Adaptive Cholesky Gaussian Processes

Simon Bartels
University of Copenhagen

Kristoffer Stensbo-Smidt
Technical University of Denmark

Pablo Moreno-Muñoz
Technical University of Denmark

Wouter Boomsma
University of Copenhagen

Jes Frellsen
Technical University of Denmark

Søren Hauberg
Technical University of Denmark

Abstract

We present a method to approximate Gaussian process regression models for 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.

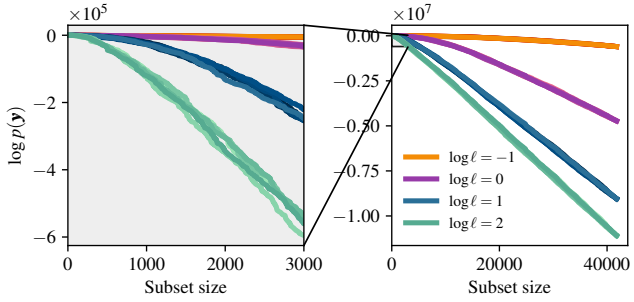


Figure 1: The figure shows the log-marginal likelihood as a function of the size of the training set for five random permutations of the `pm25` dataset. The different colors correspond to different Gaussian process models, using the squared exponential kernel with length scale ℓ. Depending on the model (but little on the permutation), the log-likelihood starts to exhibit a linear trend after processing a certain amount of inputs. More examples can be found in Appendix A.

1 INTRODUCTION

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 and Williams 2006). It has been observed (Chalupka et al. 2013) that the random-subset-of-data approximation can be a hard-to-beat baseline for approximate Gaussian process inference. However, the question of how to choose the size of the subset is non-trivial to answer. Here we make an attempt.

We first make an empirical observation when studying the behavior of the log-marginal likelihood with increasing number of observations. Figure 1 show this progression for a variety of models. We elaborate on this figure in Section 3.1, but for now note that after a certain number of observations, determined by model and dataset, the log-marginal likelihood starts to progress with a linear trend. This suggest 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 and datasets, this point cannot be set in advance but must be estimated on-the-fly.

In this paper, we investigate three main questions, namely 1) how to detect the near linear trend when processing datapoints sequentially, 2) when it is safe to assume that this trend will continue, and 3) how to implement an efficient stopping strategy, that is, without too much overhead to the exact computation. We approach these questions from a (frequentist) probabilistic numerics perspective (Hennig et al. 2015). By treating the dataset as a collection of independent and identically distributed random variables, we provide

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

expected upper and lower bounds on the log-marginal likelihood, which become tight when the above-mentioned linear trend arises. These bounds can be evaluated with little computational overhead by leveraging intermediate computations performed by the Cholesky decomposition that is commonly used for evaluating the log-marginal likelihood. We refer to our method as *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 decomposition. The main difference to previous work is that our algorithm does *not necessarily* look at the whole dataset, which makes it particularly useful in settings where the dataset is so large that even linear-time approximations are not tractable. When a dataset contains a large amount of redundant data, ACGP allows the inference procedure to stop early, saving precious compute—especially when the kernel function is expensive to evaluate.

2 BACKGROUND

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

2.1 Gaussian Process Regression

We start by briefly reviewing Gaussian process (GP) regression models and how they are trained (see Rasmussen and Williams (2006, Chapter 2 and 5.4)). We consider the training dataset $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and outputs $y_n \in \mathbb{R}$. The inputs are collected in the matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$. A GP $f \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ is a collection of random variables defined in terms of a mean function, $m(\boldsymbol{x})$, and a covariance function or *kernel*, $k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$, such that any finite amount of random variables has a Gaussian distribution. Hence, the prior over $\boldsymbol{f} := f(\boldsymbol{X})$ is $\mathcal{N}(\boldsymbol{f}; m(\boldsymbol{X}), \boldsymbol{K}_{\text{ff}})$, where we have used the shorthand notation $\boldsymbol{K}_{\text{ff}} = k(\boldsymbol{X}, \boldsymbol{X})$. Without loss of generality, we assume a zero-mean prior, $m(\cdot) := 0$. We will consider the observations $\boldsymbol{y}$ as being noise-corrupted versions of the function values $\boldsymbol{f}$, and we shall parameterize this corruption through the likelihood function $p(\boldsymbol{y} \mid \boldsymbol{f})$, which for regression tasks is typically assumed to be Gaussian, $p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I})$. For such a model, the posterior over test inputs $\boldsymbol{X}_*$ can be computed in closed-form: $p(\boldsymbol{f}_* \mid \boldsymbol{y}) = \mathcal{N}(\boldsymbol{m}_*, \boldsymbol{S}_*)$, where

$$\begin{aligned} \boldsymbol{m}_* &= k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}^{-1}\boldsymbol{y} \quad \text{and} \\ \boldsymbol{S}_* &= k(\boldsymbol{X}_*, \boldsymbol{X}_*) - k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X}, \boldsymbol{X}_*) \end{aligned}$$

with $\boldsymbol{K} := \boldsymbol{K}_{\text{ff}} + \sigma^2 \boldsymbol{I}$. By marginalizing over the function values of the likelihood distribution, we obtain the marginal likelihood, $p(\boldsymbol{y}) = \int p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}) d\boldsymbol{f}$, the de facto metric for comparing the performance of models in the Bayesian framework. While this integral is not tractable in general, it does have a closed-form solution for Gaussian process regression. Given the GP prior, $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\text{ff}})$, and the Gaussian likelihood, the log-marginal likelihood distribution can be found to be

$$\log p(\boldsymbol{y}) = -\frac{1}{2}\left(\log\det\left[2\pi\boldsymbol{K}\right] + \boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}\right). \qquad (1)$$

Evaluating this expressions costs $\mathcal{O}(N^3)$ operations.

2.2 Background on the Cholesky decomposition

Inverting covariance matrices such as $\boldsymbol{K}$ is a slow and numerically unstable procedure. Therefore, in practice, one typically leverages the Cholesky decomposition of the covariance matrices to compute the inverses. The Cholesky decomposition of a symmetric and positive definite matrix $\boldsymbol{K}$ is the unique, lower[1] triangular matrix $\boldsymbol{L}$ such that $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^\top$ (Golub and 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 ways to compute $\boldsymbol{L}$. The Cholesky of a 1×1 matrix is the square root of the scalar. For larger matrices,

$$\mathrm{chol}[\boldsymbol{K}] = \begin{bmatrix} \mathrm{chol}[\boldsymbol{K}_{:s,:s}] & \boldsymbol{0} \\ \boldsymbol{T} & \mathrm{chol}\left[\boldsymbol{K}_{s:,s:} - \boldsymbol{T}\boldsymbol{T}^\top\right] \end{bmatrix}, \qquad (2)$$

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

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

An important observation is that $\boldsymbol{K}_{s:,s:} - \boldsymbol{T}\boldsymbol{T}^\top$ is the posterior covariance matrix $\boldsymbol{S}_* + \sigma^2\boldsymbol{I}$ when considering $\boldsymbol{X}_{s:}$ as test points. We will make use of this observation in Section 3.5. The log-determinant of $\boldsymbol{K}$ can be obtained from the Cholesky using $\log\det[\boldsymbol{K}] = 2\sum_{n=1}^N \log \boldsymbol{L}_{nn}$. A similar recursive relationship exists between the quadratic form $\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y}$ and $\boldsymbol{L}^{-1}\boldsymbol{y}$ (see appendix, Equation (33)).

2.3 Related work

Much work has gone into tractable approximations to the log-marginal likelihood. Arguably, the most popular approximation methods for GPs are inducing point methods (Quiñonero-Candela and Rasmussen 2005; Snelson and Ghahramani 2006; Titsias 2009; Hensman et al. 2013; Hensman et al. 2017; Shi et al. 2020; Artemev et al. 2021), where

[1] Equivalently, one can define $\boldsymbol{L}$ to be upper triangular such that $\boldsymbol{K} = \boldsymbol{L}^\top\boldsymbol{L}$.

the dataset is approximated through a set of pseudo-data points (inducing points), summarizing information from nearby data. Other approaches involve building approximations to $\boldsymbol{K}$ (Fine and Scheinberg 2001; Rahimi and Recht 2008; Lázaro-Gredilla et al. 2010; Harbrecht et al. 2012; Wilson and Nickisch 2015; Rudi et al. 2017; Wang et al. 2019) or aggregating of distributed local approximations (Gal et al. 2014; Deisenroth and Ng 2015). One may also consider separately the approximation of the quadratic form via linear solvers such as conjugate gradients (Hestenes and Stiefel 1952; Cutajar et al. 2016) and the approximation of the log-determinant (Fitzsimons et al. 2017a; Fitzsimons et al. 2017b; Dong et al. 2017). Another line of research is scaling the hardware (Nguyen et al. 2019).

All above referenced approaches have computational complexity at least $\mathcal{O}(N)$ (with the exception of Hensman et al. (2013) since it uses mini-batching). 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 data requires more computational budget even though more data might not be helpful. This is the main motivation for our work: to derive an approximation algorithm where computational complexity does not depend on redundant data.

The work closest in spirit to the present paper is by 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. This is also the main difference to the work of Hensman et al. (2013). Their bounds allow for mini-batching, but these are inherently deterministic when applied with full batch size.

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:

$$\log p(\boldsymbol{y}) = \sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{y}_{:n}) \,. \tag{3}$$

With this equation in hand, the phenomena in Figure 1 becomes much clearer. The figure shows the value of Equation (3) for an increasing number of observations n. When the plot exhibits a linear trend, it is because the summands $\log p(y_n \mid \boldsymbol{y}_{:n})$ 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, $p(\boldsymbol{y})$, from only a subset of M observations. The cumulative view of the log-marginal likelihood in Equation (3) is our starting point. In particular, we will provide probabilistic 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 yet to be seen. We can then decompose Equation (3) into a sum of terms, which have already been computed, and a remaining sum

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

Recall that we consider the $\boldsymbol{x}_i, y_i$ as independent and identically distributed random variables. Hence, we could estimate $p(\boldsymbol{y}_\mathcal{B} \mid \boldsymbol{y}_\mathcal{A})$ as $(N-M)p(\boldsymbol{y}_\mathcal{A})/M$. Yet this is estimator is biased, since $(\boldsymbol{x}_{M+1}, y_{M+1}), \ldots, (\boldsymbol{x}_N, y_N)$ 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 $s := M - m$, the expected value of $\log p(\boldsymbol{y})$ can be bounded from above and below:

$$\mathbb{E}[\mathcal{L} \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[p(\boldsymbol{y}) \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[\mathcal{U} \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}],$$

and the observations from s to M can be used to estimate $\mathcal{L}$ and $\mathcal{U}$. Figure 2 shows a sketch of our approach. 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

$$\frac{\mathcal{U} - \mathcal{L}}{2\min(|\mathcal{U}|, |\mathcal{L}|)} < r \quad \text{and} \quad \operatorname{sign}(\mathcal{U}) = \operatorname{sign}(\mathcal{L}) \,. \tag{4}$$

If the bounds hold, then the estimator $(\mathcal{L} + \mathcal{U})/2$ achieves the desired relative error (Lemma 21 in appendix). This is in contrast to other approximations, where one specifies a computational budget, rather than a desired accuracy.

3.3 Bounds on the log-marginal likelihood

From Equation (1), we see that the log-marginal likelihood requires computing a log-determinant of the kernel matrix

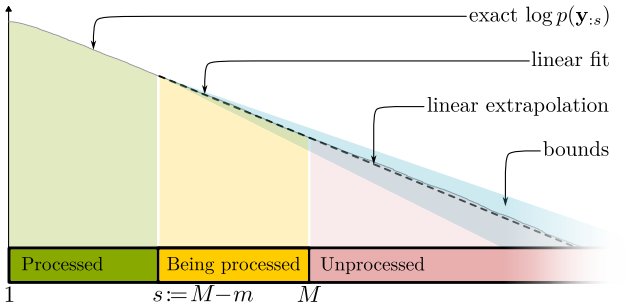


Figure 2: Illustration of how ACGP proceeds during estimation of the full $\log p(\boldsymbol{y})$. The Cholesky decomposition works by processing data in blocks of size m (see Equation (2)), so ACGP computes the log-marginal likelihood in blocks of size m as well. In the illustration, s datapoints have been fully processed, meaning we have the exact $\log p(\boldsymbol{y}_{:s})$ for those. As the next m data are being processed, we can compute the bounds on the *full* $\log p(\boldsymbol{y})$ (*i.e.*, including the unprocessed data) after step 2 of the Cholesky decomposition. If the stopping conditions in Equation (4) are met, we return the linear extrapolation as estimate of $\log p(\boldsymbol{y})$. Theorems 2 and 3 describe the conditions under which this estimate achieves the desired error with high probability.

and a quadratic term. In the following we present upper and lower bounds for both the log-determinant ($\mathcal{U}_\mathrm{D}$ and $\mathcal{L}_\mathrm{D}$, respectively) and the quadratic term ($\mathcal{U}_\mathrm{Q}$ and $\mathcal{L}_\mathrm{Q}$). We will need the posterior equations for the observations, *i.e.*, $p(y_n \mid \boldsymbol{y}_{:n})$, and we will need them as functions of test inputs $\boldsymbol{x}_*$ and $\boldsymbol{x}'_*$. To this end, define

$$\boldsymbol{m}_*^{(n)}(\boldsymbol{x}_*) := k(\boldsymbol{x}_*, \boldsymbol{X}_{:n}) \boldsymbol{K}_{:n,:n}^{-1} \boldsymbol{y}_{:n}$$

and

$$\begin{aligned}\boldsymbol{\Sigma}_*^{(n)}(\boldsymbol{x}_*, \boldsymbol{x}'_*) := k(\boldsymbol{x}_*, \boldsymbol{x}'_*) + \sigma^2 \delta_{\boldsymbol{x}_*, \boldsymbol{x}'_*} \\ - k(\boldsymbol{x}_*, \boldsymbol{X}_{:n}) \boldsymbol{K}_{:n,:n}^{-1} k(\boldsymbol{X}_{:n}, \boldsymbol{x}'_*),\end{aligned}$$

such that $p(y_n \mid \boldsymbol{y}_{:n}) = \mathcal{N}(y_n; \boldsymbol{m}_*^{(n)}(\boldsymbol{x}_n), \boldsymbol{\Sigma}_*^{(n)}(\boldsymbol{x}_n, \boldsymbol{x}_n))$, which allows us to rewrite Equation (3) as

$$\begin{aligned}\log p(\boldsymbol{y}) \propto \sum_{n=1}^{N} \log \boldsymbol{\Sigma}_*^{(n-1)}(\boldsymbol{x}_n, \boldsymbol{x}_n) \\ + \sum_{n=1}^{N} \frac{(y_n - \boldsymbol{m}_*^{(n-1)}(\boldsymbol{x}_n))^2}{\boldsymbol{\Sigma}_*^{(n-1)}(\boldsymbol{x}_n, \boldsymbol{x}_n)} .\end{aligned} \tag{5}$$

This 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 covariance. Therefore define the shorthands

$$\boldsymbol{V} := \mathrm{Diag}\left[\boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{X}_{s:M}, \boldsymbol{X}_{s:M})\right]$$

and

$$\boldsymbol{C} := \sum_{i=1}^{\frac{M}{2}} \boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{x}_{s+2i}, \boldsymbol{x}_{s+2i-1}) \boldsymbol{e}_{2i} \boldsymbol{e}_{2i}^\top ,$$

where $\boldsymbol{e}_j \in \mathbb{R}^m$ is the j-th standard basis vector. The matrix $\boldsymbol{V}$ is simply the diagonal of the posterior covariance matrix $\boldsymbol{\Sigma}_*$. The matrix $\boldsymbol{C}$ consists of every *second* entry of the first off-diagonal of $\boldsymbol{\Sigma}_*$. 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.

3.3.1 Bounds on the log-determinant

Both bounds, lower and upper, use that $\log\det[\boldsymbol{K}] = \log\det[\boldsymbol{K}_{:s,:s}] + \log\det\left[\boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{X}_{s:}, \boldsymbol{X}_{s:})\right]$ 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 (5). It is well-established that, for a fixed input, more observations decrease the posterior variance, and this decrease cannot cross the threshold σ^2 (Rasmussen and 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 $\boldsymbol{X}_{s:M}$ 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:

$$\mathcal{U}_\mathrm{D} = \log\det[\boldsymbol{K}_{:s,:s}] + (N-s)\mu_D, \tag{6}$$

$$\mu_D := \frac{1}{m}\sum_{i=1}^{m} \log(\boldsymbol{V}_{ii}) . \quad /\!/ \textit{ average log posterior variance}$$

To arrive at the lower bound on the log-determinant, we need an expression for how fast the average posterior variance could decrease which is governed by the covariance between inputs. The variable ρ_D measures the average covariance, and we show in Theorem 10 in the appendix that this overestimates the decrease per step with high probability. Since the decrease cannot exceed σ^2, we introduce ψ_D to denote the step which would cross this threshold.

$$\begin{aligned}\mathcal{L}_\mathrm{D} = \log\det[\boldsymbol{K}_{:s,:s}] + (N - \psi_D)\log\sigma^2 \\ + (\psi_D - s)\left(\mu_D - \frac{\psi_D - s - 1}{2}\rho_D\right)\end{aligned} \tag{7}$$

$$\rho_D := \frac{2}{m\sigma^4}\sum_{i=1}^{m} \boldsymbol{C}_{2i,2i}^2 \quad /\!/ \textit{ average square covariance}$$

$$\psi_D := \max\left(N, s + \left\lfloor \frac{\tilde{\mu}_D - \log\sigma^2}{\tilde{\rho}_D} + \frac{1}{2} \right\rfloor\right) \tag{8}$$

$$/\!/ \textit{ steps } \mu_D \textit{ can decrease by } \rho$$

where variables with a tilde refer to a preceding estimate, that is, exchanging the indices M for $M - m$ and s for

$s - m$. Both bounds collapse to the exact solution when $s = N$. The bounds are close when the average covariance between inputs, ρ_D, is small. This occurs for example when the average variance is close to σ^2 since the variance is an upper bound to the covariance. Another case where ρ_D is small is when points are not correlated to begin with.

3.3.2 Bounds on the quadratic term

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

$$\mathcal{L}_\mathrm{Q} = \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + (N-s)\left(\mu_Q - \max(0, \rho_Q)\right) \quad (9)$$

$$\mu_Q := \frac{1}{m}\boldsymbol{r}_*^\top \boldsymbol{V}^{-1}\boldsymbol{r}_* \quad /\!/ \textit{ average calibrated square error}$$

$$\rho_Q := \frac{N-s-1}{2m} \cdot \sum_{j=\frac{s+2}{2}}^{\frac{M}{2}} \frac{\boldsymbol{r}_{*,2j}\boldsymbol{r}_{*,2j-1}\boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{x}_{2j}, \boldsymbol{x}_{2j-1})}{\boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{x}_{2j}, \boldsymbol{x}_{2j})\boldsymbol{\Sigma}_*^{(s)}(\boldsymbol{x}_{2j-1}, \boldsymbol{x}_{2j-1})}$$

$/\!/$ *calibrated error correlation*

Our upper bound arises from the element-wise perspective of Equation (5). We assume that the expected mean square error $(y_n - \boldsymbol{m}_*^{(n-1)}(\boldsymbol{x}_n))^2$ decreases with more observations. 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:

$$\mathcal{U}_\mathrm{Q} = \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + (N-s)\left(\mu_Q + \rho'_Q\right) \quad (10)$$

$$\rho'_Q := \frac{N-s-1}{m}\frac{1}{\sigma^4}\boldsymbol{r}_*^\top \boldsymbol{C}\boldsymbol{V}^{-1}\boldsymbol{C}\boldsymbol{r}_*$$

$/\!/$ *square error correlation*

In the appendix (Theorem 14), we present a tighter bound which uses a similar construction as for the lower bound on the log-determinant, switching the form at a step ψ. Again, the bounds collapse to the true quantity when $s = N$. The bounds will give good estimates when the average covariance between inputs is low or when the model can predict new data well, that is, when $\boldsymbol{r}_*$ is close to 0.

3.4 Validity of bounds and stopping condition

For the upper bound on the quadratic form, we need to make a (technical) assumption. It expresses the intuition that the (expected) mean square error should not increase with more data—a model should not become worse as its training set increases. It is possible to construct counter-examples where this assumption is violated: for example when $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $p(\boldsymbol{f}) = \mathcal{N}(\mathbf{0}, \boldsymbol{K})$, the posterior mean is with high probability no longer zero-mean. However, our experiments in Section 4 indicate that this assumption is not problematic in practice.

Assumption 1. *Assume that*

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

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

Theorem 2. *Assume that* $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ *are independent and identically distributed and that Assumption 1 holds. For any* $s \in \{1, \ldots, N\}$, *the bounds defined in Equations* (6), (7), (9) *and* (10) *hold in expectation:*

$$\mathbb{E}[\mathcal{L}_D \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[\log\det[\boldsymbol{K}] \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[\mathcal{U}_D \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}]$$

and

$$\mathbb{E}[\mathcal{L}_Q \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}] \leq \mathbb{E}[\mathcal{U}_Q \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}]\,.$$

The proof can be found in Appendix G, and a sketch in Appendix E.

