mathcal{F}_s \right] \quad (73)$$

// since  $\mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s}$  is  $\mathcal{F}_s$ -measurable

$$\geq \mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s} + 2\alpha \mathbf{1}^\top \mathbb{E} [\text{Diag}(\mathbf{e}) \mathbf{e}] - \alpha^2 \mathbf{1}^\top \mathbb{E} [\text{Diag}(\mathbf{e}) (k_s(\mathbf{X}_{s+1:}, \mathbf{X}_{s+1:}) + \sigma^2(\mathbf{X}_{s+1:})) \text{Diag}(\mathbf{e}) \mid \mathcal{F}_s] \mathbf{1} \quad (74)$$

// applying the inequality for quadratic forms and using the  $\mathcal{F}_s$ -measurability of  $\alpha$

$$\begin{aligned} &= \mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s} + 2\alpha(N-s) \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 \mid \mathcal{F}_s] \\ &\quad - \alpha^2(N-s) \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 (k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \mid \mathcal{F}_s] \\ &\quad - \alpha^2 \left( (N-s) \frac{N-s+1}{2} - (N-s) \right) \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))(y_{s+2} - m_s(\mathbf{x}_{s+2})) k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2}) \mid \mathcal{F}_s] \end{aligned} \quad (75)$$

// using Assumption 6, grouping variance and covariance terms separately

$$\begin{aligned} &= \mathbf{y}_{:s}^\top \mathbf{K}_{:,s;:,s}^{-1} \mathbf{y}_{:s} + 2\alpha(N-s) \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 \mid \mathcal{F}_s] \\ &\quad - \alpha^2(N-s) \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))^2 (k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) + \sigma^2(\mathbf{x}_{s+1})) \mid \mathcal{F}_s] \\ &\quad - \alpha^2(N-s) \frac{N-s-1}{2} \mathbb{E}[(y_{s+1} - m_s(\mathbf{x}_{s+1}))(y_{s+2} - m_s(\mathbf{x}_{s+2})) k_s(\mathbf{x}_{s+1}, \mathbf{x}_{s+2}) \mid \mathcal{F}_s] \end{aligned} \quad (76)$$

// simplifying

□

## I. Utility Proofs

**Lemma 19** (Bounding the relative error). *Let  $D, \hat{D} \in [\mathcal{L}, \mathcal{U}]$ , and assume  $\text{sign}(\mathcal{L}) = \text{sign}(\mathcal{U}) \neq 0$ . Then the relative error of the estimator  $\hat{D}$  can be bounded as*

$$\frac{|D - \hat{D}|}{|D|} \leq \frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{\min(|\mathcal{L}|, |\mathcal{U}|)}.$$

*Proof.* First observe that if  $D_N > \hat{D}$  then  $|D_N - \hat{D}| = D_N - \hat{D} \leq \mathcal{U} - \hat{D}$ . If  $D_N \leq \hat{D}$ , then  $|D_N - \hat{D}| = \hat{D} - D_N \leq \hat{D} - \mathcal{L}$ . Hence,

$$|D_N - \hat{D}| \leq \max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L}).$$

Case  $\mathcal{L} > 0$ : In this case  $|D_N| = D_N \geq \mathcal{L} = |\mathcal{L}|$ , and we obtain for the relative error:

$$\frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{|D_N|} \leq \frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{|\mathcal{L}|}.$$

Case  $\mathcal{U} < 0$ : In that case  $|\mathcal{L}| \geq |D_N| \geq |\mathcal{U}|$ , and the relative error can be bounded as follows.

$$\frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{|D_N|} \leq \frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{|\mathcal{U}|}$$

Since we assumed  $\text{sign}(\mathcal{L}) = \text{sign}(\mathcal{U})$  these were all cases that required consideration. Combining all observations yields

$$\begin{aligned} \frac{|D_N - \hat{D}|}{|D_N|} &\leq \max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L}) \max\left(\frac{1}{|\mathcal{U}|}, \frac{1}{|\mathcal{L}|}\right) \\ &= \frac{\max(\mathcal{U} - \hat{D}, \hat{D} - \mathcal{L})}{\min(|\mathcal{U}|, |\mathcal{L}|)} \end{aligned}$$

□

**Lemma 20.** *The log-determinant of a kernel matrix can be written as a sum of conditional variances.*

$$\log \det \mathbf{K} = \sum_{j=1}^N \log(k_{j-1}(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2(\mathbf{x}_j)) \quad (77)$$