Theorem 3. *Let* $r > 0$ *be a desired relative error and set* $\mathcal{U} := -\frac{1}{2}(\mathcal{L}_D + \mathcal{L}_Q + N\log 2\pi)$ *and* $\mathcal{L} := -\frac{1}{2}(\mathcal{U}_D + \mathcal{U}_Q + N\log 2\pi)$. *If the stopping conditions hold, that is,* $\operatorname{sign}(\mathcal{U}) = \operatorname{sign}(\mathcal{L})$ *and Equation* (4) *is true, then* $\log p(\boldsymbol{y})$ *can be estimated from* $(\mathcal{U} + \mathcal{L})/2$ *such that, under the condition* $\mathcal{L}_D \leq \log(\det[\boldsymbol{K}]) \leq \mathcal{U}_D$ *and* $\mathcal{L}_Q \leq \boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \leq \mathcal{U}_Q$, *the relative error is smaller than* r, *formally:*

$$|\log p(\boldsymbol{y}) - (\mathcal{U} + \mathcal{L})/2| \leq r|\log p(\boldsymbol{y})|. \quad (11)$$

The proof follows from Lemma 21 in the appendix.

Theorem 2 is a first step to obtain a probabilistic statement for Equation (11), that is, a statement of the form $\mathbb{P}\left(\left|\frac{\log p(\boldsymbol{y}) - \frac{1}{2}(\mathcal{U} + \epsilon_{\mathcal{U},\delta} + \mathcal{L} - \epsilon_{\mathcal{L},\delta})}{\log p(\boldsymbol{y})}\right| > r\right) \leq \delta$. In earlier work

(Bartels et al. 2023), we have shown that such a statement can be obtained for the log-determinant. Theoretically, we can obtain such a statement using standard concentration inequalities and a union bound over s. In practice, the error guarding constants ϵ 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 $\boldsymbol{\Sigma}_*^{(s)}$ is exactly the downdated matrix in Step 2 of the algorithm outlined in Section 2.2. Similarly, the expressions for the bounds on the quadratic form appear while solving the linear equation system $\boldsymbol{L}^{-1}\boldsymbol{y}$. A slight modification to the Cholesky algorithm is enough to compute these bounds on the fly during the decomposition with little overhead.

The stopping conditions can be checked before or after Step 3 of the Cholesky decomposition (Section 2.2). Here, we explore the former option since Step 3 is the bottleneck due to being less parallelizable than the other steps.

Note that the definition of the bounds does not involve variables $\boldsymbol{x}, y$ 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 Appendix E (Algorithms 1 and 2). For technical reasons, the bounds we use in practice, deviate in some places from the ones presented. We describe the details fully in Appendix E.5. Additionally, we provide a PYTHON implementation of our modified Cholesky decomposition and scripts to replicate the experiments of this paper.[2]

4 EXPERIMENTS

We now examine the 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 $\log p(\boldsymbol{y})$ which wraps OPENBLAS. The details of our experimental setup can be found in Appendix B.

[2]The code is available at the following repository: https://github.com/SimonBartels/acgp

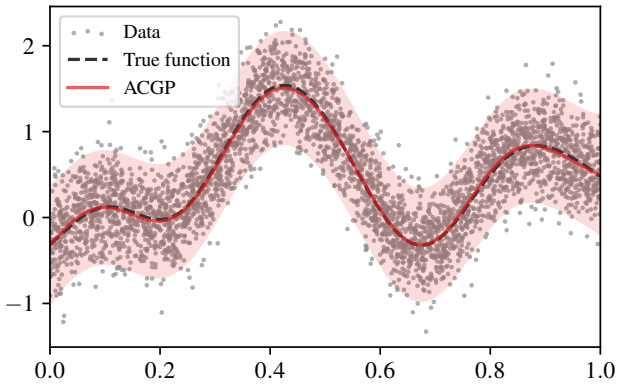


Figure 3: The figure shows one of the sampled functions from the synthetic experiment in Section 4.1 as well as the posterior predictive distribution recovered by ACGP. Since the entire dataset of 10^{12} observations is too large to visualize, we show only the data that were selected by ACGP before stopping; in this case just 4000. The relative error on $\log p(\boldsymbol{y})$ for 10^4 observations was 0.054. Notice how, despite the larger relative error, the posterior process mean closely follows the actual underlying function.

4.1 Performance on synthetic data

ACGP will stop the computation when the posterior covariance matrix of the remaining points conditioned on the processed points is essentially diagonal.This scenario occurs for example when using a squared exponential kernel with long lengthscale and small observational noise on densely sampled dataset.

To test ACGP in this scenario, we sample a function from a GP prior with zero mean and a squared exponential kernel with length scale $\log \ell = -2$. From this function, we uniformly sample 10^{12} observations $(\boldsymbol{x}, y)$ in the interval $[0, 1]$ using an observation noise of $\sigma^2 := 0.1$, that is, $\boldsymbol{y} = f(\boldsymbol{x}) + \mathcal{N}(0, 0.1)$, see Figure 3. This is a scenario where ACGP excels, since it does not need to load the dataset into memory in advance, whereas methods with at least linear complexity cannot even start computation.

The task is to estimate the true $\log p(\boldsymbol{y})$ of the full dataset, and we run ACGP with a relative error of $r = 0.01$ and a blocksize of 1000 to obtain this estimate. Since we cannot evaluate the actual $\log p(\boldsymbol{y})$ for all 10^{12} observations, we use the predicted and actual $\log p(\boldsymbol{y})$ for 10^4 observations as proxy for assessing the performance of ACGP. We repeat the experiment for 10 different random seeds. Recall that ACGP estimates $\mathbb{E}[\log p(\boldsymbol{y}) \mid \boldsymbol{X}_{:s}, \boldsymbol{y}_{:s}]$ as opposed to $\log p(\boldsymbol{y})$, directly. Hence, there are two sources of error for ACGP: the deviation of $\log p(\boldsymbol{y})$ from its expected value and the deviation of the empirical estimates from their expectations.[3] The average $\log p(\boldsymbol{y}_{:10^4})$ is -2699.67 ± 70.81, and

[3]This shows the benefit of developing our theory further, to

thus, due to the relative variance, a relative error of $r = 0.01$ will be hard to achieve. When run on all 10^{12} observations, ACGP stops after processing just 4600 ± 1562 on average, obtaining an actual relative error on the estimate of $\log p(\boldsymbol{y})$ of 0.047 ± 0.034. To decrease this error, one can either decrease the specified relative error of ACGP or increase the blocksize, which will lead to more stable predictions. For the experiments in the remainder of this paper, we choose the latter strategy and set the blocksize to 10^4, which is also better suited for parallel computations.

4.2 Bound quality

The purpose of this section is to demonstrate that with a large enough blocksize m, our estimates are often correct on large datasets. We examine our bounds presented in Section 3 and compare them to those proposed by Artemev et al. (2021, Lemma 2 and Lemma 3) (CGLB). Specifically, for the determinant we compare to their $\mathcal{O}(N)$ upper bound (Artemev et al. 2021, Eq. 11) and their $\log(\det[\boldsymbol{Q}])$ as lower bound. We set the number of inducing inputs M for CGLB to 512, 1024, 2048, and 4096. For ACGP, we define $m := 40 \cdot 256 = 10\,240$ which is the number of cores times the default OpenBLAS blocksize for our machines. We compare both methods using squared exponential kernel (SE) and the Ornstein-Uhlenbeck kernel (OU),

$$k_{\text{SE}}(\boldsymbol{x}, \boldsymbol{z}) := \theta \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\ell^2}\right), \tag{12}$$

$$k_{\text{OU}}(\boldsymbol{x}, \boldsymbol{z}) := \theta \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|}{\ell}\right), \tag{13}$$

where we fix $\sigma^2 := 10^{-3}$ and $\theta := 1$, and we vary ℓ as $\log \ell \in \{-1, 0, 1, 2\}$. 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 and Graff 2019): `metro` and `pm25` (Liang et al. 2015). We chose these datasets in addition as they are of similar size, they are marked as regression tasks and without missing values. We note here that shuffling the datasets does not exactly establish the i.i.d. assumption of Theorem 2. In practice, the results of this section demonstrate that ACGP performs satisfactorily also in the sampling-without-replacement case.

Empirically, CGLB seems to better estimate the quadratic term, whereas ACGP is faster to identify the log-determinant. Figure 4 shows a typical example. Note that, for the quadratic form, the upper bounds tend to be less tight than the lower bounds. Generally, there is no clear winner; sometimes ACGP estimates both quantities faster and sometimes CGLB. See Appendix C for figures on all results.

obtain probably-approximately-correct bounds. Such bounds introduce error-guarding constants to protect against fluctuations.

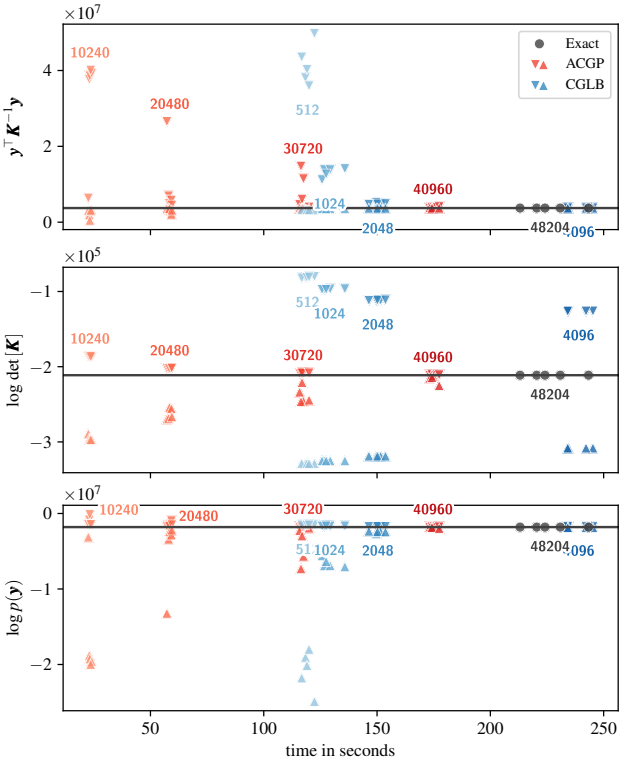


Figure 4: Comparison of the upper and lower bounds for ACGP and CGLB on the `metro` dataset using the OU kernel with a length scale of $\log \ell = 0$ and the time it takes to compute them. The black line indicates the result obtained using exact GP regression with points above and below it marking the upper and lower bounds, respectively. The experiment was repeated five times with different seeds to illustrate the variability in the computation time, shown here as multiple points of the same color. 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 reason why CGLB has more difficulties to approximate the log-determinant is that the bound involves $\operatorname{trace}[\boldsymbol{K} - \boldsymbol{Q}]$ where $\boldsymbol{Q}$ is a low rank approximation to $\boldsymbol{K}$. If $\boldsymbol{K}_{\text{ff}}$ is of high rank, the gap in the 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

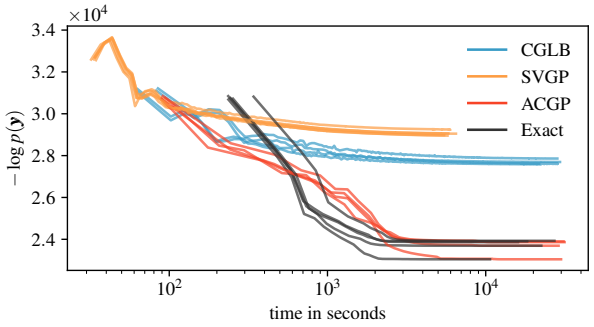
We repeat the hyper-parameter tuning experiments performed by Artemev et al. (2021) using the same set-up, see Appendix B for details. We use the same kernel function, a Matérn $\frac{3}{2}$, and the same optimizer, L-BFGS-B (Liu

and Nocedal 1989), with SciPy (Virtanen et al. 2020) default parameters. Artemev et al. (2021) report their best results using $M = 2048$ inducing inputs. For reference, we also compare against Sparse Variational Gaussian process regression (SGPR) by Titsias (2009) initialized with the same 512, 1024 and 2048 inducing inputs as CGLB. We use root mean square error (RMSE), negative log predictive density (NLPD) and exact, marginal log-likelihood on the training set, $\log p(\boldsymbol{y})$, as performance metrics. The results for all experiments discussed in this section can be found in Appendix C.1. Here, we will focus on the behavior of each method during training.

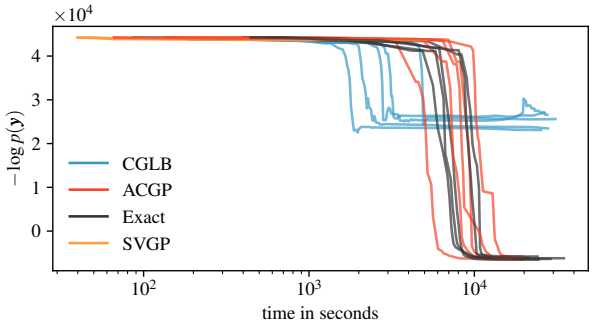
A possible application of ACGP is that an optimizer can decide how precise function evaluations need to be. To explore this possibility, we successively decrease the "relative change in function value" (`ftol`) convergence criterion of L-BFGS-B as $(2/3)^{\text{restart}+1}$ and set this as value for r. With this choice, ACGP does not have any more free parameters than a standard optimizer. The blocksize is a problem independent parameter and it is set to the same value as in Section 4.2.

We explore two different computing environments. For datasets smaller than 20 000 data points, we ran our experiments on a single GPU. 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. The results can be found in Appendix C.1. 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) in that we use only CPUs on machines where the kernel matrix still fits fully into memory. 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 5a shows for the `protein` dataset. This observation is as expected. However, approximation can also hinder fast convergence as Figure 5b 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 worse objective function values. The results for `kin40k` are similar to `protein` and the results for `pm25` are similar to `metro`. These and additional results can be found in Appendix C.2. 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.



(a) `protein` dataset. The iteratively increasing precision may allow ACGP to reach better solutions faster than exact inference at the price of later convergence.



(b) `metro` dataset. Function evaluations with CGLB are generally the fastest at the cost of plateauing at higher objective function values.

Figure 5: Typical examples of the evolution of the exact log marginal likelihood $p(\boldsymbol{y})$ while optimizing hyperparameters. See Appendix C.1 for additional plots for all datasets, as well as for SVGP runs.

5 CONCLUSIONS

The Cholesky decomposition is the de facto way to invert matrices when training Gaussian processes, yet it tends to be considered a black box. However, if one opens this black box, it turns out that the Cholesky decomposition computes the marginal log-likelihood of the full dataset, and, crucially, in intermediate steps, the posteriors of unprocessed training data conditioned on the processed. Making the community aware of this remarkable insight is one of our main contributions of our paper. Our main novelty is to use this insight to bound the (expected) marginal log-likelihood of the full dataset from only a subset. With only small modifications to this classic matrix decomposition, we can use these upper and lower bounds to stop the decomposition before all observations have been processed. This has the practical benefit that the kernel matrix $\boldsymbol{K}$ does not have to computed prior to performing the decomposition, but can rather be computed on-the-fly.

Empirical results indicate that the approach carries significant promise. 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 existing tools.

Acknowledgements

Shout-out to Damien Garreau for a substantial amount of suggestions for this paper. Further, 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 Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606, NNF20OC0065611). 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, 42062) from VILLUM FONDEN, from the Danish Ministry of Education and Science, and from Digital Pilot Hub and Skylab Digital. The authors acknowledge the Pioneer Centre for AI, DNRF grant P1.

References

Artemev, A, DR Burt, and M van der Wilk (2021). "Tighter Bounds on the Log Marginal Likelihood of Gaussian Process Regression Using Conjugate Gradients". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M Meila and T Zhang. Vol. 139. Proceedings of Machine Learning Research, pp. 362–372.

Bartels, S, W Boomsma, J Frellsen, and D Garreau (2023). "Kernel-Matrix Determinant Estimates from stopped Cholesky Decomposition". In: *journal of machine learning research* to appear. 2107.10587.

Camachol, R (1998). "Inducing models of human control skills". In: *Machine Learning: ECML-98*. Ed. by C Nédellec and C Rouveirol, pp. 107–118.

Chalupka, K, Williams, C. K. I., and I Murray (2013). "A Framework for Evaluating Approximation Methods for Gaussian Process Regression". In: *Journal of Machine Learning Research* 14.1, pp. 333–350.

Cutajar, K, M Osborne, J Cunningham, and M Filippone (2016). "Preconditioning Kernel Matrices". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by MF Balcan and KQ Weinberger. Vol. 48. Proceedings of Machine Learning Research, pp. 2529–2538.

Deisenroth, M and JW Ng (2015). "Distributed Gaussian Processes". In: *International Conference on Machine Learning (ICML)*, pp. 1481–1490.

Dong, K, D Eriksson, H Nickisch, D Bindel, and AG Wilson (2017). "Scalable Log Determinants for Gaussian Process Kernel Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 6330–6340.

Dua, D and C Graff (2019). *UCI Machine Learning Repository*.

Fan, X, I Grama, and Q Liu (2012). "Hoeffding's inequality for supermartingales". In: *Stochastic Processes and their Applications* 122.10, pp. 3545–3559.

Fanaee-T, H and J Gama (2013). "Event labeling combining ensemble detectors and background knowledge". In: *Progress in Artificial Intelligence*, pp. 1–15.

Fine, S and K Scheinberg (2001). "Efficient SVM Training Using Low-Rank Kernel Representations". In: *Journal of Machine Learning Research* 2, pp. 243–264.

Fitzsimons, J, K Cutajar, M Osborne, S Roberts, and M Filippone (2017a). "Bayesian Inference of Log Determinants". In: *Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, August 11-15, 2017, Sydney, Australia*. Ed. by G Elidan, K Kersting, and AT Ihler.

Fitzsimons, J, D Granziol, et al. (2017b). "Entropic Trace Estimates for Log Determinants". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by M Ceci, J Hollmén, L Todorovski, C Vens, and S Džeroski, pp. 323–338.

Gal, Y, M Van Der Wilk, and CE Rasmussen (2014). "Distributed variational inference in sparse Gaussian process regression and latent variable models". In: *arXiv preprint arXiv:1402.1389*.

George, A, MT Heath, and J Liu (1986). "Parallel Cholesky factorization on a shared-memory multiprocessor". In: *Linear Algebra and its Applications* 77, pp. 165–187.

Golub, G and C Van Loan (2013). *Matrix computations*. 4th ed. Johns Hopkins Univ Pr.

Harbrecht, H, M Peters, and R Schneider (2012). "On the low-rank approximation by the pivoted Cholesky decomposition". In: *Applied Numerical Mathematics* 62.4, pp. 428–440.

Hennig, P, M Osborne, and M Girolami (2015). "Probabilistic numerics and uncertainty in computations". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 471.2179.

Hensman, J, N Durrande, A Solin, et al. (2017). "Variational Fourier Features for Gaussian Processes." In: *J. Mach. Learn. Res.* 18.1, pp. 5537–5588.

Hensman, J, N Fusi, and ND Lawrence (2013). "Gaussian Processes for Big Data". In: *Uncertainty in Artificial Intelligence (UAI)*, pp. 282–290.

Hestenes, M and E Stiefel (1952). "Methods of conjugate gradients for solving linear systems". In: *Journal of Research of the National Bureau of Standards* 49.6, pp. 409–436.

Kim, H and YW Teh (2018). "Scaling up the Automatic Statistician: Scalable Structure Discovery using Gaussian

Processes". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by A Storkey and F Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research, pp. 575–584.

Lázaro-Gredilla, M, J Quiñonero-Candela, CE Rasmussen, and AR Figueiras-Vidal (2010). "Sparse Spectrum Gaussian Process Regression". In: *Journal of Machine Learning Research* 11, pp. 1865–1881.

Liang, X, T Zou, et al. (2015). "Assessing Beijing's $PM_{2.5}$ pollution: severity, weather impact, APEC and winter heating". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2182, p. 20150257.

Liu, DC and J Nocedal (1989). "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.1, pp. 503–528.

Nguyen, DT, M Filippone, and P Michiardi (2019). "Exact Gaussian Process Regression with Distributed Computations". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 1286–1295.

Paszke, A, S Gross, et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H Wallach, H Larochelle, et al., pp. 8024–8035.

Quiñonero-Candela, J and C Rasmussen (2005). "A unifying view of sparse approximate Gaussian process regression". In: *J of Machine Learning Research* 6, pp. 1939–1959.

Rahimi, A and B Recht (2008). "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems*. Ed. by JC Platt, D Koller, Y Singer, and ST Roweis. Vol. 20. Curran Associates, Inc, pp. 1177–1184.

Rasmussen, C and C Williams (2006). *Gaussian Processes for Machine Learning*. MIT.

Rudi, A, L Carratino, and L Rosasco (2017). "FALKON: An Optimal Large Scale Kernel Method". In: *Advances in Neural Information Processing Systems*. Ed. by I Guyon, UV Luxburg, et al. Vol. 30.

Schwaighofer, A and V Tresp (2002). "Transductive and Inductive Methods for Approximate Gaussian Process Regression". In: *Advances in Neural Information Processing Systems*. Ed. by S Becker, S Thrun, and K Obermayer. Vol. 15. MIT Press.

Shi, J, M Titsias, and A Mnih (2020). "Sparse orthogonal variational inference for Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1932–1942.

Snelson, E and Z Ghahramani (2006). "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems* 18, p. 1257.

Titsias, M (2009). "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial intelligence and statistics*. PMLR, pp. 567–574.

Virtanen, P, R Gommers, et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272.

Wang, K, G Pleiss, et al. (2019). "Exact Gaussian processes on a million data points". In: *Advances in Neural Information Processing Systems* 32, pp. 14648–14659.

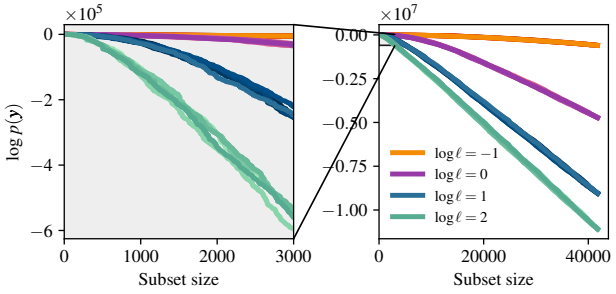
Wang, Q, X Zhang, Y Zhang, and Q Yi (2013). "AUGEM: Automatically generate high performance Dense Linear Algebra kernels on x86 CPUs". In: *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12.

Weiss, SM and N Indurkhya (1995). "Rule-based Machine Learning Methods for Functional Prediction". In: *Journal of Artificial Intelligence Research* 3.1, pp. 383–403.

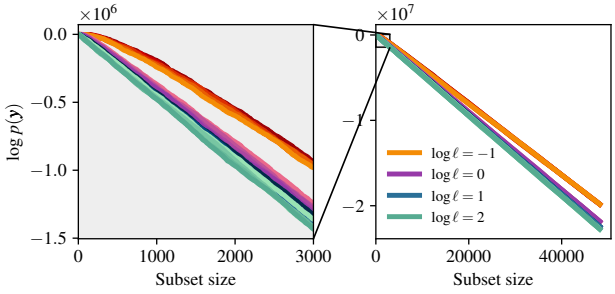
Wilson, A and H Nickisch (2015). "Kernel interpolation for scalable structured Gaussian processes (KISS-GP)". In: *International Conference on Machine Learning*. PMLR, pp. 1775–1784.

A EVOLUTION OF THE LOG-MARGINAL LIKELIHOOD

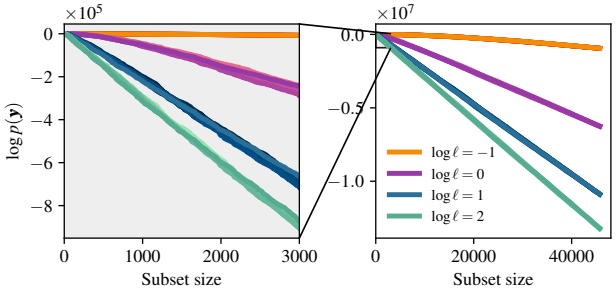
This section contains figures for the progression of the log-marginal likelihood for five different permutations of the same datasets as used in Section 4.2 of the main paper. Figure 6 shows the results for the squared exponential kernel (Equation (14)) with $\theta := 1$ and $\sigma^2 := 10^{-3}$, and Figure 7 shows the results for the Ornstein-Uhlenbeck kernel (Equation (15)) using the same parameters.



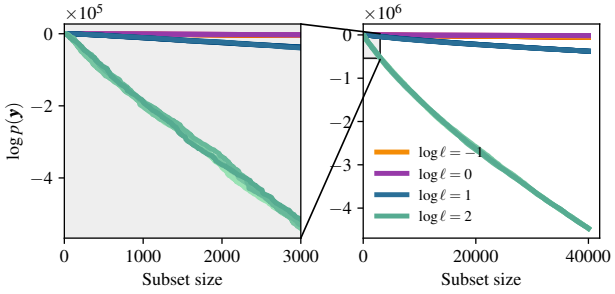
(a) Log-marginal likelihood evolution for `pm25`.



(b) Log-marginal likelihood evolution for `metro`.

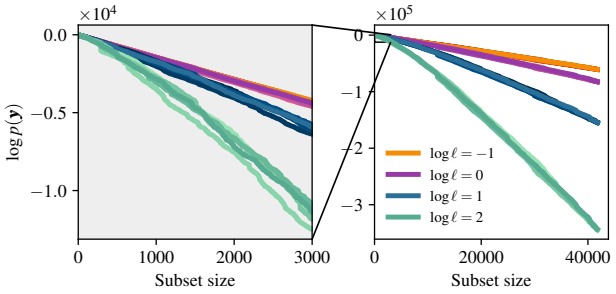


(c) Log-marginal likelihood evolution for `protein`.

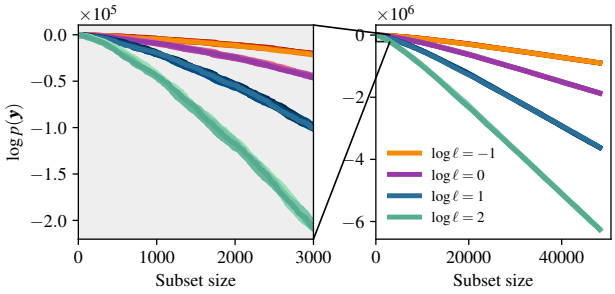


(d) Log-marginal likelihood evolution for `kin40k`.

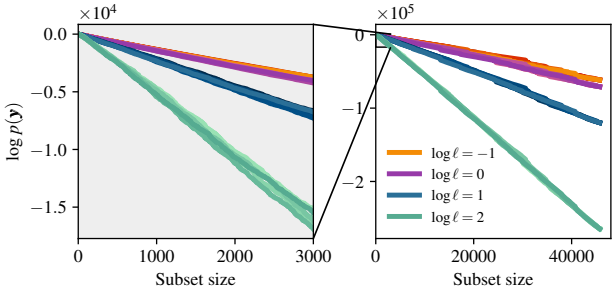
Figure 6: The figure shows the log-marginal likelihood as a function of the size of the training set for the large datasets described in Table 1 using the squared exponential kernel. See Appendix A for a description of the experimental setup.



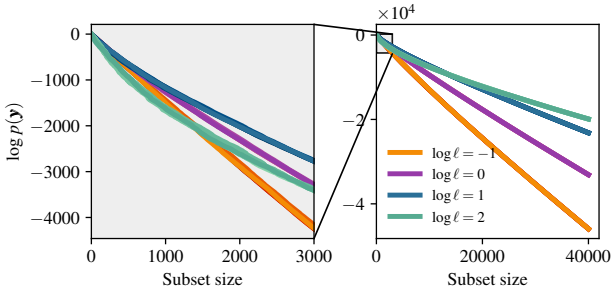
(a) Log-marginal likelihood evolution for `pm25`.



(b) Log-marginal likelihood evolution for `metro`.



(c) Log-marginal likelihood evolution for `protein`.



(d) Log-marginal likelihood evolution for `kin40k`.

Figure 7: The figure shows the log-marginal likelihood as a function of the size of the training set for the large datasets described in Table 1 using the Ornstein-Uhlenbeck kernel. See Appendix A for a description of the experimental setup.

B EXPERIMENTAL DETAILS

Table 1: Overview over all datasets used for the experiments in Section 4. The total dataset size (training and testing) is denoted N and D denotes the dimensionality.

Key	N	D	Source
`bike`	17 379	17	Fanaee-T and Gama (2013). Available at this UCI page.
`elevators`	16 599	18	Camachol (1998).
`kin40k`	40 000	8	Schwaighofer and Tresp (2002).
`metro`	48 204	66	No citation request. Available at this UCI page.
`pm25`	43 824	79	**Liang2015pmDataset**. Available at this UCI page.
`poletelecomm`	15 000	26	Weiss and Indurkhya (1995).
`protein`	45 730	9	No citation request. Available at this UCI page.
`pumadyn`	8192	32	No citation request. Available at this website.

For an overview of the datasets we use, see Table 1. The datasets are all normalized to have zero mean and unit variance for each feature. 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.

B.1 Bound quality experiments

For CGLB, we compute the bounds with varying number of inducing inputs $M := \{512, 1024, 2048, 4096\}$ and measure the time it takes to compute the bounds. For ACGP, we define the blocksize $m := 256 \cdot 40 = 10\,192$ which is the default OPENBLAS block size on our machines times the number of cores. This ensures that the sample size for our bounds is sufficiently large for accurate estimation, and at the same time the number of page-faults should be comparable to the default

Cholesky implementation. 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).

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

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

where we fix $\theta := 1$ and we vary ℓ as $\log \ell \in \{-1, 0, 1, 2\}$. We use a Gaussian likelihood and fix the noise to $\sigma^2 := 10^{-3}$.

B.2 Hyper-parameter tuning

In this section, we describe our experimental setup for the hyper-parameter optimization experiments, which closely follows that of Artemev et al. (2021). We randomly split each dataset into a training set consisting of 2/3 of examples, and a test set consisting of the remaining third. We use a Matérn$\frac{3}{2}$ kernel function and L-BFGS-B as the optimizer with SciPy (Virtanen et al. 2020) default parameters if not specified otherwise. All algorithms are stopped the latest after 2000 optimization steps, after 12 hours of compute time, or when optimization has failed three times. We repeat each experiment five times with a different shuffle of the dataset and report the results in Tables 2 and 3.

For CGLB, it is necessary to decide on a number of inducing inputs. From the results reported by Artemev et al. (2021), it appears that using $M = 2048$ inducing inputs yields the best trade-off in terms of speed and performance, hence we use this value in our experiments. For the exact Cholesky and CGLB, the L-BFGS-B convergence criterion "relative change in function value" (`ftol`) is set to 0.

For ACGP, we need to decide on both the desired relative error, r, as well as the block size m. We successively decrease the optimizer's tolerance `ftol` as $(2/3)^{\text{restart}+1}$ and we set the same value for r. That is, regardless of whether the optimization of ACGP stopped successfully or for abnormal reasons, the optimization restarts aiming for higher precision. The effect of this is that, early in the hyper-parameter optimization, ACGP will stop early, thus providing only an approximation to the optimal hyper-parameter values, but also saving computations. With each restart, ACGP increases the precision, ensuring that we get closer and closer to the optimal hyper-parameter values at the expense of approaching the computational demand of an exact GP. The block size m is set to the same value as for the bound quality experiments, Section 4.2, $40 \cdot 256 = 10\,192$, which is the number of cores times the OpenBLAS block size. This ensures that the sample size for our bounds is sufficiently large for accurate estimation, and at the same time the number of page-faults should be comparable to the default Cholesky implementation. Note that m is a global parameter, independent of the dataset. Hence, natural choices for both r and m are determined by parameters of standard software, which have sensible, machine-dependent default values. ACGP can therefore be considered parameter-free.

Differing from the previous section, we use for ACGP the biased estimator $(N - M) \log p(\boldsymbol{y}_{:M})/M$ instead of $\mathcal{U}/2 + \mathcal{L}/2$ to approximate $\log p(\boldsymbol{y})$ when stopping. Since stopping occurs when log-determinant and quadratic form evolve roughly linearly, the two estimators are not far off each other. The main reason for using the biased estimator is of technical nature: for auto-differentiation, it is easier and faster to implement a custom backward function which can handle the in-place operations of our Cholesky implementation. This custom backward function needs roughly a factor two of the computation of $\log p(\boldsymbol{y})$ whereas the Torch-default needs a factor six. This shows that when comparing to exact inference, auto-differentiation can be disadvantageous and make the Cholesky appear slower than it is. Regarding CGLB, computation time is not dominated by the gradient but only the function evaluation itself.

C ADDITIONAL RESULTS

In this section, we report additional results for both the hyper-parameter tuning experiments (section C.1) as well as plots to show the quality of the bounds on both the log-determinant term, the quadratic term, and the log-marginal likelihood (see Appendix C.3).

C.1 Additional results for hyper-parameter tuning

Denote with N_* the number of test instances, and with μ and σ^2 the mean and variance approximations of a method. As performance metrics we use root mean square error (RMSE)

$$\sqrt{\frac{1}{N^*}\sum_{n=1}^{N^*}(y_n^* - \mu(\boldsymbol{x}_n^*))^2}\ ,$$

negative log predictive density (NLPD)

$$\frac{1}{2N^*}\sum_{n=1}^{N^*}\frac{(y_n^* - \mu(\boldsymbol{x}_n^*))^2}{\sigma^2(\boldsymbol{x}_n^*)} + \log\left(2\pi\sigma^2(\boldsymbol{x}_n^*)\right)\ ,$$

and the negative marginal log likelihood $-\log p(\boldsymbol{y})$. Tables 2 and 3 summarize the results reported for each dataset, averaging over the outcomes of the final optimization step of each repetition. For each metric, we indicate whether a higher (↑) or lower (↓) value indicates a better result.

The results for the exact GP regression are marked in italics to emphasize that these are results we are trying to approach, not to beat. As the other methods are all approximations to the exact GP, there is little hope of achieving better performance. The best result among the approximation methods for each dataset is highlighted in bold.

Table 2: Summary of the CPU hyper-parameter tuning results from Section 4.3. For each metric, we report its final value over the course of optimization. For SVGP, we did not compute the exact marginal log-likelihoods, to save cluster time.

Dataset	**Model**	**RMSE** / 10^{-2} (↓)	**NLPD** / 10^{-1} (↓)	$\log p(\boldsymbol{y})$ / 10^{4} (↑)
metro	*Exact*	*24.01 ± 0.26*	*−12.90 ± 0.17*	*0.6193 ± 0.0243*
	ACGP	**24.11 ± 0.27**	**−12.83 ± 0.23**	**0.6078 ± 0.0250**
	CGLB (1024)	46.45 ± 6.80	8.18 ± 0.85	−2.4526 ± 0.1995
	CGLB (2048)	40.57 ± 7.94	7.45 ± 1.02	−2.5217 ± 0.1745
	CGLB (4096)	35.83 ± 1.04	6.80 ± 0.24	−2.5193 ± 0.0533
	SVGP (1024)	93.28 ± 0.35	13.50 ± 0.04	−4.3892 ± 0.0052
	SVGP (2048)	92.32 ± 0.37	13.40 ± 0.04	−4.3701 ± 0.0053
	SVGP (4096)	88.98 ± 1.56	13.04 ± 0.17	−4.2950 ± 0.0326
pm25	*Exact*	*45.99 ± 2.68*	*3.63 ± 0.64*	*−2.0216 ± 0.0739*
	ACGP	44.45 ± 1.34	**3.25 ± 0.40**	−1.9438 ± 0.0130
	CGLB (1024)	**34.63 ± 2.91**	5.60 ± 0.42	**−1.9259 ± 0.0909**
	CGLB (2048)	44.30 ± 1.94	7.11 ± 0.50	−2.3956 ± 0.0867
	CGLB (4096)	53.06 ± 0.89	7.89 ± 0.13	−2.5681 ± 0.0322
	SVGP (1024)	72.08 ± 7.32	10.97 ± 0.99	−3.1405 ± 0.2014
	SVGP (2048)	56.55 ± 0.96	8.70 ± 0.11	−2.6308 ± 0.0206
	SVGP (4096)	59.30 ± 11.22	8.95 ± 1.66	−2.6904 ± 0.4079
kin40k	*Exact*	*7.42 ± 0.07*	*−12.38 ± 0.05*	*2.0835 ± 0.0041*
	ACGP	**7.42 ± 0.07**	**−12.38 ± 0.05**	**2.0835 ± 0.0041**
	CGLB (1024)	9.17 ± 0.06	−7.13 ± 0.02	1.4726 ± 0.0058
	CGLB (2048)	8.68 ± 0.06	−8.27 ± 0.02	1.6181 ± 0.0048
	CGLB (4096)	8.46 ± 0.06	−9.18 ± 0.02	1.6905 ± 0.0051
	SVGP (1024)	13.94 ± 0.09	−4.20 ± 0.02	0.6124 ± 0.0046
	SVGP (2048)	12.05 ± 0.08	−5.68 ± 0.02	0.8858 ± 0.0047
	SVGP (4096)	10.68 ± 0.07	−7.02 ± 0.02	1.1290 ± 0.0043
protein	*Exact*	*55.85 ± 0.62*	*6.52 ± 0.46*	*−2.3686 ± 0.0331*
	ACGP	**55.55 ± 0.17**	**6.29 ± 0.04**	**−2.3847 ± 0.0092**
	CGLB (1024)	57.58 ± 0.41	8.51 ± 0.05	−2.8194 ± 0.0093
	CGLB (2048)	56.85 ± 0.46	8.31 ± 0.06	−2.7689 ± 0.0107
	CGLB (4096)	56.06 ± 0.44	8.06 ± 0.06	−2.7070 ± 0.0108
	SVGP (1024)	62.21 ± 0.35	9.41 ± 0.04	−2.9994 ± 0.0110
	SVGP (2048)	60.04 ± 0.38	9.00 ± 0.05	−2.9049 ± 0.0105
	SVGP (4096)	58.11 ± 0.43	8.60 ± 0.06	−2.8133 ± 0.0106

Table 3: Summary of the GPU hyper-parameter tuning results from Section 4.3. For each metric, we report its final value over the course of optimization. We did not compute the exact marginal log-likelihoods, to save cluster time.

Dataset	**Model**	**RMSE** / 10^{-2} (↓)	**NLPD** / 10^{-1} (↓)	$\log p(\boldsymbol{y})$ / 10^4 (↑)
bike	*Exact*	*0.09 ± 0.04*	*−50.32 ± 0.08*	*4.9424 ± 0.0073*
	ACGP	**0.21 ± 0.10**	**−50.31 ± 0.03**	**4.9321 ± 0.0019**
	CGLB (1024)	0.53 ± 0.06	−33.24 ± 0.82	3.4946 ± 0.0503
	CGLB (2048)	0.32 ± 0.04	−37.81 ± 0.51	3.8572 ± 0.0487
	CGLB (4096)	0.38 ± 0.15	−41.23 ± 1.08	4.1049 ± 0.1105
	SVGP (1024)	1.27 ± 0.07	−26.63 ± 0.38	2.7446 ± 0.0447
	SVGP (2048)	0.93 ± 0.12	−30.48 ± 0.59	3.0752 ± 0.0516
	SVGP (4096)	0.90 ± 0.23	−32.40 ± 1.51	3.1980 ± 0.1411
poletelecomm	*Exact*	*8.13 ± 0.41*	*−7.81 ± 2.66*	*0.8423 ± 0.0874*
	ACGP	**7.30 ± 0.23**	**−12.32 ± 0.16**	**1.0149 ± 0.0086**
	CGLB (1024)	7.90 ± 0.15	−10.73 ± 0.04	0.8705 ± 0.0052
	CGLB (2048)	7.65 ± 0.17	−11.46 ± 0.06	0.9238 ± 0.0051
	CGLB (4096)	7.39 ± 0.18	−12.12 ± 0.09	0.9822 ± 0.0060
	SVGP (1024)	9.20 ± 0.11	−9.09 ± 0.03	0.7378 ± 0.0054
	SVGP (2048)	8.24 ± 0.15	−10.48 ± 0.05	0.8450 ± 0.0047
	SVGP (4096)	7.53 ± 0.17	−11.81 ± 0.07	0.9543 ± 0.0055
elevators	*Exact*	*35.12 ± 0.36*	*3.78 ± 0.09*	*−0.4690 ± 0.0048*
	ACGP	**34.79 ± 0.24**	**3.70 ± 0.07**	**−0.4653 ± 0.0030**
	CGLB (1024)	35.42 ± 0.40	3.86 ± 0.10	−0.4714 ± 0.0045
	CGLB (2048)	35.26 ± 0.37	3.81 ± 0.10	−0.4699 ± 0.0046
	CGLB (4096)	35.41 ± 0.47	3.86 ± 0.13	−0.4703 ± 0.0051
	SVGP (1024)	35.62 ± 0.37	3.91 ± 0.09	−0.4746 ± 0.0042
	SVGP (2048)	35.42 ± 0.36	3.86 ± 0.09	−0.4715 ± 0.0049
	SVGP (4096)	35.23 ± 0.35	3.81 ± 0.09	−0.4697 ± 0.0048
pumadyn	*Exact*	*22.55 ± 1.03*	*−0.63 ± 0.51*	*0.0088 ± 0.0261*
	ACGP	22.85 ± 0.08	−0.45 ± 0.01	−0.0016 ± 0.0017
	CGLB (1024)	**20.55 ± 0.05**	**−1.63 ± 0.03**	**0.0559 ± 0.0030**
	CGLB (2048)	21.97 ± 2.87	−1.04 ± 1.20	0.0259 ± 0.0620
	CGLB (4096)	20.55 ± 0.06	−1.63 ± 0.03	0.0555 ± 0.0046
	SVGP (1024)	98.65 ± 1.22	14.06 ± 0.12	−0.7749 ± 0.0000
	SVGP (2048)	98.65 ± 1.22	14.06 ± 0.12	−0.7749 ± 0.0000
	SVGP (4096)	98.65 ± 1.22	14.06 ± 0.12	−0.7749 ± 0.0000

C.2 Additional plots for hyper-parameter tuning

The plots for the hyper-parameter optimization are shown in figures 8–31. 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 16–23, we show the root-mean-square error, RMSE, that each methods obtains on the test set at each optimisation step, and figures 24–31 show the same for NLPD. In figures 8–15, we show the log-marginal likelihood, $\log p(\boldsymbol{y})$, that an exact GP would have achieved with the specific set of hyper-parameters at each optimization step for each method.