*Proof.* Denote with  $\mathbf{L}$  the Cholesky decomposition of  $\mathbf{K}$ . Then we obtain

$$\log \det \mathbf{K} = \log \det(\mathbf{L}\mathbf{L}^\top) \quad (78)$$

*// using  $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$*

$$= \log\left(\det(\mathbf{L}) \det(\mathbf{L}^\top)\right) \quad (79)$$

*// for square matrices  $\mathbf{B}, \mathbf{C}$ :  $\det(\mathbf{BC}) = \det(\mathbf{B}) \det(\mathbf{C})$*

$$= \log\left(\prod_{j=1}^N L_{jj}^2\right) \quad (80)$$

*// for triangular matrices the determinant is the product of the diagonal elements*

$$= \sum_{j=1}^N 2 \log L_{jj} \quad (81)$$

*// property of log*

With Lemma 24 the result follows. □

**Lemma 21.** *The term  $\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}$  can be written as*

$$\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} = \sum_{n=1}^N \frac{(y_n - m_{n-1}(\mathbf{x}_n))}{k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)}. \quad (82)$$

*Proof.* Define

$$\begin{aligned} \mathbf{k}_j(\mathbf{x}) &:= [k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_j)]^\top \in \mathbb{R}^j \\ \mathbf{k}_{j+1} &:= \mathbf{k}_j(\mathbf{x}_{j+1}) \in \mathbb{R}^j \\ p_j &:= k(\mathbf{x}_j, \mathbf{x}_j) + \sigma^2 - \mathbf{k}_j^\top (\mathbf{K}_{j-1} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_j \\ \boldsymbol{\alpha} &:= (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{n+1} \end{aligned}$$

First note, that using block-matrix inversion we can write

$$(\mathbf{K}_{n+1} + \sigma^2 \mathbf{I})^{-1} = \begin{bmatrix} (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} + \boldsymbol{\alpha} p_{n+1}^{-1} \boldsymbol{\alpha}^\top & -\boldsymbol{\alpha} p_{n+1}^{-1} \\ -\boldsymbol{\alpha}^\top p_{n+1}^{-1} & p_{n+1}^{-1} \end{bmatrix}.$$

This allows to write

$$\begin{aligned}
 & \mathbf{y}_{n+1}^\top (\mathbf{K}_{n+1} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_{n+1} \\
 &= \begin{bmatrix} \mathbf{y}_n^\top & y_{n+1} \end{bmatrix} \begin{bmatrix} (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} + \boldsymbol{\alpha} p_{n+1}^{-1} \boldsymbol{\alpha}^\top & -\boldsymbol{\alpha} p_{n+1}^{-1} \\ -\boldsymbol{\alpha}^\top p_{n+1}^{-1} & p_{n+1}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{y}_n \\ y_{n+1} \end{bmatrix} \\
 & \quad // \text{ using above observation} \\
 &= \begin{bmatrix} \mathbf{y}_n^\top & y_{n+1} \end{bmatrix} \begin{bmatrix} (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_n + \boldsymbol{\alpha} p_{n+1}^{-1} \boldsymbol{\alpha}^\top \mathbf{y}_n - \boldsymbol{\alpha} p_{n+1}^{-1} y_{n+1} \\ -\boldsymbol{\alpha}^\top p_{n+1}^{-1} \mathbf{y}_n + p_{n+1}^{-1} y_{n+1} \end{bmatrix} \\
 & \quad // \text{ simplifying from the right} \\
 &= \mathbf{y}_n^\top (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_n + \mathbf{y}_n^\top \boldsymbol{\alpha} p_{n+1}^{-1} \boldsymbol{\alpha}^\top \mathbf{y}_n - \mathbf{y}_n^\top \boldsymbol{\alpha} p_{n+1}^{-1} y_{n+1} - y_{n+1} \boldsymbol{\alpha}^\top p_{n+1}^{-1} \mathbf{y}_n + y_{n+1} p_{n+1}^{-1} y_{n+1} \\
 & \quad // \text{ simplifying from the left} \\
 &= \mathbf{y}_n^\top (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_n + p_{n+1}^{-1} (\mathbf{y}_n^\top \boldsymbol{\alpha} \boldsymbol{\alpha}^\top \mathbf{y}_n - \mathbf{y}_n^\top \boldsymbol{\alpha} y_{n+1} - y_{n+1} \boldsymbol{\alpha}^\top \mathbf{y}_n + y_{n+1} y_{n+1}) \\
 & \quad // \text{ pulling out } p_{n+1}^{-1} \\
 &= \mathbf{y}_n^\top (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_n + p_{n+1}^{-1} ((\mathbf{y}_n^\top \boldsymbol{\alpha})^2 - 2\mathbf{y}_n^\top \boldsymbol{\alpha} y_{n+1} + y_{n+1}^2) \\
 & \quad // \text{ simplifying} \\
 &= \mathbf{y}_n^\top (\mathbf{K}_n + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_n + p_{n+1}^{-1} (\mathbf{y}_n^\top \boldsymbol{\alpha} - y_{n+1})^2 \\
 & \quad // \text{ simplifying}
 \end{aligned}$$