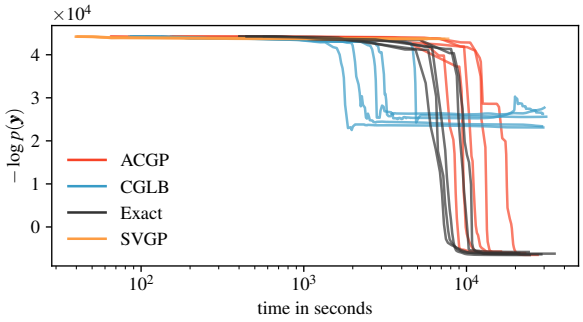


Figure 8: $\log p(\boldsymbol{y})$ for the `metro` dataset.

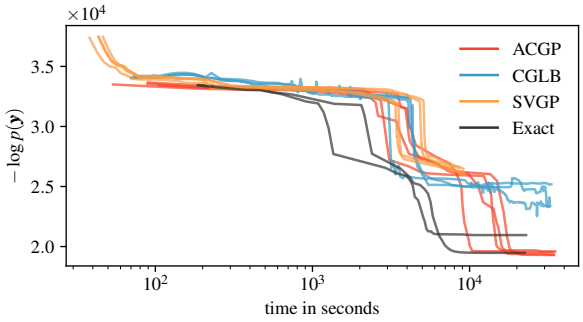


Figure 9: $\log p(\boldsymbol{y})$ for the `pm25` dataset.

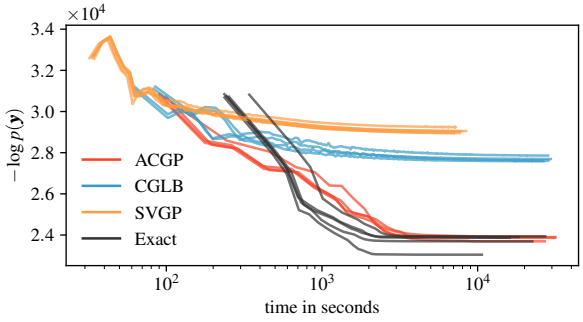


Figure 10: $\log p(\boldsymbol{y})$ for the `protein` dataset.

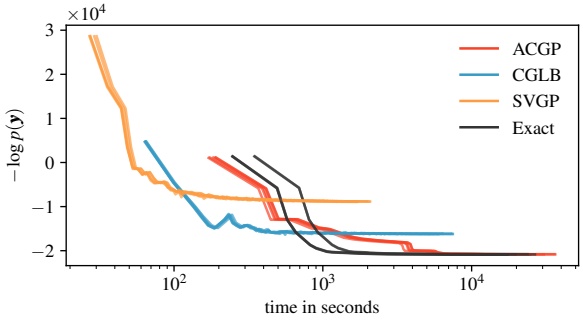


Figure 11: $\log p(\boldsymbol{y})$ for the `kin40k` dataset.

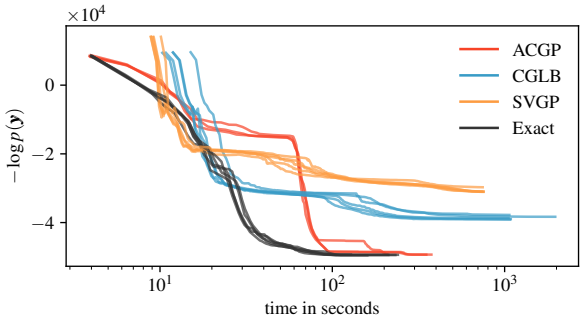


Figure 12: $\log p(\boldsymbol{y})$ for the `bike` dataset.

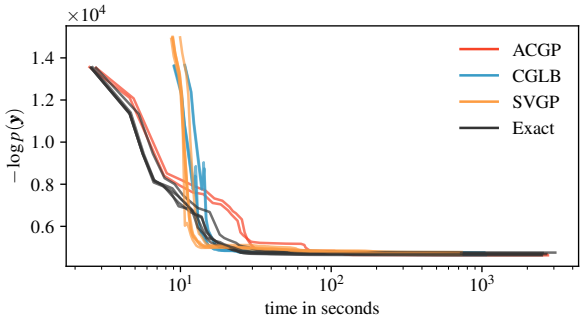


Figure 13: $\log p(\boldsymbol{y})$ for the `elevators` dataset.

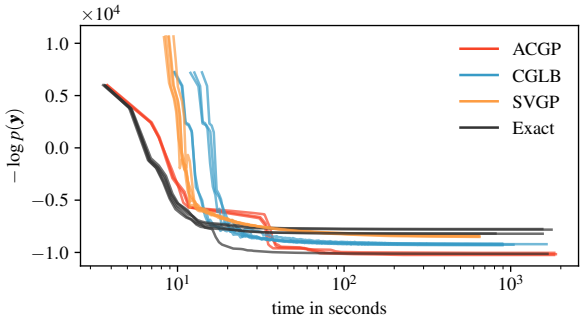


Figure 14: $\log p(\boldsymbol{y})$ for the `pole` dataset.

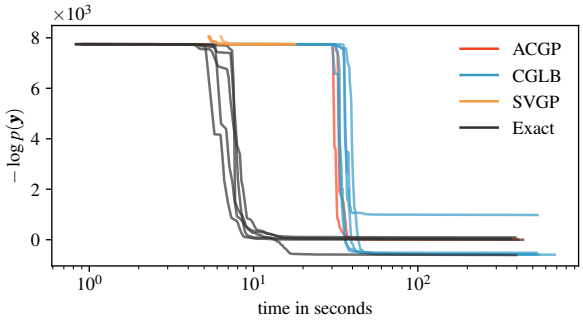


Figure 15: $\log p(\boldsymbol{y})$ for the `pumadyn32nm` dataset.

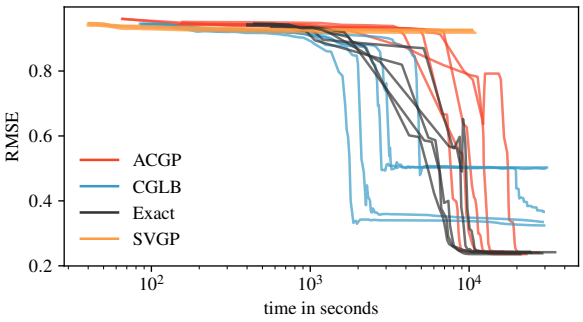


Figure 16: RMSE for the `metro` dataset.

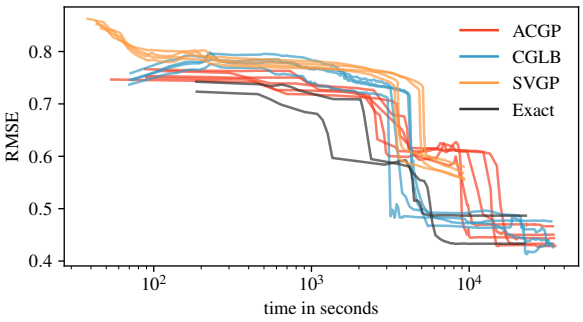


Figure 17: RMSE for the `pm25` dataset.

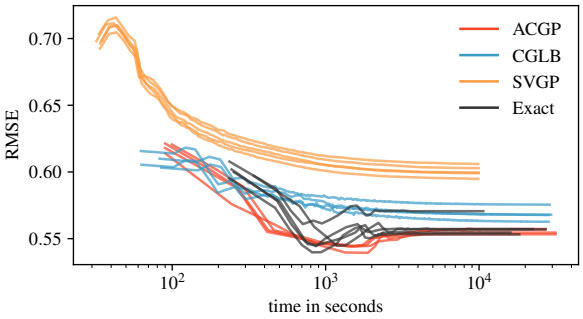


Figure 18: RMSE for the `protein` dataset.

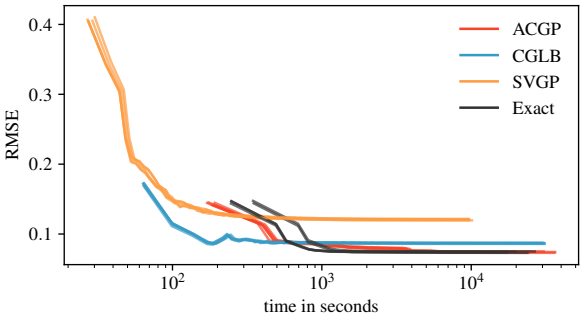


Figure 19: RMSE for the `kin40k` dataset.

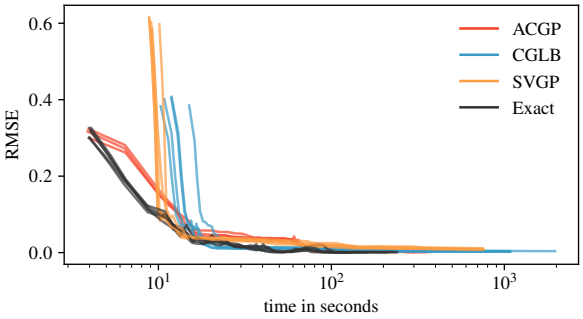


Figure 20: RMSE for the `bike` dataset.

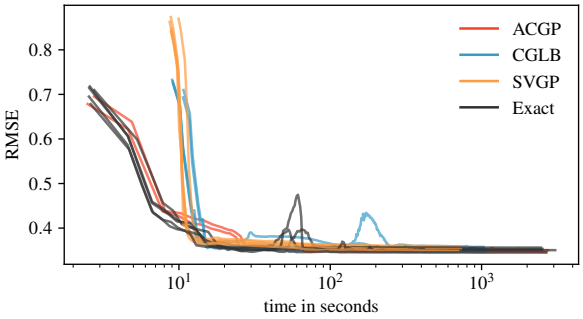


Figure 21: RMSE for the `elevators` dataset.

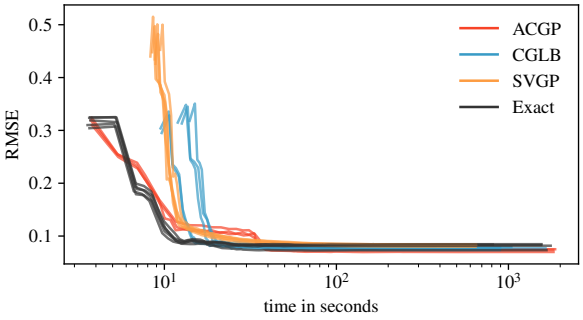


Figure 22: RMSE for the `pole` dataset.

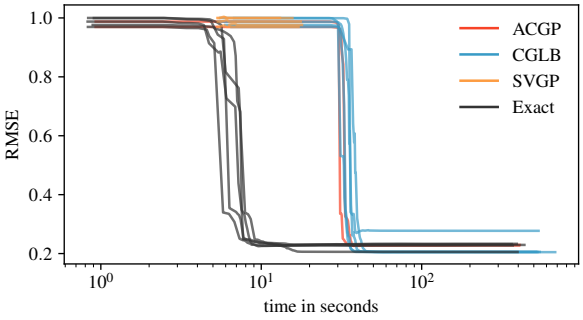


Figure 23: RMSE for the `pumadyn32nm` dataset.

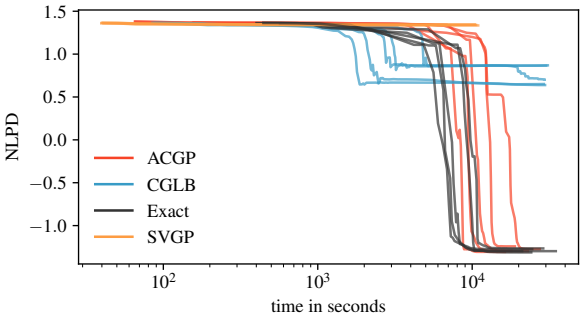


Figure 24: NLPD for the `metro` dataset.

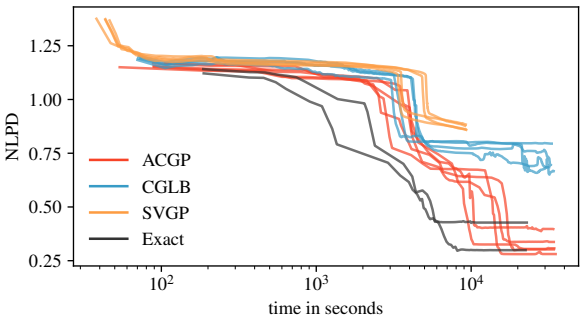


Figure 25: NLPD for the `pm25` dataset.

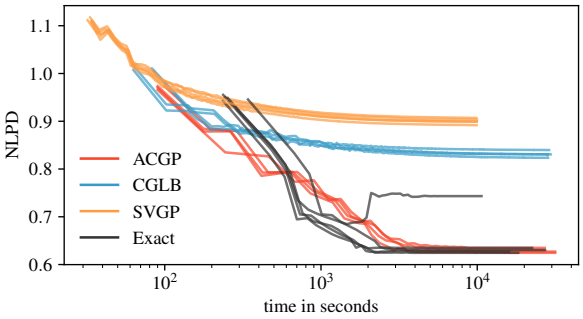


Figure 26: NLPD for the `protein` dataset.

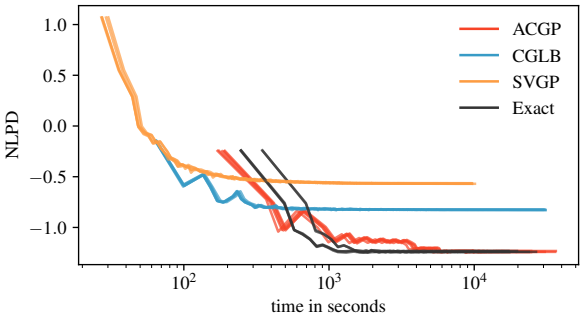


Figure 27: NLPD for the `kin40k` dataset.

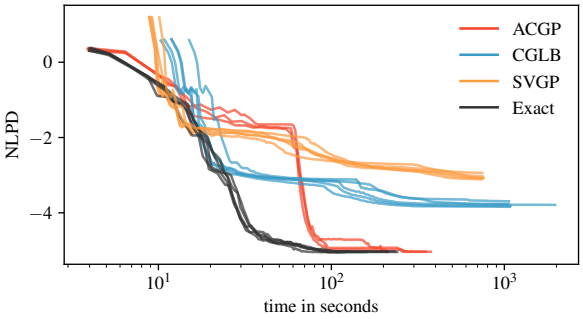


Figure 28: NLPD for the `bike` dataset.

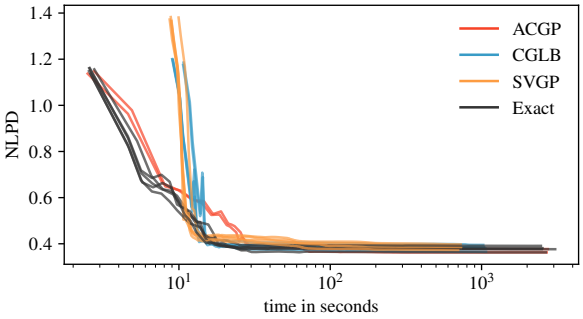


Figure 29: NLPD for the `elevators` dataset.

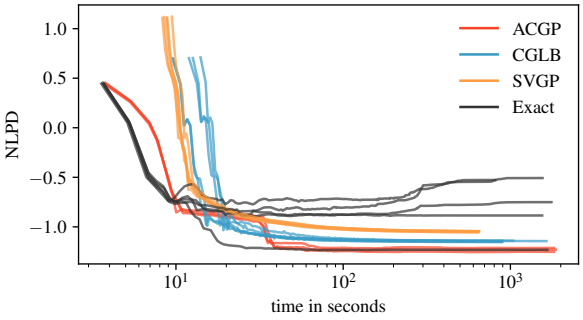


Figure 30: NLPD for the `pole` dataset.

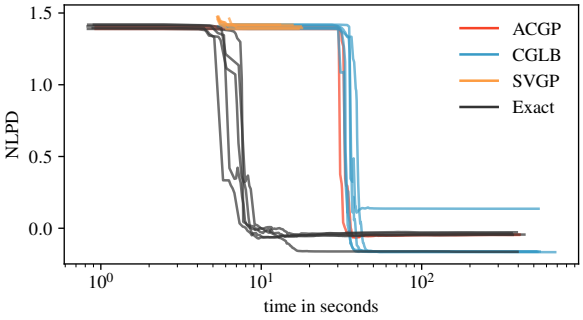
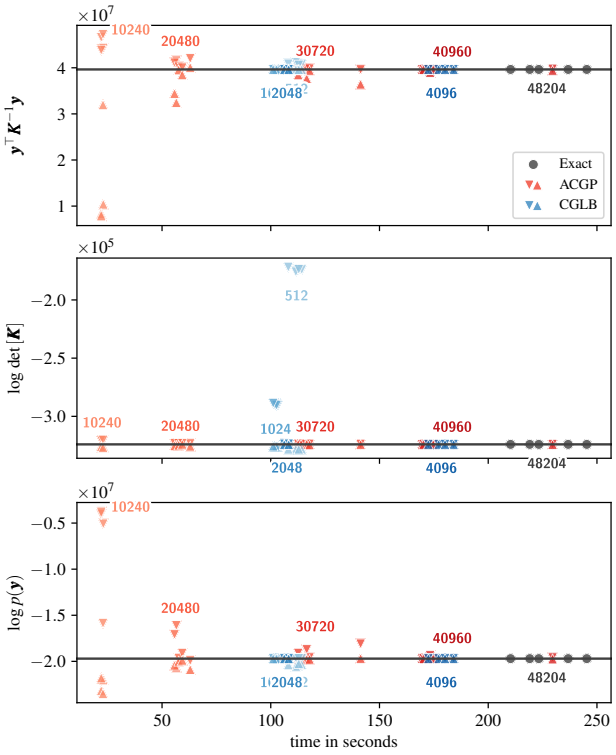


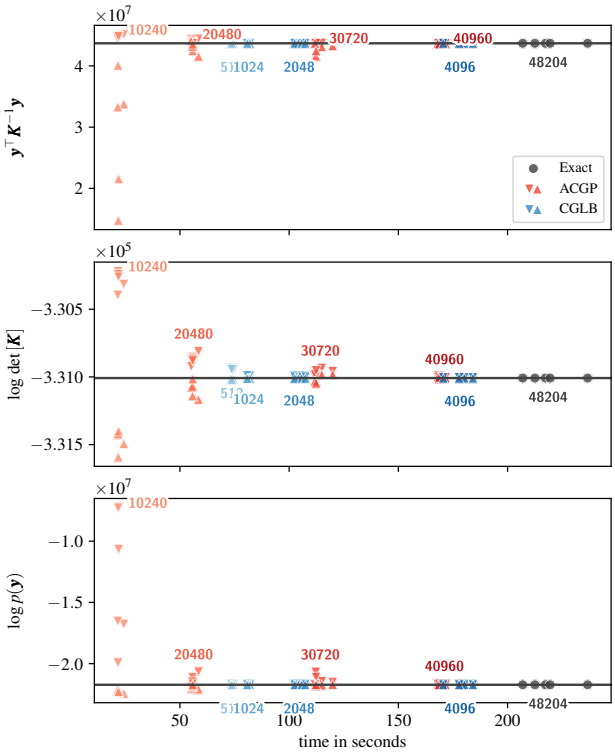
Figure 31: NLPD for the `pumadyn32nm` dataset.

C.3 Additional plots for the bound quality experiments

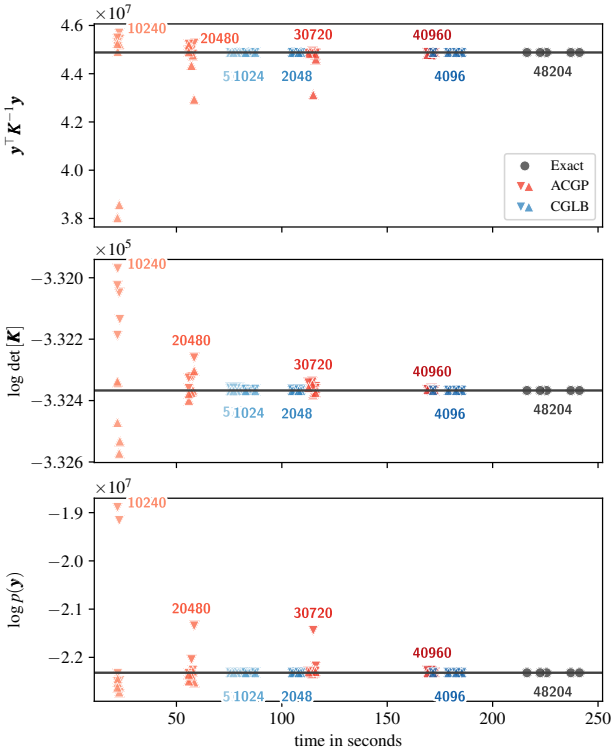
C.3.1 Bounds for experiments on `metro`



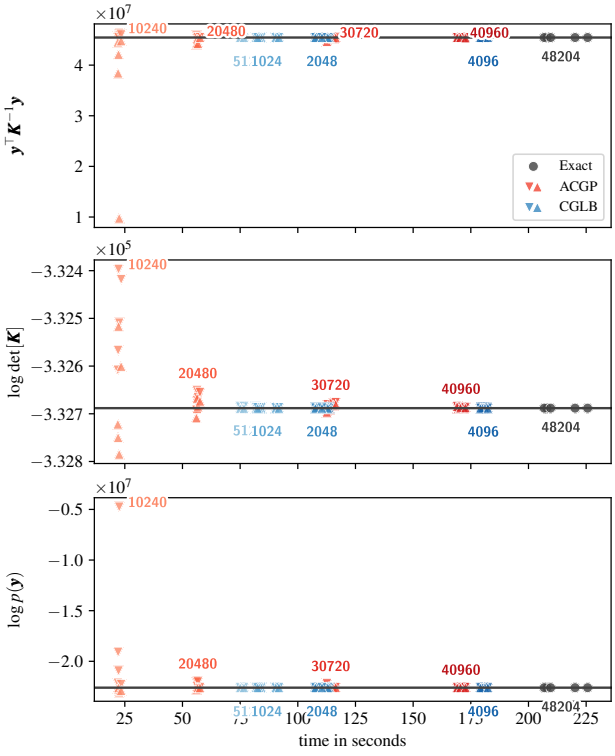
(a) SE kernel, $\log \ell = -1$.



(b) SE kernel, $\log \ell = 0$.

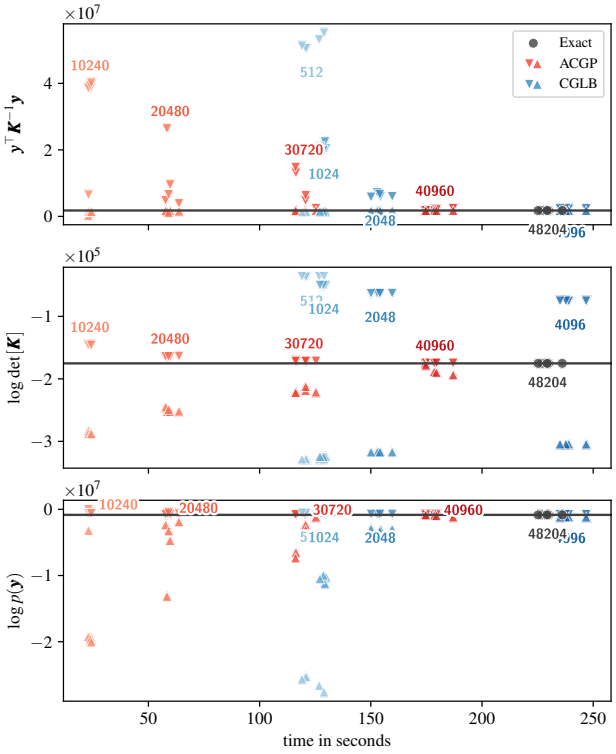


(c) SE kernel, $\log \ell = 1$.

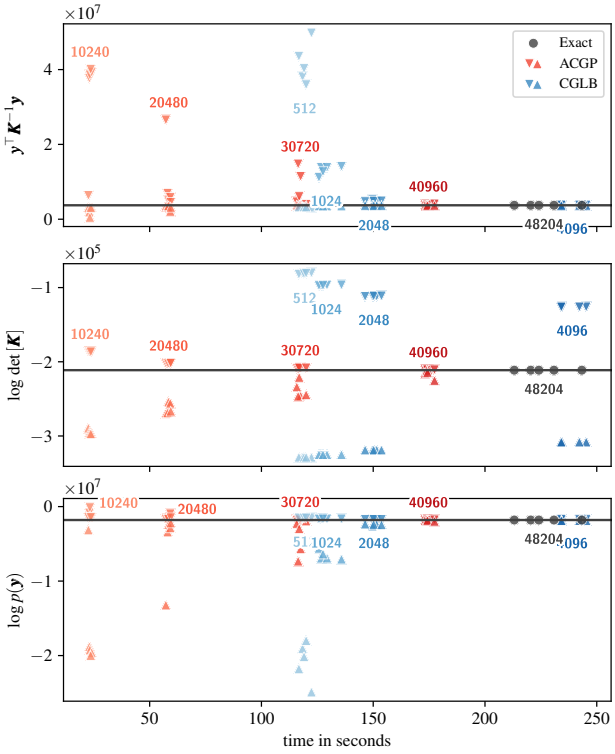


(d) SE kernel, $\log \ell = 2$.

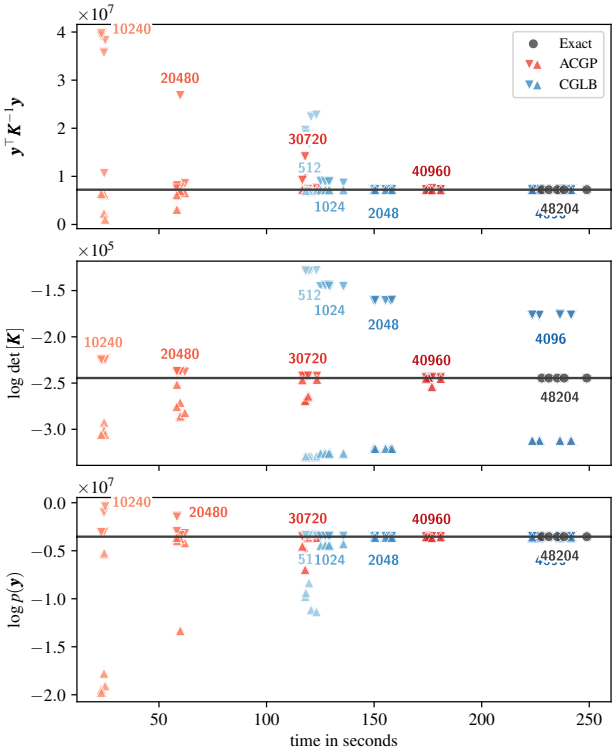
Figure 32: Upper and lower bounds for the `metro` dataset when using a squared exponential (SE) kernel.



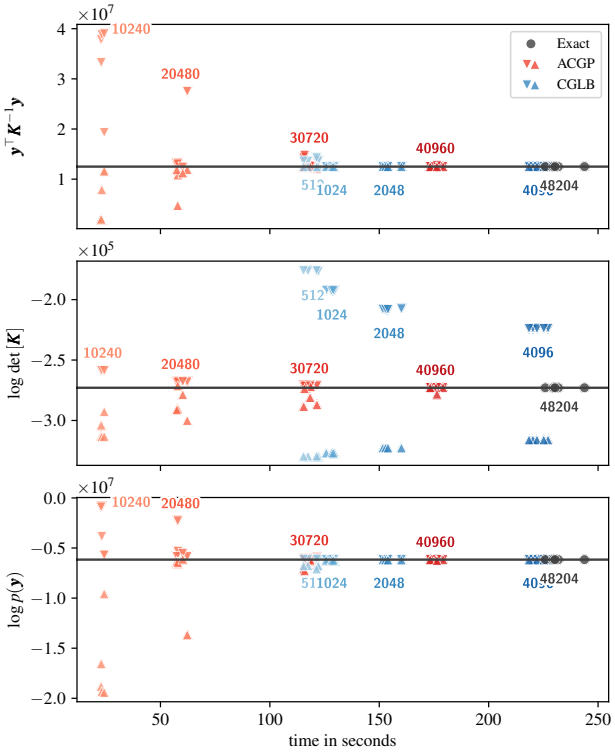
(a) OU kernel, $\log \ell = -1$.



(b) OU kernel, $\log \ell = 0$.



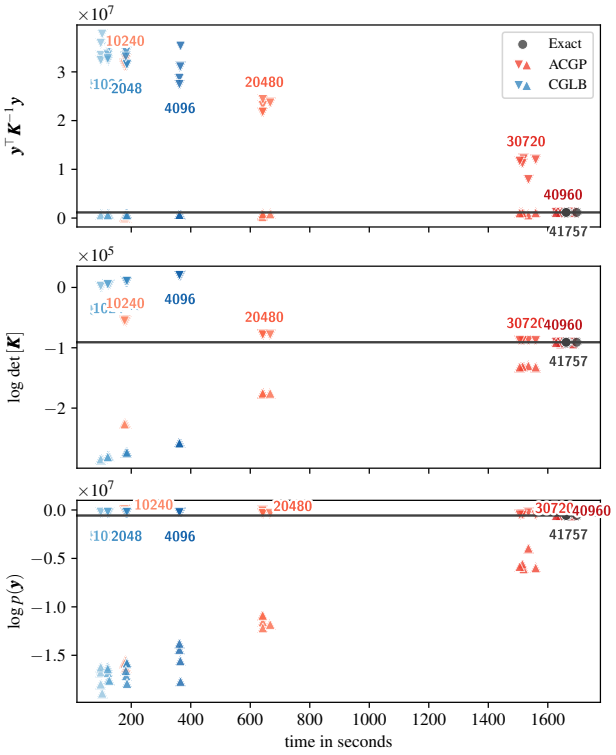
(c) OU kernel, $\log \ell = 1$.



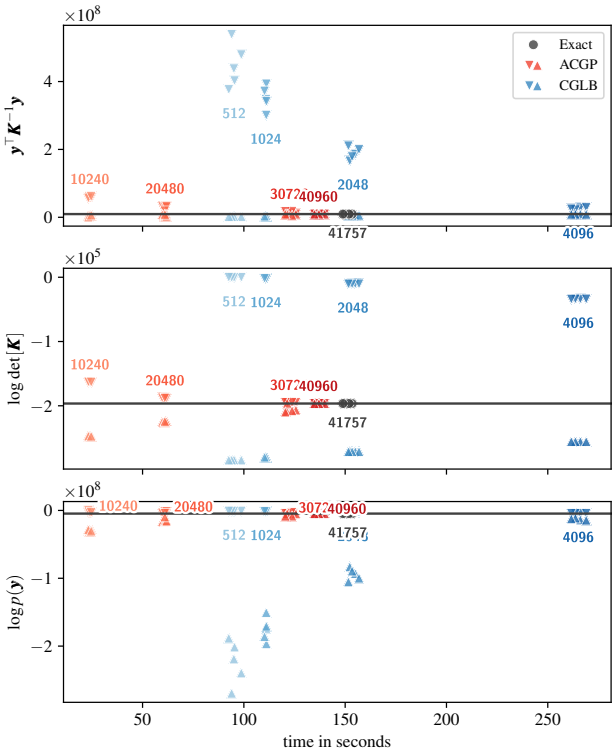
(d) OU kernel, $\log \ell = 2$.

Figure 33: Upper and lower bounds for the `metro` dataset when using a Ornstein-Uhlenbeck (OU) kernel.

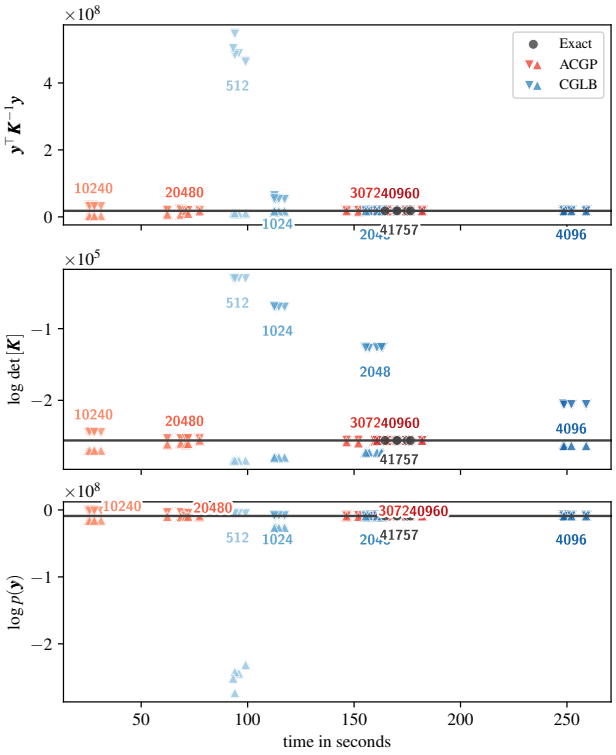
C.3.2 Bounds for experiments on `pm25`



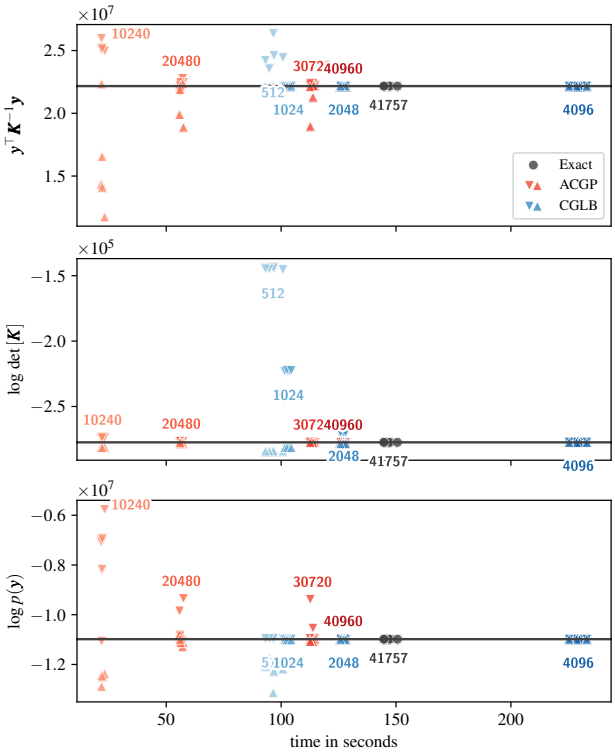
(a) SE kernel, $\log \ell = -1$.



(b) SE kernel, $\log \ell = 0$.

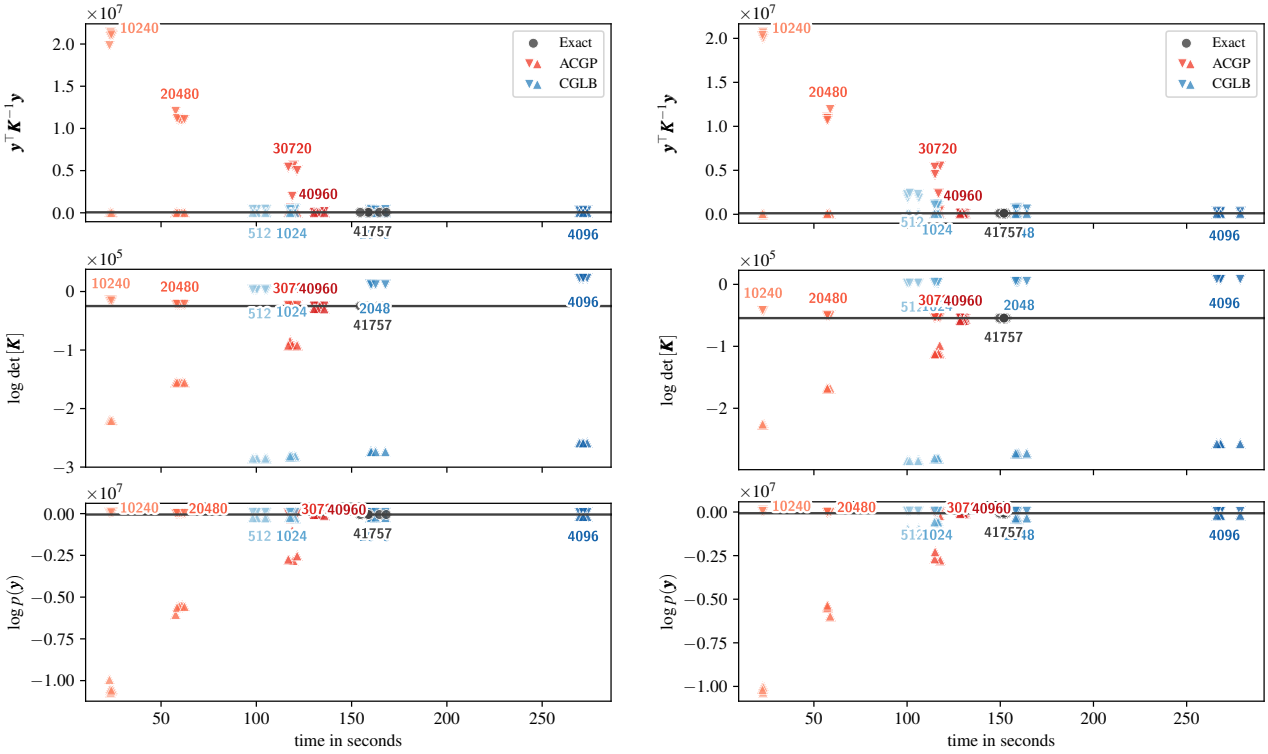


(c) SE kernel, $\log \ell = 1$.



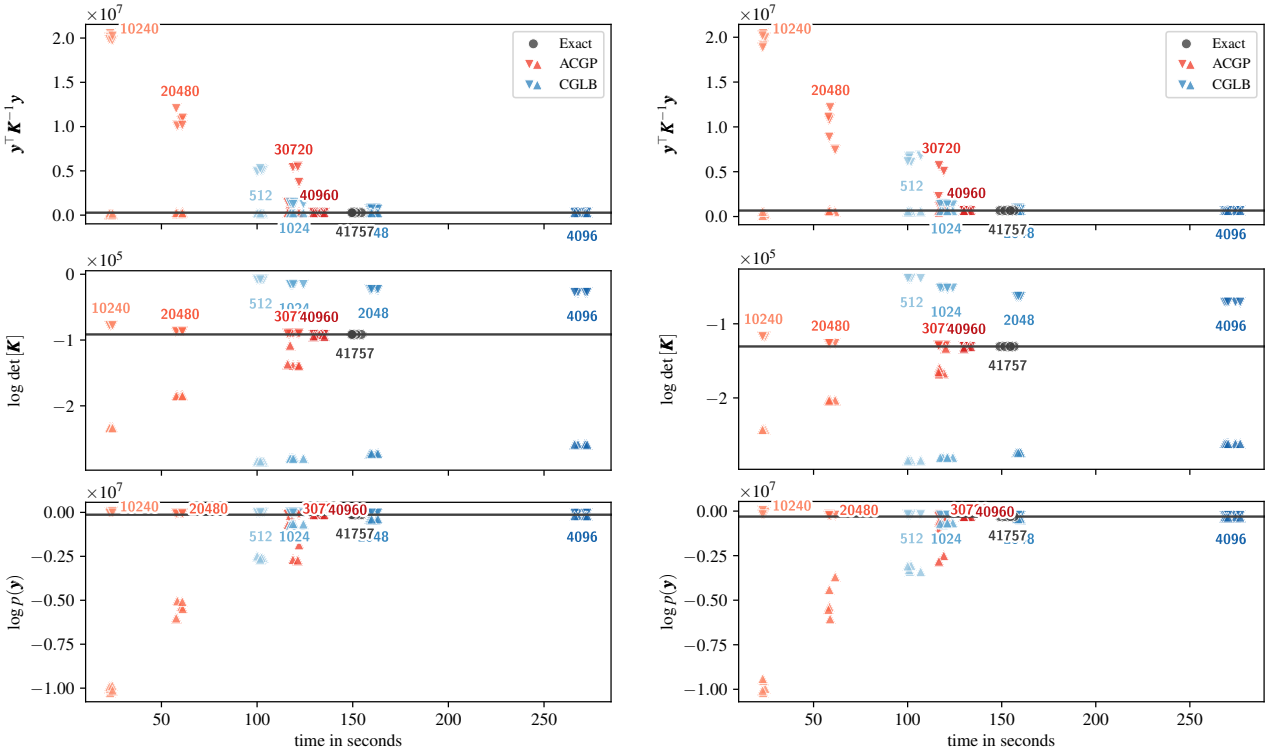
(d) SE kernel, $\log \ell = 2$.

Figure 34: Upper and lower bounds for the `pm25` dataset when using a squared exponential (SE) kernel.



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

(b) OU kernel, $\log \ell = 0$.

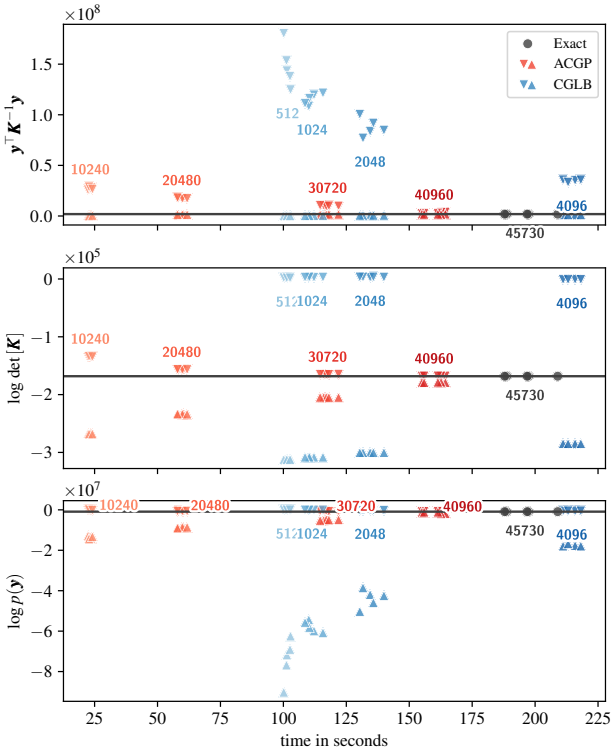


(c) OU kernel, $\log \ell = 1$.