Now observe that the last addend is indeed the mean square error divided by the posterior variance. By induction the result follows.  $\square$

**Lemma 22.** For all  $t, m \in \mathbb{N}$  with  $1 \leq t + m \leq N$

$$k_{t+m}(\mathbf{x}_a, \mathbf{x}_b) = k_t(\mathbf{x}_a, \mathbf{x}_b) - k_t(\mathbf{x}_a, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_m)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b)$$

where  $k_t(\mathbf{x}_a, \mathbf{x}_b) := k(\mathbf{x}_a, \mathbf{x}_b) - k(\mathbf{x}_a, \mathbf{X}_t) (k(\mathbf{X}_t, \mathbf{X}_t) + \sigma^2(\mathbf{X}_t))^{-1} k(\mathbf{X}_t, \mathbf{x}_b)$  and  $\bar{\mathbf{X}} := \mathbf{X}_{t:t+m}$ .

*Proof.*

$$\begin{aligned}
 & k_{t+m}(\mathbf{x}_a, \mathbf{x}_b) \\
 &= k(\mathbf{x}_a, \mathbf{x}_b) - k(\mathbf{x}_a, \mathbf{X}_{t+m}) \mathbf{A}_{t+m}^{-1} k(\mathbf{X}_{t+m}, \mathbf{x}_b) \\
 & \quad // \text{ by definition} \\
 &= k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \begin{bmatrix} k(\mathbf{X}_t) + \sigma^2 \mathbf{I}_t & k(\mathbf{X}_t, \bar{\mathbf{X}}) \\ k(\bar{\mathbf{X}}, \mathbf{X}_t) & k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t \end{bmatrix}^{-1} \begin{bmatrix} k(\mathbf{X}_t, \mathbf{x}_b) \\ k(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \quad // \text{ in block notation} \\
 &= k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \begin{bmatrix} \mathbf{A}_t & k(\mathbf{X}_t, \bar{\mathbf{X}}) \\ k(\bar{\mathbf{X}}, \mathbf{X}_t) & k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t \end{bmatrix}^{-1} \begin{bmatrix} k(\mathbf{X}_t, \mathbf{x}_b) \\ k(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \quad // \text{ using the definition of } \mathbf{A}_t \\
 &= k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \cdot \\
 & \quad \left[ \begin{array}{cc} \mathbf{A}_t^{-1} + \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}))^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} & -\mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}))^{-1} \\ - (k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}))^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} & (k(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}))^{-1} \end{array} \right]^{-1} \\
 & \quad \begin{bmatrix} k(\mathbf{X}_t, \mathbf{x}_b) \\ k(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \quad // \text{ applying block-matrix inversion} \\
 &= k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \cdot \\
 & \quad \left[ \begin{array}{cc} \mathbf{A}_t^{-1} + \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} & -\mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} \\ - (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} & (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} \end{array} \right]^{-1}
 \end{aligned}$$

$$\begin{aligned}
 & \begin{bmatrix} k(\mathbf{X}_t, \mathbf{x}_b) \\ k(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \text{// applying the definition of } k_t \\
 & = k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \cdot \\
 & \begin{bmatrix} \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) + \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) - \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{x}_b) \\ - (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) + (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \text{// evaluating multiplication with right-most vector} \\
 & = k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \cdot \\
 & \begin{bmatrix} \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) - \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} (k(\bar{\mathbf{X}}, \mathbf{x}_b) - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b)) \\ (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} (k(\bar{\mathbf{X}}, \mathbf{x}_b) - k(\bar{\mathbf{X}}, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b)) \end{bmatrix} \\
 & \text{// rearranging} \\
 & = k(\mathbf{x}_a, \mathbf{x}_b) - \begin{bmatrix} k(\mathbf{x}_a, \mathbf{X}_t) & k(\mathbf{x}_a, \bar{\mathbf{X}}) \end{bmatrix} \cdot \\
 & \begin{bmatrix} \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) - \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) \\ (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) \end{bmatrix} \\
 & \text{// applying the definition of } k_t \\
 & = k(\mathbf{x}_a, \mathbf{x}_b) - k(\mathbf{x}_a, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) \\
 & \quad + k(\mathbf{x}_a, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) - k(\mathbf{x}_a, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) \\
 & \text{// evaluating the vector product} \\
 & = k(\mathbf{x}_a, \mathbf{x}_b) - k(\mathbf{x}_a, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \mathbf{x}_b) - (k(\mathbf{x}_a, \bar{\mathbf{X}}) - k(\mathbf{x}_a, \mathbf{X}_t) \mathbf{A}_t^{-1} k(\mathbf{X}_t, \bar{\mathbf{X}})) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) \\
 & \text{// rearranging} \\
 & = k_t(\mathbf{x}_a, \mathbf{x}_b) - k_t(\mathbf{x}_a, \bar{\mathbf{X}}) (k_t(\bar{\mathbf{X}}) + \sigma^2 \mathbf{I}_t)^{-1} k_t(\bar{\mathbf{X}}, \mathbf{x}_b) \\
 & \text{// applying the definition of } k_t
 \end{aligned}$$