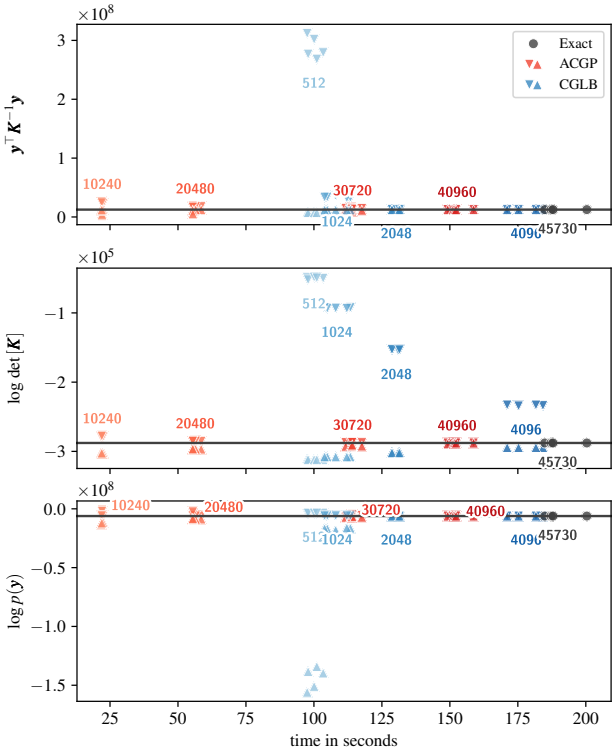
(d) OU kernel, $\log \ell = 2$.

Figure 35: Upper and lower bounds for the `pm25` dataset when using a Ornstein-Uhlenbeck (OU) kernel.

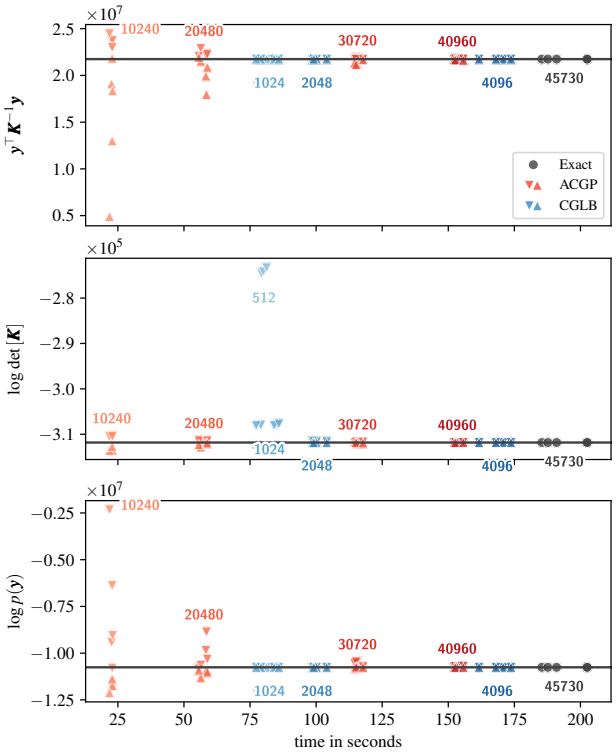
C.3.3 Bounds for experiments on `protein`



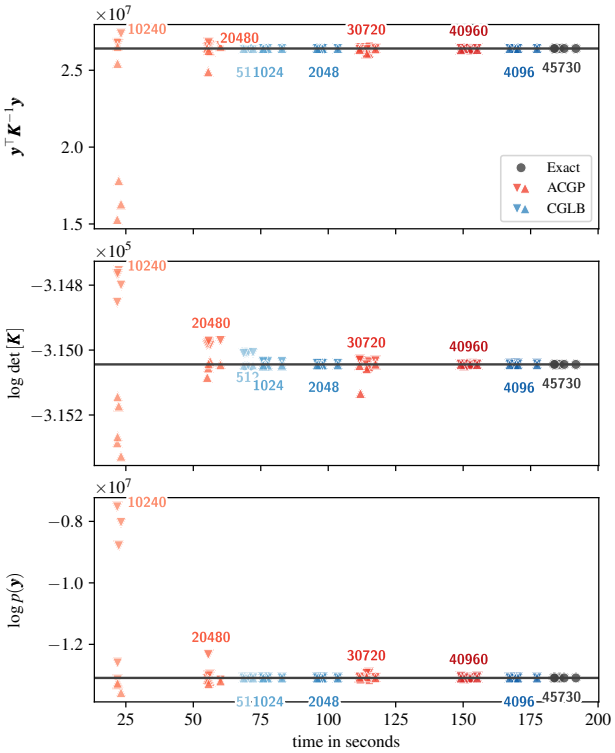
(a) SE kernel, $\log \ell = -1$.



(b) SE kernel, $\log \ell = 0$.

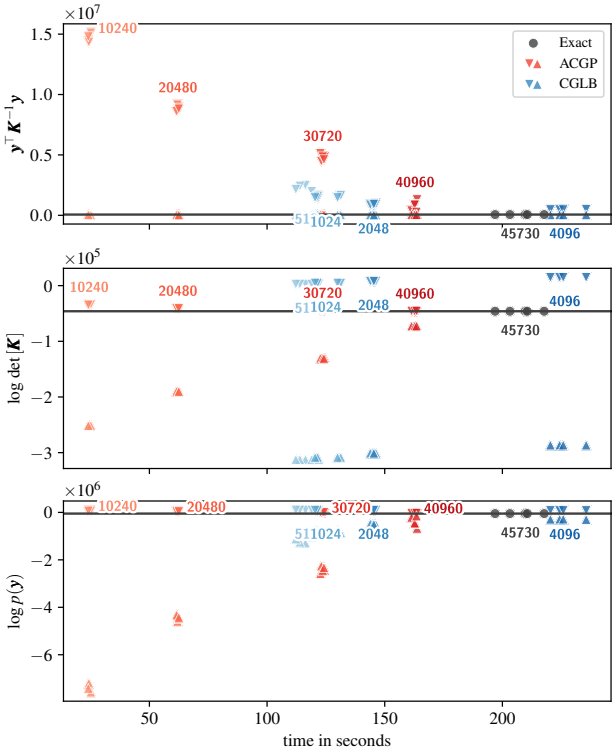


(c) SE kernel, $\log \ell = 1$.

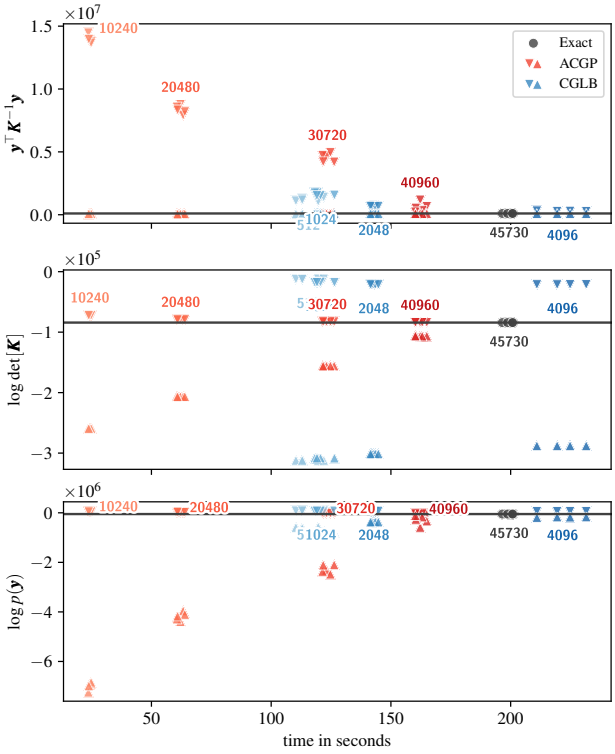


(d) SE kernel, $\log \ell = 2$.

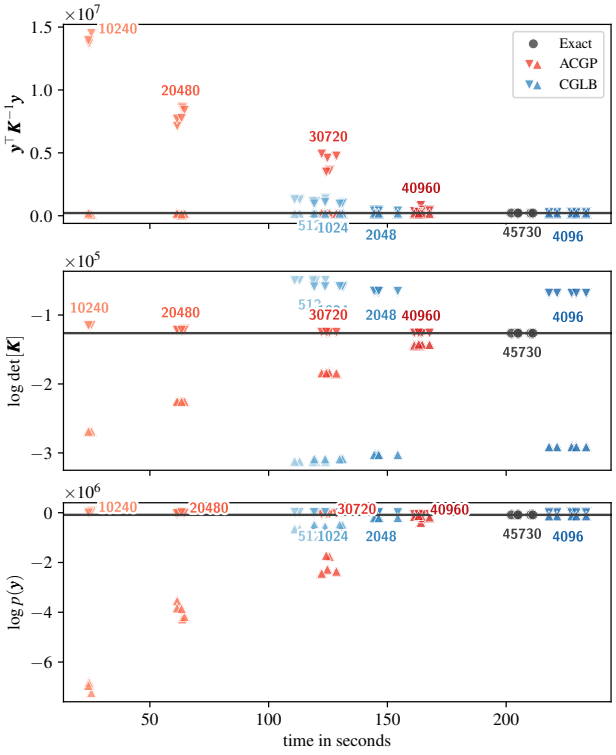
Figure 36: Upper and lower bounds for the `protein` dataset when using a squared exponential (SE) kernel.



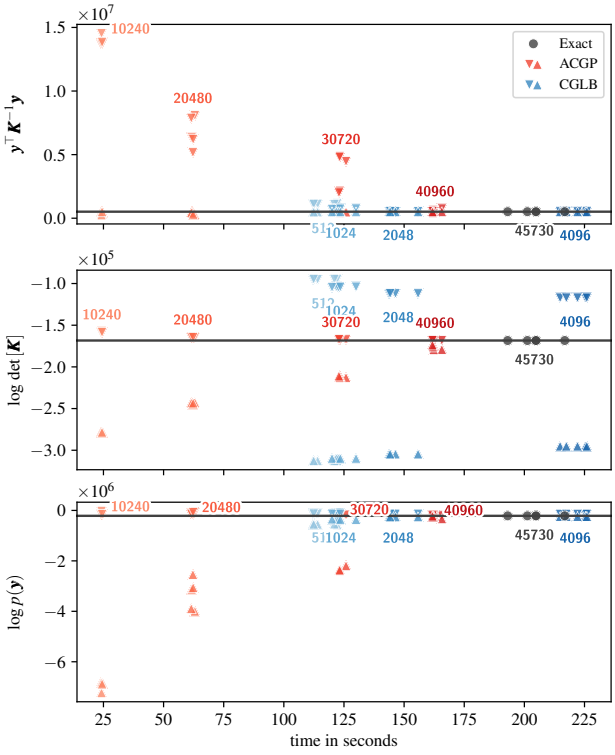
(a) OU kernel, $\log \ell = -1$.



(b) OU kernel, $\log \ell = 0$.



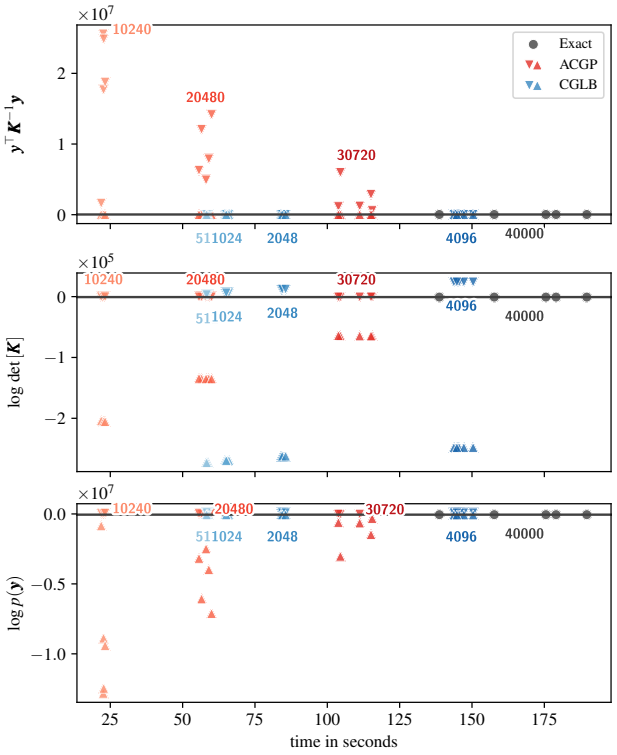
(c) OU kernel, $\log \ell = 1$.



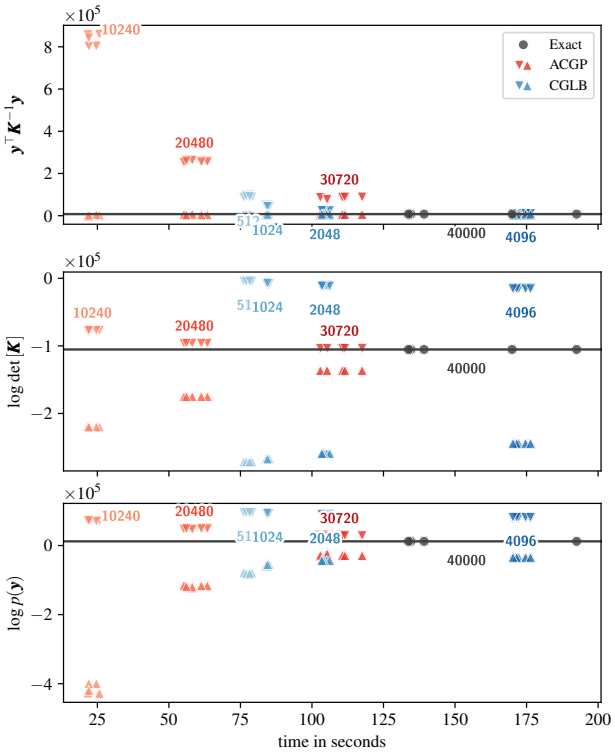
(d) OU kernel, $\log \ell = 2$.

Figure 37: Upper and lower bounds for the `protein` dataset when using a Ornstein-Uhlenbeck (OU) kernel.

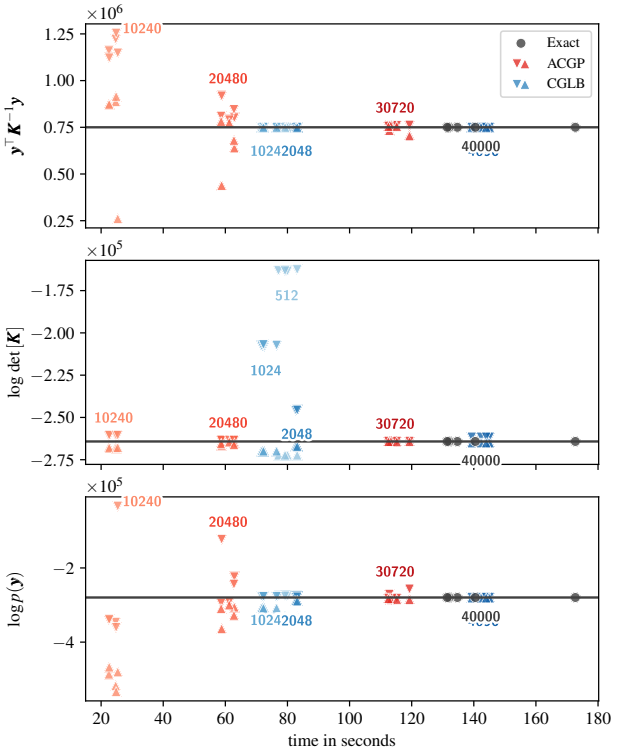
C.3.4 Bounds for experiments on `kin40k`



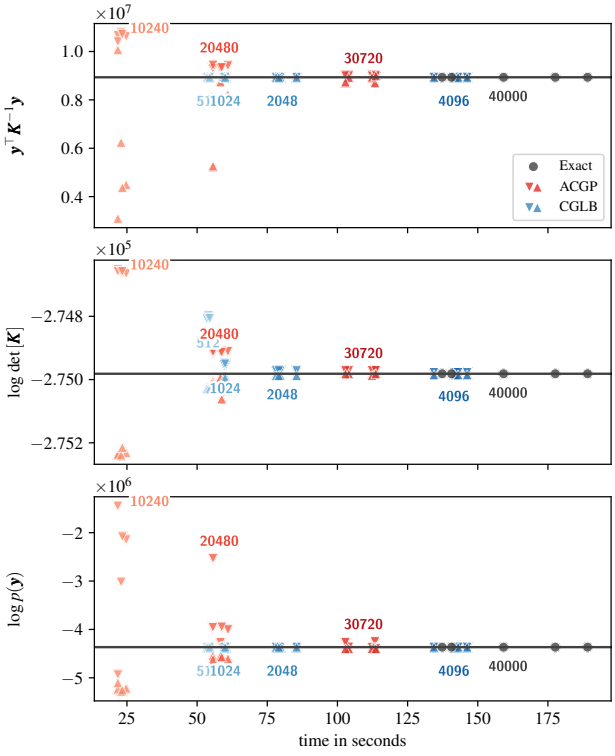
(a) SE kernel, $\log \ell = -1$.



(b) SE kernel, $\log \ell = 0$.

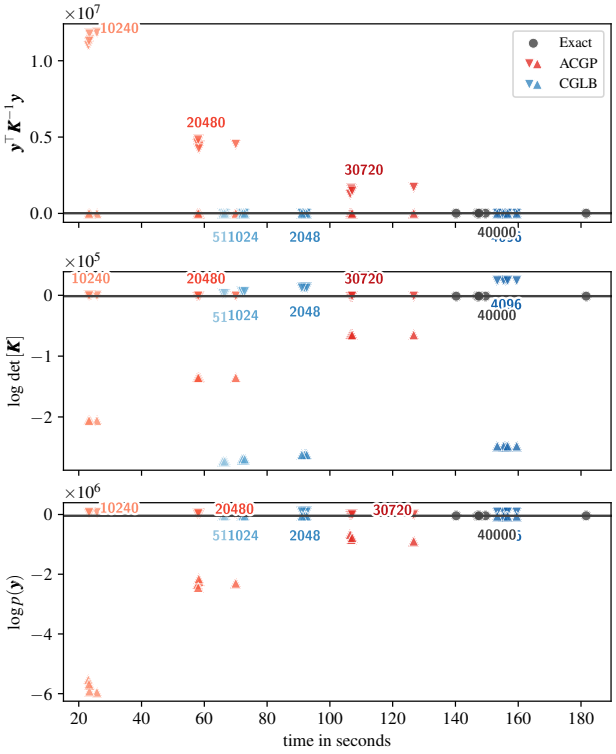


(c) SE kernel, $\log \ell = 1$.

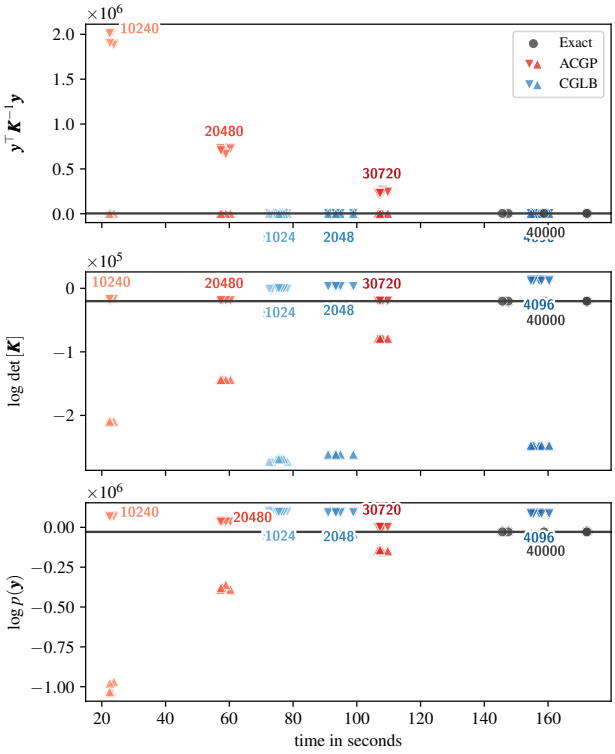


(d) SE kernel, $\log \ell = 2$.

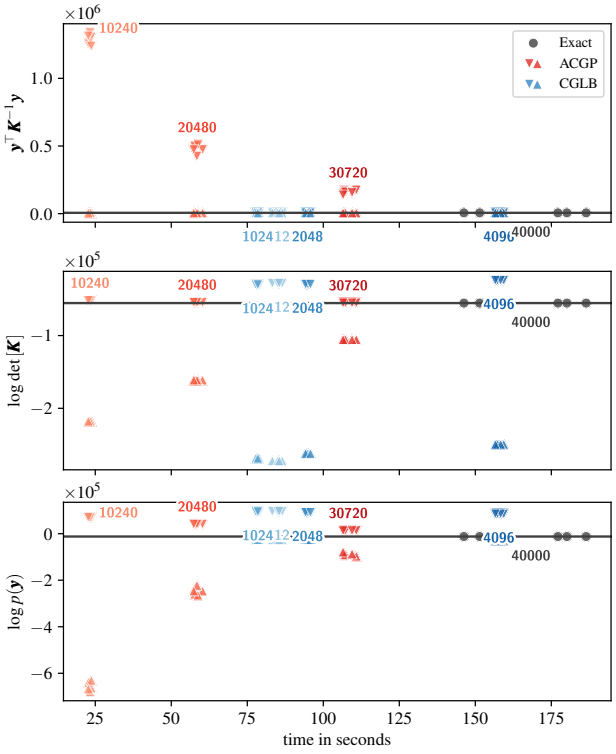
Figure 38: Upper and lower bounds for the `kin40k` dataset when using a squared exponential (SE) kernel.



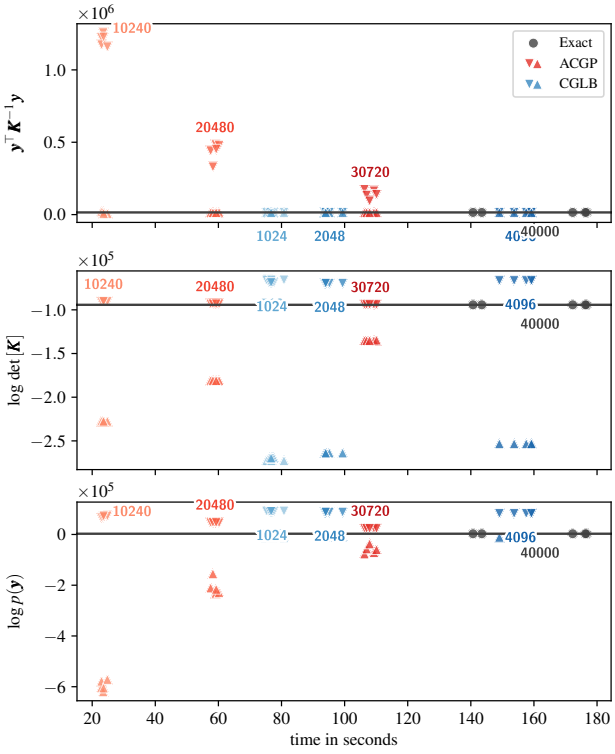
(a) OU kernel, $\log \ell = -1$.



(b) OU kernel, $\log \ell = 0$.



(c) OU kernel, $\log \ell = 1$.



(d) OU kernel, $\log \ell = 2$.

Figure 39: Upper and lower bounds for the `kin40k` dataset when using a Ornstein-Uhlenbeck (OU) kernel.

D NOTATION

We use a PYTHON-inspired index notation, abbreviating for example $[y_1, \dots, y_n]^\top$ as $\boldsymbol{y}_{:n}$—observe that the indexing starts at 1. Indexing binds before any other operation such that $\boldsymbol{K}^{-1}_{:s,:s}$ is the inverse of $\boldsymbol{K}_{:s,:s}$ and *not* all elements up to s of $\boldsymbol{K}^{-1}$. For $s \in \{1, \dots, N\}$ define $\mathcal{F}_s := \sigma(\boldsymbol{x}_1, y_1, \dots, \boldsymbol{x}_s, y_s)$ to be the σ-algebra generated by $\boldsymbol{x}_1, y_1, \dots, \boldsymbol{x}_s, y_s$. 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.

E PROOF SKETCH

In this section of the appendix, we provide additional intuition on the theorems and proofs for the theory behind ACGP.

E.1 The cumulative perspective

Using Bayes rule, we can write $\log p(\boldsymbol{y})$ equivalently as

$$\log p(\boldsymbol{y}) = -\frac{1}{2}\left(\log\det\left[\boldsymbol{K}_N\right] + \boldsymbol{y}^\top \boldsymbol{K}_N^{-1}\boldsymbol{y} + N\log(2\pi)\right) = \sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{y}_{:n-1}). \tag{16}$$

For each potential stopping point t we can decompose Equation (16) into a sum of three terms:

$$\log p(\boldsymbol{y}) = \underbrace{\sum_{n=1}^{s} \log p(y_n \mid \boldsymbol{y}_{:n-1})}_{A\text{: fully processed}} + \underbrace{\sum_{n=s+1}^{t} \log p(y_n \mid \boldsymbol{y}_{:n-1})}_{B\text{: partially processed}} + \underbrace{\sum_{n=t+1}^{N} \log p(y_n \mid \boldsymbol{y}_{:n-1})}_{C\text{: remaining}},$$

where $s < t$. We will use the partially processed points between s and t, to obtain unbiased upper and lower bounds on the expected value of $\log p(\boldsymbol{y}_{s+1:} \mid \boldsymbol{y}_{:s})$:

$$\mathbb{E}[\mathcal{L}_t \mid \boldsymbol{x}_1, y_1, \dots \boldsymbol{x}_s, y_s] \leq A + \mathbb{E}[B + C \mid \boldsymbol{x}_1, y_1, \dots \boldsymbol{x}_s, y_s] \leq \mathbb{E}[\mathcal{U}_t \mid \boldsymbol{x}_1, y_1, \dots \boldsymbol{x}_s, y_s]. \tag{17}$$

E.2 General bounds

The posterior of the nth observation conditioned on the previous is Gaussian with

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

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

$$\log\det\left[\boldsymbol{K}_N\right] = \sum_{n=1}^{N} \log\left(k_{n-1}(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2\right), \tag{18}$$

$$\boldsymbol{y}^\top \boldsymbol{K}_N^{-1}\boldsymbol{y} = \sum_{n=1}^{N} \frac{\left(y_n - m_{n-1}(\boldsymbol{x}_n)\right)^2}{k_{n-1}(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2}. \tag{19}$$

Our strategy is to find function families u (and l) which upper (and lower) bound the expectation

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

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,

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

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:

$$\mathbb{E}\left[\sum_{n=s+1}^{N} f_{n,t}^{x}(\boldsymbol{z}_n, \ldots, \boldsymbol{z}_1) \mid \sigma(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_s)\right] \tag{20}$$

$$= (N-s)\,\mathbb{E}\left[g(\boldsymbol{z}_{s+1}, \boldsymbol{z}_{s+1}, \boldsymbol{z}_1 \ldots, \boldsymbol{z}_n) \mid \sigma(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_s)\right]$$

$$+ (N-s)\,\frac{N-s+1}{2}\mathbb{E}\left[g(\boldsymbol{z}_{s+1}, \boldsymbol{z}_{s+2}, \boldsymbol{z}_1 \ldots, \boldsymbol{z}_n) \mid \sigma(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_s)\right]. \tag{21}$$

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

$$\approx \frac{N-t}{t-s}\sum_{n=s+1}^{t} g(\boldsymbol{z}_n, \boldsymbol{z}_n, \boldsymbol{z}_1 \ldots, \boldsymbol{z}_s) \tag{22}$$

$$+ \frac{2(N-t)}{t-s}\,\frac{N-s+1}{2}\sum_{i=1}^{\frac{t-s}{2}} g(\boldsymbol{z}_{s+2i}, \boldsymbol{z}_{s+2i-1}, \boldsymbol{z}_1 \ldots, \boldsymbol{z}_s).$$

E.3 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. Hence in this case, we simply ignore the interaction between the remaining variables. We set $g(\boldsymbol{x}_n, \boldsymbol{x}_i) := \delta_{ni} \log\left(k_s(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2\right)$ where δ_{ni} denotes Kronecker's δ.

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 $c = \sigma^2$, $a = k_s(\boldsymbol{x}_n, \boldsymbol{x}_n)$ and $b = k_s(\boldsymbol{x}_n, \boldsymbol{X}_{s+1:n-1})\left(k_s(\boldsymbol{X}_{s+1:n-1}, \boldsymbol{X}_{s+1:n-1}) + \sigma^2\right)^{-1} k_s(\boldsymbol{X}_{s+1:n-1}, \boldsymbol{x}_n)$. Underestimating the eigenvalues of $k_s(\boldsymbol{X}_{s+1:n-1}, \boldsymbol{X}_{s+1:n-1})$ by 0 we obtain a lower bound, where each quantity can be estimated. Formally, for any $s \leq t$,

$$\log\left(k_{n-1}(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2\right) \geq \left(\log\left(k_s(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2\right) - \sum_{i=s+1}^{n-1} \frac{k_s(\boldsymbol{x}_n, \boldsymbol{x}_i)^2}{\sigma^2\sigma^2}\right). \tag{23}$$

This bound can be worse than the deterministic lower bound $\log \sigma^2$. It depends on how large n is, how large the average correlation is and how small $\log \sigma^2$ is. Denote with μ the estimator for the left addend and with ρ the estimator for the second addend. We can determine the number of steps $n - s$ that this bound is better by solving for the maxima of a quadratic equation:

$$p\left(\mu - \frac{p-1}{2}\rho\right) \geq p \log \sigma^2 \tag{24}$$

The tipping point ψ is

$$\psi := \max\left(N, s + \left\lfloor \frac{\mu - \log \sigma^2}{\rho} + \frac{1}{2} \right\rfloor\right). \tag{25}$$

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

Observe that, the smaller $k_s(\boldsymbol{x}_j, \boldsymbol{x}_{j+1})^2$ 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 $k_s(\boldsymbol{x}_j, \boldsymbol{x}_{j+1})^2 = 0$ is the basic assumption of inducing input approximations (Quiñonero-Candela and Rasmussen 2005).

E.4 Bounds on the quadratic form

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

$$\frac{x}{c+a-b} \leq \frac{x(c+b)}{c(c+a)}, \tag{26}$$

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

$$\frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2} \leq_E \frac{(y_j - m_s(\boldsymbol{x}_j))^2}{\sigma^2 \left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right)} \left(\sigma^2 + \sum_{i=s+1}^{j-1} \frac{(k_s(\boldsymbol{x}_j, \boldsymbol{x}_i))^2}{\sigma^2}\right) \tag{27}$$

For a lower bound observe that

$$\boldsymbol{y}^\top \boldsymbol{K}^{-1} \boldsymbol{y} = \boldsymbol{y}_{:s}^\top \boldsymbol{K}_s^{-1} \boldsymbol{y}_{:s} + (\boldsymbol{y}_{s+1:N} - m_s(\boldsymbol{X}_{t+1:N}))^\top \boldsymbol{Q}_{s+1:N}^{-1} (\boldsymbol{y}_{s+1:N} - m_s(\boldsymbol{X}_{t+1:N})) \tag{28}$$

where $\boldsymbol{Q}_{s+1:j} := k_s(\boldsymbol{X}_{s+1:j}, \boldsymbol{X}_{s+1:j}) + \sigma^2 \boldsymbol{I}$ with $j \geq s+1$ for the posterior covariance matrix of $\boldsymbol{X}_{s+1:j}$ conditioned on $\boldsymbol{X}_{1:s}$. We use a trick we first encountered in Kim and Teh (2018): $\boldsymbol{y}^\top \boldsymbol{A}^{-1} \boldsymbol{y} \geq 2\boldsymbol{y}^\top \boldsymbol{b} - \boldsymbol{b}^\top \boldsymbol{A} \boldsymbol{b}$, for any $\boldsymbol{b}$. For brevity introduce $\boldsymbol{e} := \boldsymbol{y}_{s+1:N} - m_s(\boldsymbol{X}_{t+1:N})$. After applying the inequality with $\boldsymbol{b} := \mathrm{Diag}[\boldsymbol{Q}_{s+1:N}]^{-1} \boldsymbol{e}$, we obtain

$$2 \sum_{n=s+1}^{N} \frac{(y_n - m_s(\boldsymbol{x}_n))^2}{k_s(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2} - \sum_{n,n'=s+1}^{N} \frac{(y_n - m_s(\boldsymbol{x}_n))}{k_s(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2} [\boldsymbol{Q}_{s+1:N}]_{nn'} \frac{(y_{n'} - m_s(\boldsymbol{x}_{n'}))}{k_s(\boldsymbol{x}_{n'}, \boldsymbol{x}_{n'}) + \sigma^2} \tag{29}$$

which is now in the form of Equation (20).

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

E.5 Using the Bounds for Stopping the Cholesky

We will use the following stopping strategy: 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

$$\frac{\mathcal{U} - \mathcal{L}}{2 \min(|\mathcal{U}|, |\mathcal{L}|)} \leq r \text{ and} \tag{30}$$

$$\operatorname{sign} \mathcal{U} = \operatorname{sign} L, \tag{31}$$

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, and instead use the biased estimator $(N - \tau)\frac{1}{\tau} \log p(\boldsymbol{y}_{:\tau})$. Since stopping occurs when log-determinant and quadratic form evolve roughly linearly, the two estimators are not far off each other. The main reason for using the biased estimator is of a technical nature; it is easier and faster to implement a custom backward function which can handle the in-place operations of our Cholesky implementation.*

Remark 5. *To estimate the average correlation between elements of the kernel matrix, we use all elements of the off-diagonal instead of only every second. This has no effect on our main result, but it becomes important when developing PAC bounds.*

Remark 6. *The lower bound on the log-determinant, and the upper bound on the quadratic form switch their form at a step ψ (Theorems 10 and 14). Currently, to prove our results, this requires ψ to be $\mathcal{F}_s$-measurable, and for that reason we use estimators using inputs only up to index s, to define ψ. However, a PAC bound proof would allow to condition on the event that the estimators (plus some ϵ) overestimate their expected values with high probability. Under that condition, we could use the true expected value (which is $\mathcal{F}_s$-measurable) to define ψ. Hence, in our practical implementation we use estimators based on inputs with indices up to M to define ψ.*

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

$$\boldsymbol{L}_N = \begin{bmatrix} \boldsymbol{L}_s & \boldsymbol{0} \\ \boldsymbol{T} & \mathrm{chol}\left[k(\boldsymbol{X}_{s+1:,s+1:}) - \boldsymbol{T}\boldsymbol{T}^\top\right] \end{bmatrix} \tag{32}$$

One can verify that $\boldsymbol{L}_N$ is indeed the Cholesky of $\boldsymbol{K}_N$ by evaluating $\boldsymbol{L}_N \boldsymbol{L}_N^\top$. Observe that $k(\boldsymbol{X}_{s+1:,s+1:}) - \boldsymbol{T}\boldsymbol{T}^\top$ is the posterior covariance matrix of the $\boldsymbol{y}_{s+1:}$ conditioned on $\boldsymbol{y}_{:s}$. 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

$$\boldsymbol{\alpha}_N = \begin{bmatrix} \boldsymbol{\alpha}_s \\ \text{chol}\left[k(\boldsymbol{X}_{s+1:,s+1:}) - \boldsymbol{T}\boldsymbol{T}^\top\right]^{-1}\left(\boldsymbol{y}_{s+1:} - \boldsymbol{T}_s\boldsymbol{\alpha}_s\right) \end{bmatrix} \tag{33}$$

Now observe that $\boldsymbol{T}_s\boldsymbol{\alpha}_s = m_s(\boldsymbol{X}_{s+1:})$. 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 marked in gray.

procedure ACGP($k(\cdot,\cdot)$, $\mu(\cdot)$, σ^2, $\boldsymbol{X}$, $\boldsymbol{y}$, m, $N_{\max}$)
  $\boldsymbol{A} \leftarrow \boldsymbol{0}^{N_{\max}\times N_{\max}}, \boldsymbol{\alpha} \leftarrow \boldsymbol{0}^{N_{\max}}$ // allocate memory
  $\boldsymbol{A}_{1:m,1:m} \leftarrow k(\boldsymbol{X}_{1:m}) + \sigma^2$ // initialize kernel matrix
  $\boldsymbol{\alpha}_{1:m} \leftarrow \boldsymbol{y}_{1:m} - \mu(\boldsymbol{X}_{1:m})$ // evaluate mean function for the same datapoints
  $\boldsymbol{A}_{1:m,1:m} \leftarrow \text{chol}(\boldsymbol{A}_{1:m,1:m})$ // call to low-level Cholesky
  $\boldsymbol{\alpha}_{1:m} \leftarrow \boldsymbol{A}^{-1}_{1:m,1:m}\boldsymbol{\alpha}_{1:m}$ // second back-substitution step
  $i \leftarrow m+1$, $j \leftarrow \min(i+m, N)$
  while $i < N_{\max}$ **do**
    $\boldsymbol{A}_{i:j,1:i} \leftarrow k(\boldsymbol{X}_{i:j}, \boldsymbol{X}_{1:i})$ // evaluate required block-off-diagonal part of the kernel matrix
    $\boldsymbol{A}_{i:j,1:i} \leftarrow \boldsymbol{A}_{i:j,1:i}\boldsymbol{A}^{-\top}_{1:i,1:i}$ // solve triangular linear equation system
    $\boldsymbol{A}_{i:j,i:j} \leftarrow k(\boldsymbol{X}_{i:j}) + \sigma^2$ // evaluate required block-diagonal part of the kernel matrix
    $\boldsymbol{\alpha}_{i:j} \leftarrow \boldsymbol{y}_{i:j} - \mu(\boldsymbol{X}_{i:j})$ // evaluate mean function for the same datapoints
    $\boldsymbol{A}_{i:j,i:j} \leftarrow \boldsymbol{A}_{i:j,i:j} - \boldsymbol{A}_{i:j,1:i}\boldsymbol{A}^{\top}_{i:j,1:i}$ // down-date
    // now $\boldsymbol{A}_{i:j,i:j} = \boldsymbol{Q}_{s+1:j}$
    $\boldsymbol{\alpha}_{i:j} \leftarrow \boldsymbol{\alpha}_{i:j} - \boldsymbol{A}_{i:j,1:i}\boldsymbol{\alpha}_{1:i}$ // now $\boldsymbol{\alpha}_{i:j}$ contains $\boldsymbol{y}_{i:j} - m_i(\boldsymbol{X}_{i:j})$
    $\mathcal{L}, \mathcal{U} \leftarrow$**EvaluateBounds**$(i, j)$ // costs $\mathcal{O}(j-i)$
    if Equations (30) and (31) fulfilled **then**
      return estimator
    end if
    $\boldsymbol{A}_{i:j,i:j} \leftarrow \text{chol}(\boldsymbol{A}_{i:j,i:j})$ // finish computing Cholesky for data-points up to index j
    $\boldsymbol{\alpha}_{i:j} \leftarrow \boldsymbol{A}^{-1}_{i:j,i:j}\boldsymbol{\alpha}_{i:j}$ // finish solving linear equation system for index up to j
    $i \leftarrow i+m$, $j \leftarrow \min(i+m, N_{\max})$
  end while
  // now $\boldsymbol{A} = \boldsymbol{L}$ and $\boldsymbol{\alpha} = \boldsymbol{L}^{-1}(\boldsymbol{y} - \mu(\boldsymbol{X}))$
  return estimator
end procedure

F ASSUMPTIONS

Assumption 7. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(\boldsymbol{x}_j, y_j)_{j=1}^N$ be a sequence of independent and identically distributed random vectors with $\boldsymbol{x} : \Omega \to \mathbb{R}^D$ and $y : \Omega \to \mathbb{R}$.*

Assumption 8. *For all s, i, j, t with $s < i \leq j \leq N$ and functions $f(\boldsymbol{x}_j, \boldsymbol{x}_i; \boldsymbol{x}_1, \ldots \boldsymbol{x}_s) \geq 0$*

$$\mathbb{E}\left[f(\boldsymbol{x}_j, \boldsymbol{x}_i)\left(y_j - m_{j-1}(\boldsymbol{x}_j)\right)^2 \mid \mathcal{F}_s\right] \leq \mathbb{E}\left[f(\boldsymbol{x}_j, \boldsymbol{x}_i)\left(y_j - m_s(\boldsymbol{x}_j)\right)^2 \mid \mathcal{F}_s\right] \tag{34}$$

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

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

G MAIN THEOREM

This section restates Theorem 2 and connects the different proofs in the sections to follow.