□

**Lemma 23.**

$$\sum_{j=t+1}^n \sum_{i=t_0+1}^{j-1} 1 = (n-t) \left( \frac{n+t+1}{2} - t_0 \right) \quad (83)$$

*Proof.*

$$\sum_{j=t+1}^n \sum_{i=t_0+1}^{j-1} 1 = \sum_{j=t+1}^n (j-1-t_0) \quad (84)$$

$$= \sum_{j=0}^{n-t-1} (j-1-t_0+t+1) \quad (85)$$

$$= (t-t_0)(n-t) + \sum_{j=0}^{n-t-1} j \quad (86)$$

$$= (t-t_0)(n-t) + \frac{(n-t-1)(n-t)}{2} \quad (87)$$

$$= (n-t) \left( \frac{n-t-1}{2} + t-t_0 \right) \quad (88)$$

$$= (n-t) \left( \frac{n+t-1}{2} - t_0 \right) \quad (89)$$

□

**Lemma 24** (Link between the Cholesky and Gaussian process regression). *Denote with  $\mathbf{C}_N$  the Cholesky decomposition of  $\mathbf{K}$ , so that  $\mathbf{C}_N \mathbf{C}_N^\top = \mathbf{K}$ . The  $n$ -th diagonal element of  $\mathbf{C}_N$ , squared, is equivalent to  $k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n)$ :*

$$[\mathbf{C}_N]_{nn}^2 = k_{n-1}(\mathbf{x}_n, \mathbf{x}_n) + \sigma^2(\mathbf{x}_n).$$

*Proof.* With abuse of notation, define  $\mathbf{C}_1 := \sqrt{k(\mathbf{x}_1, \mathbf{x}_1)}$  and

$$\mathbf{C}_N := \begin{bmatrix} \mathbf{C}_{N-1} & \mathbf{0} \\ \mathbf{k}_N^\top \mathbf{C}_{N-1}^{-\top} & \sqrt{k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N} \end{bmatrix}.$$

We will show that the lower triangular matrix  $\mathbf{C}_N$  satisfies  $\mathbf{C}_N \mathbf{C}_N^\top = \mathbf{K}_N + \sigma^2 \mathbf{I}_N$ . Since the Cholesky decomposition is unique (Golub & Van Loan, 2013, Theorem 4.2.7),  $\mathbf{C}_N$  must be the Cholesky decomposition of  $\mathbf{K}$ . Furthermore, by definition of  $\mathbf{C}_N$ ,  $[\mathbf{C}_N]_{NN}^2 = k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N$ . The statement then follows by induction.

To remain within the text margins, define

$$x := \mathbf{k}_N^\top \mathbf{C}_{N-1}^{-\top} \mathbf{C}_{N-1}^{-1} \mathbf{k}_N + k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N.$$

We want to show that  $\mathbf{C}_N \mathbf{C}_N^\top = \mathbf{K}_N + \sigma^2 \mathbf{I}_N$ .

$$\begin{aligned} \mathbf{C}_N \mathbf{C}_N^\top &= \begin{bmatrix} \mathbf{C}_{N-1} & \mathbf{0} \\ \mathbf{k}_N^\top \mathbf{C}_{N-1}^{-\top} & \sqrt{k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N} \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} \mathbf{C}_{N-1}^\top & \mathbf{C}_{N-1}^{-1} \mathbf{k}_N \\ \mathbf{0}^\top & \sqrt{k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{C}_{N-1} \mathbf{C}_{N-1}^\top & \mathbf{C}_{N-1} \mathbf{C}_{N-1}^{-1} \mathbf{k}_N \\ \mathbf{k}_N^\top \mathbf{C}_{N-1}^{-\top} \mathbf{C}_{N-1}^\top & x \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_{N-1} + \sigma^2 \mathbf{I}_{N-1} & \mathbf{k}_N \\ \mathbf{k}_N^\top & x \end{bmatrix} \end{aligned}$$

Also  $x$  can be simplified further.

$$\begin{aligned} x &= \mathbf{k}_N^\top \mathbf{C}_{N-1}^{-\top} \mathbf{C}_{N-1}^{-1} \mathbf{k}_N + k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N \\ &= \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N + k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2 - \mathbf{k}_N^\top (\mathbf{K}_{n-1} + \sigma^2 \mathbf{I}_{n-1})^{-1} \mathbf{k}_N \\ &= k(\mathbf{x}_N, \mathbf{x}_N) + \sigma^2. \end{aligned}$$

□