Algorithm 2 bound algorithm as used in our experiments. The algorithm deviates slightly from our theory. We use Equation (54) for the upper bound in the quadratic form, and we use all off-diagonal entries (instead of only every second).

procedure EVALUATEBOUNDS(s, t)
  $D \leftarrow \sum_{j=1}^{s} \log \boldsymbol{A}_{jj}$ // in practice we reuse the sum from the last iteration
  $Q \leftarrow \sum_{j=1}^{s} \boldsymbol{\alpha}_j^2$
  $\mu \leftarrow \frac{1}{t-s} \sum_{j=s+1}^{t} \log \boldsymbol{A}_{jj}$ // average variance of the new points conditioned on all points processed until s
  $\mathcal{U}_D \leftarrow D + (N-s)\mu$
  $\rho \leftarrow \frac{1}{t-s-1} \sum_{j=s+1}^{t-1} \frac{\boldsymbol{A}_{j,j+1}^2}{\sigma^2 \sigma^2}$ // average square correlation (deviating from theory!)
  $\psi \leftarrow \min(N, s + \lfloor \frac{\mu - \log \sigma^2}{\rho} + \frac{1}{2} \rfloor$ // number of steps the probabilistic bound is better than the deterministic
  $\mathcal{L}_D \leftarrow D + (\psi - s)\left(\mu - \frac{\psi - s - 1}{2}\rho\right) + (N - \psi)\log \sigma^2$
  $\mu \leftarrow \frac{1}{t-s} \sum_{j=s+1}^{t} \frac{\boldsymbol{\alpha}_j^2}{\boldsymbol{A}_{j,j}}$ // average error calibration
  $\rho \leftarrow \max\left(0, \frac{1}{t-s-1} \sum_{j=s+1}^{t-1} \frac{\boldsymbol{\alpha}_j \boldsymbol{\alpha}_{j+1} \boldsymbol{A}_{j,j+1}}{\boldsymbol{A}_{j,j} \boldsymbol{A}_{j+1,j+1}}\right)$ // calibrated error correlation
  $\mathcal{L}_Q \leftarrow Q + \max(0, (N-s)(2\mu - \rho(N-s-1)))$
  $\rho \leftarrow \frac{1}{t-s-1} \sum_{j=s+1}^{t-1} \frac{\boldsymbol{\alpha}_j^2 \boldsymbol{A}_{j,j+1}^2}{\boldsymbol{A}_{j,j} \sigma^2 \sigma^2}$ // square error correlation
  $\hat{\mu} \leftarrow \frac{1}{t-s} \sum_{j=s+1}^{t} \frac{\boldsymbol{\alpha}_j^2}{\sigma^2}$ // worst-case estimate for the quadratic
  $\psi \leftarrow \min(N, s + \lfloor \frac{\mu - \hat{\mu}}{\rho} + \frac{1}{2} \rfloor)$ // number of steps the bound is better than the worst-case estimate
  $\mathcal{U}_Q \leftarrow Q + (\psi - s)(\mu - \frac{\psi - s - 1}{2}\rho) + (N - \psi)\hat{\mu}$
  return $\mathcal{L}_D + \mathcal{L}_Q, \mathcal{U}_D + \mathcal{U}_Q$
end procedure

Theorem 9. *Assume that Assumption 7 and Assumption 8 hold. For any even $m \in \{2, 4, \ldots, N-2\}$ and any $s \in \{1, \ldots, N-m\}$, the bounds defined in Equation* (6) *and Theorems 10, 14 and 19 hold in expectation:*

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

Proof. Follows from Theorems 10, 14 and 19 and Lemma 23. □

H PROOF FOR THE LOWER BOUND ON THE DETERMINANT

Theorem 10. *Assume that Assumption 7 holds, and that $m \in \{2, 4, \ldots, N\}$ is an even number. Set $t := s + m$, then, for all $s \in \{1, \ldots, N-m\}$*

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

$$\mathcal{L}_D := \log\det[\boldsymbol{K}_{:s,:s}] + (\psi_s - s)\left(\log \mu_t - \frac{\psi_s - s - 1}{2}\rho_t\right) + (N - \psi_s)\log\sigma^2 \tag{35}$$

// the lower bound

$$\log \mu_t := \frac{1}{m} \sum_{j=s+1}^{t} \log\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right) \tag{36}$$

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

$$\rho_t := \frac{2}{m} \sum_{j=\frac{s+1}{2}}^{\frac{t-1}{2}} \frac{k_s(\boldsymbol{x}_{2j+1}, \boldsymbol{x}_{2j})^2}{\sigma^2 \sigma^2} \tag{37}$$

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

$$\psi_s := s + \max p \text{ where } p \in \mathbb{N} \text{ is such that} \tag{38}$$

$$p\left(\log \mu_{t-m} - \frac{p-1}{2}\rho_{t-m}\right) \geq p \log \sigma^2 \tag{39}$$

$$
\begin{aligned}
&/\!/ \text{ number of steps that we suspect the decrease in variance to be controllable} \\
&= \min\left(N, s + \left\lfloor \frac{\log \mu_{t-m} - \log \sigma^2}{\rho_{t-m}} + \frac{1}{2} \right\rfloor\right) \qquad (40)
\end{aligned}
$$

where, if $s - m < 1$*, we set* $\log \mu_{t-m} := \log \sigma^2$ *and* $\rho_{t-m} := 1$.

Remark 11. *For our proof we require* ψ_s *to be* $\mathcal{F}_s$*-measurable. We conjecture that for a PAC bound proof this requirement can be relaxed. Therefore, in our implementation, we use* $\log \mu_{t-m} := \log \mu_t$ *and* $\rho_{t-m} := \rho_t$.

Proof.

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{L}_D \middle| \mathcal{F}_s\right] - \mathbb{E}\left[\log\det\left[\boldsymbol{K}\right] \middle| \mathcal{F}_s\right] = \mathbb{E}\left[\mathcal{L}_D - \log\det\left[\boldsymbol{K}\right] \middle| \mathcal{F}_s\right] \\
&= \mathbb{E}\left[(\psi_s - s)\left(\log \mu_t - \frac{\psi_s - s - 1}{2}\rho_t\right) + (N - \psi_s)\log\sigma^2 - \sum_{j=s+1}^{N} \log\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right) \middle| \mathcal{F}_s\right] \\
&/\!/ \textit{ using the definition of } \mathcal{L}_t \textit{ and slightly simplifying using Lemma 22} \\
&\leq \mathbb{E}\left[(\psi_s - s)\left(\log \mu_t - \frac{\psi_s - s - 1}{2}\rho_t\right) - \sum_{j=s+1}^{\psi_s} \log\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right) \middle| \mathcal{F}_s\right] \\
&/\!/ \textit{ using that } \log\sigma^2 \leq \log\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right) \textit{ for all } j \\
&= (\psi_s - s)\left(\mathbb{E}\left[\log\left(\sigma^2 + k_s(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1})\right) \middle| \mathcal{F}_s\right]\right) - \frac{\psi_s - s - 1}{2}\mathbb{E}\left[\frac{k_s(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2})^2}{\sigma^2\sigma^2} \middle| \mathcal{F}_s\right]\Bigg) \\
&\quad - \mathbb{E}\left[\sum_{j=s+1}^{\psi_s} \log\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right) \middle| \mathcal{F}_s\right] \\
&/\!/ \textit{ using Assumption 7} \\
&\leq 0 \\
&/\!/ \textit{ Lemma 13}
\end{aligned}
$$

□

Lemma 12. *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 13. *For all* $n \geq t \geq s \in \mathbb{N}$*:*

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

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

$$\mathbb{E}\left[\sum_{j=t+1}^{n} \log k_{j-1}(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2 \middle| \mathcal{F}_s\right] \tag{41}$$

$$= \mathbb{E}\left[\sum_{j=t+1}^{n} \log\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) - k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(k_s(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{X}}_j) + \sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)\right) \middle| \mathcal{F}_s\right] \tag{42}$$

// Lemma 25

$$\geq \mathbb{E}\left[\sum_{j=t+1}^{n} \left(\log\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right) - \frac{1}{\sigma^2} k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(k_s(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{X}}_j) + \sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)\right) \middle| \mathcal{F}_s\right] \tag{43}$$

// Lemma 12

$$\geq \mathbb{E}\left[\sum_{j=t+1}^{n} \left(\log\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right) - \frac{1}{\sigma^2} k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(\sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)\right) \middle| \mathcal{F}_s\right] \tag{44}$$

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

$$\geq \mathbb{E}\left[\sum_{j=t+1}^{n} \left(\log\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right) - \frac{1}{\sigma^2} \sum_{i=t+1}^{j-1} \frac{k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)^2}{\sigma^2}\right) \middle| \mathcal{F}_s\right] \tag{45}$$

// writing the vector multiplication as sum

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

// using Assumption 7 and then applying Lemma 26

□

I PROOF FOR THE UPPER BOUND ON THE QUADRATIC FORM

Theorem 14. *Assume that Assumption 7 and Assumption 8 hold. Let $m \in \mathbb{N}$ be even, then for all $s \in \{1, \ldots, N-m\}$*

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

where

$$\mathcal{U}_Q := \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1} \boldsymbol{y}_{:s} + (\psi_s - s)\left(\mu_t + \frac{\psi_s - s - 1}{2}\rho_t\right) + (N - \psi_s)\overline{\mu}_t \tag{46}$$

// the upper bound

$$\mu_t := \frac{1}{t-s} \sum_{j=s+1}^{t} \frac{(y_j - m_s(\boldsymbol{x}_j))^2}{k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2} \tag{47}$$

$$\rho_t := \frac{2}{t-s} \sum_{j=\frac{s+2}{2}}^{\frac{t}{2}} \frac{(y_{2j} - m_s(\boldsymbol{x}_{2j}))^2 k_s(\boldsymbol{x}_{2j}, \boldsymbol{x}_{2j-1})^2}{(k_s(\boldsymbol{x}_{2j}, \boldsymbol{x}_{2j}) + \sigma^2)\,\sigma^2\sigma^2} \tag{48}$$

$$\overline{\mu}_t := \frac{1}{t-s} \sum_{j=s+1}^{t} \frac{(y_j - m_s(\boldsymbol{x}_j))^2}{\sigma^2} \tag{49}$$

$$\psi_s := \min\left(N, s + \left\lfloor \frac{\overline{\mu}_{t-m} - \mu_{t-m}}{\rho_{t-m}} + \frac{1}{2} \right\rfloor\right) . \tag{50}$$

where, if $s - m < 1$, we set $\overline{\mu}_{t-m} = \mu_{t-m} := 0$ and $\rho_{t-m} := 1$.

Proof.

$$
\begin{aligned}
&\mathbb{E}\left[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \mid \mathcal{F}_s\right] - \mathbb{E}\left[\mathcal{U}_Q \mid \mathcal{F}_s\right] \\
&= \mathbb{E}\left[\sum_{j=s+1}^{N} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} - (\psi_s - s)\left(\mu_t + \frac{\psi_s - s - 1}{2}\rho_t\right) + (N-\psi_s)\overline{\mu}_t \mid \mathcal{F}_s\right] \\
&\text{// using the definition of } \mathcal{U}_Q \text{ and slightly simplifying with Lemma 24} \\
&= \mathbb{E}\left[\sum_{j=s+1}^{N} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \mid \mathcal{F}_s\right] - (\psi_s - s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2} \mid \mathcal{F}_s\right] \\
&\quad - (\psi_s - s)\frac{\psi_s - s - 1}{2}\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2 k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+2})^2}{(k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2)\,\sigma^2} \mid \mathcal{F}_s\right] \\
&\quad - (N-\psi_s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{\sigma^2} \mid \mathcal{F}_s\right] \\
&\text{// using Assumption 7} \\
&\leq 0 \\
&\text{// Lemma 17 with } n = \psi_s \text{ and } t = s, \text{ and Lemma 15 with } n = N
\end{aligned}
$$

□

Lemma 15. *For all $\psi, n, s \in \mathbb{N}$ with $s \leq \psi \leq n$:*

$$
\mathbb{E}\left[\sum_{j=\psi+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \mid \mathcal{F}_s\right] \leq (n-\psi)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{\sigma^2} \mid \mathcal{F}_s\right]
$$

Proof.

$$
\begin{aligned}
\mathbb{E}\left[\sum_{j=\psi+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \mid \mathcal{F}_s\right] &\leq \mathbb{E}\left[\sum_{j=\psi+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{\sigma^2} \mid \mathcal{F}_s\right] && (51) \\
&\text{// the posterior variance cannot fall below } \sigma^2 \\
&\leq \mathbb{E}\left[\sum_{j=\psi+1}^{n} \frac{(y_j - m_s(\boldsymbol{x}_j))^2}{\sigma^2} \mid \mathcal{F}_s\right] && (52) \\
&\text{// by Assumption 8} \\
&= (n-\psi)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{\sigma^2} \mid \mathcal{F}_s\right] && (53) \\
&\text{// using Assumption 7}
\end{aligned}
$$

□

Lemma 16. *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.

$$
\begin{aligned}
\frac{x}{c+a-b} &= \frac{x}{c}\left(\frac{c}{c+a-b}\right) \\
&= \frac{x}{c}\left(1 - \frac{a-b}{c+a-b}\right)
\end{aligned}
$$

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

□

Lemma 17. *For all $s, t, n \in \mathbb{N}$ with $n \geq t \geq s$:*

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

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

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{j=t+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2} \mid \mathcal{F}_s\right] \\
&= \mathbb{E}\left[\sum_{j=t+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) - k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(k_s(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{X}}_j) + \sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)} \mid \mathcal{F}_s\right] \\
&/\!/ \textit{ Lemma 25} \\
&\leq \mathbb{E}\left[\sum_{j=t+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2\left(\sigma^2 + k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(k_s(\overline{\boldsymbol{X}}_j, \overline{\boldsymbol{X}}_j) + \sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)\right)}{\sigma^2\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right)} \mid \mathcal{F}_s\right] \\
&/\!/ \textit{ Lemma 16} \\
&\leq \mathbb{E}\left[\sum_{j=t+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2\left(\sigma^2 + k_s(\boldsymbol{x}_j, \overline{\boldsymbol{X}}_j)\left(\sigma^2\right)^{-1} k_s(\overline{\boldsymbol{X}}_j, \boldsymbol{x}_j)\right)}{\sigma^2\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right)} \mid \mathcal{F}_s\right] \\
&/\!/ \textit{ underestimating the eigenvalues of } k_s(\overline{\boldsymbol{X}}, \overline{\boldsymbol{X}}) \textit{ by } 0 \\
&= \mathbb{E}\left[\sum_{j=t+1}^{n} \frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2\left(\sigma^2 + \sum_{i=s+1}^{j-1} \frac{k_s(\boldsymbol{x}_j, \boldsymbol{x}_i)^2}{\sigma^2}\right)}{\sigma^2\left(\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)\right)} \mid \mathcal{F}_s\right] \\
&/\!/ \textit{ writing the vector-product explicitly as a sum} \\
&= \sum_{j=t+1}^{n}\left(\mathbb{E}\left[\frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)} \mid \mathcal{F}_s\right] + \sum_{i=s+1}^{j-1} \mathbb{E}\left[\frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2 k_s(\boldsymbol{x}_j, \boldsymbol{x}_i)^2}{\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right)\sigma^2\sigma^2} \mid \mathcal{F}_s\right]\right) \\
&/\!/ \textit{ linearity of expectation} \\
&= \sum_{j=t+1}^{n}\left(\mathbb{E}\left[\frac{(y_j - m_s(\boldsymbol{x}_j))^2}{\sigma^2 + k_s(\boldsymbol{x}_j, \boldsymbol{x}_j)} \mid \mathcal{F}_s\right] + \sum_{i=s+1}^{j-1} \mathbb{E}\left[\frac{(y_j - m_s(\boldsymbol{x}_j))^2 k_s(\boldsymbol{x}_j, \boldsymbol{x}_i)^2}{\left(k_s(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2\right)\sigma^2\sigma^2} \mid \mathcal{F}_s\right]\right) \\
&/\!/ \textit{ by assumption Equation } (34) \\
&= \sum_{j=t+1}^{n}\left(\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{\sigma^2 + k_s(\boldsymbol{x}_{s+1}, \boldsymbol{x}_{s+1})} \mid \mathcal{F}_s\right] + \sum_{i=s+1}^{j-1} \mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2 k_s(\boldsymbol{x}_{s+1}, \boldsymbol{x}_{s+2})^2}{\left(k_s(\boldsymbol{x}_{s+1}, \boldsymbol{x}_{s+1}) + \sigma^2\right)\sigma^2\sigma^2} \mid \mathcal{F}_s\right]\right)
\end{aligned}
$$

// using Assumption 7

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

// by Lemma 26

□

Remark 18. *Similar to the proof of Theorem 10, 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 $p \in [0, N-s]$ such that*

$$\mu_s + \frac{p-1}{2}\rho_s \le \frac{1}{m}\sum_{j=s-m+1}^{s}\frac{(y_j - m_s(\boldsymbol{x}_{\boldsymbol{x}_j}))^2}{\sigma^2},$$

and replace the upper bound by

$$\mathcal{U}_t \coloneqq \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + p\left(\mu_t + \frac{p-1}{2}\rho_t\right) + \frac{N-p-s}{t-s}\sum_{j=s+1}^{t}\frac{(y_j - m_s(\boldsymbol{x}_{\boldsymbol{x}_j}))^2}{\sigma^2}. \tag{54}$$

J PROOF FOR THE LOWER BOUND ON THE QUADRATIC FORM

Theorem 19. *Assume that Assumption 7 holds. Let $m \in \{2,\dots,N-2\}$ be an even number less than N. For $s \in \{1,\dots,N-m\}$,*

$$\mathbb{E}\left[\mathcal{L}_Q \middle| \mathcal{F}_s\right] \le \mathbb{E}\left[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \middle| \mathcal{F}_s\right]$$

where

$$\mathcal{L}_Q \coloneqq \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + (N-s)\left(\mu_t - (N-s-1)\max(0,\rho_t)\right) \tag{55}$$

$$\mu_t \coloneqq \frac{1}{t-s}\sum_{j=s+1}^{t}\frac{(y_j - m_s(\boldsymbol{x}_j))^2}{k_s(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \tag{56}$$

$$\rho_t \coloneqq \frac{2}{t-s}\sum_{j=\frac{s+2}{2}}^{\frac{t}{2}}\frac{(y_{2j} - m_s(\boldsymbol{x}_{2j}))(y_{2j-1} - m_s(\boldsymbol{x}_{2j-1}))k_s(\boldsymbol{x}_{2j},\boldsymbol{x}_{2j-1})}{(k_s(\boldsymbol{x}_{2j},\boldsymbol{x}_{2j})+\sigma^2)(k_s(\boldsymbol{x}_{2j-1},\boldsymbol{x}_{2j-1})+\sigma^2)} \tag{57}$$

Proof.

$$\mathbb{E}\left[\mathcal{L}_Q \mid \mathcal{F}_s\right] - \mathbb{E}\left[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \mid \mathcal{F}_s\right]$$
$$= \mathbb{E}\left[(N-s)\left(\mu_t - (N-s-1)\max(0,\rho_t)\right) - \sum_{j=s+1}^{N}\frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \mid \mathcal{F}_s\right]$$

// using the definition of $\mathcal{L}_Q$ and slightly simplifying

$$\le \mathbb{E}\left[(N-s)\left(\mu_t - (N-s-1)\rho_t\right) - \sum_{j=s+1}^{N}\frac{(y_j - m_{j-1}(\boldsymbol{x}_j))^2}{k_{j-1}(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2} \mid \mathcal{F}_s\right]$$

// allowing ρ to be negative increases the lower bound

$$= (N-s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2}\middle| \mathcal{F}_s\right]$$
$$- (N-s)(N-s-1)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))(y_{s+2} - m_s(\boldsymbol{x}_{s+2}))k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+2})}{(k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2)(k_s(\boldsymbol{x}_{s+2},\boldsymbol{x}_{s+2})+\sigma^2)}\middle| \mathcal{F}_s\right]$$

// using Assumption 7

$$\le 0$$

// using Lemma 20

□

Lemma 20. *For all $\mathcal{F}_s$-measurable $\alpha \in \mathbb{R}$:*

$$\mathbb{E}\left[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \mid \mathcal{F}_s\right] \geq \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + \alpha(2-\alpha)(N-s)\mathbb{E}\left[\frac{(y_{s+1}-m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2}\middle|\mathcal{F}_s\right] - \alpha^2(N-s)(N-s-1)\mathbb{E}\left[\frac{(y_{s+1}-m_s(\boldsymbol{x}_{s+1}))(y_{s+2}-m_s(\boldsymbol{x}_{s+2}))k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+2})}{(k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2)(k_s(\boldsymbol{x}_{s+2},\boldsymbol{x}_{s+2})+\sigma^2)}\middle|\mathcal{F}_s\right] \quad (58)$$

Proof. Using Lemma 24, we can write the quadratic form as a sum of two quadratic forms:

$$\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + (\boldsymbol{y}_{s+1:} - m_s(\boldsymbol{X}_{s+1:}))^\top \left(k_s(\boldsymbol{X}_{s+1:},\boldsymbol{X}_{s+1:}) + \sigma^2\boldsymbol{I}\right)^{-1}(\boldsymbol{y}_{s+1:} - m_s(\boldsymbol{X}_{s+1:})). \quad (59)$$

For the right-hand addend, we use a trick we first encountered in Kim and Teh (2018): $\boldsymbol{a}^\top\boldsymbol{A}^{-1}\boldsymbol{a} \geq 2\boldsymbol{a}^\top\boldsymbol{b} - \boldsymbol{b}^\top\boldsymbol{A}\boldsymbol{b}$, for any $\boldsymbol{b}$ given $\boldsymbol{A}$ is symmetric and positive definite. Define $\boldsymbol{e} := (\boldsymbol{y}_{s+1:} - m_s(\boldsymbol{X}_{s+1:}))$, $\boldsymbol{D} := \mathrm{Diag}[\left(k_s(\boldsymbol{X}_{s+1:},\boldsymbol{X}_{s+1:}) + \sigma^2\boldsymbol{I}\right)]$ and choose $\boldsymbol{b} := \alpha\boldsymbol{D}^{-1}\boldsymbol{e}$.

$$\mathbb{E}\left[\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \mid \mathcal{F}_s\right] \quad (60)$$

$$= \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + \mathbb{E}\left[(\boldsymbol{y}_{s+1:} - m_s(\boldsymbol{X}_{s+1:}))^\top \left(k_s(\boldsymbol{X}_{s+1:},\boldsymbol{X}_{s+1:}) + \sigma^2\boldsymbol{I}\right)^{-1}(\boldsymbol{y}_{s+1:} - m_s(\boldsymbol{X}_{s+1:})) \mid \mathcal{F}_s\right] \quad (61)$$

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

$$\geq \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + 2\alpha\mathbb{E}\left[\boldsymbol{e}^\top\boldsymbol{D}^{-1}\boldsymbol{e} \mid \mathcal{F}_s\right] - \alpha^2\mathbb{E}\left[\boldsymbol{e}^\top\boldsymbol{D}^{-1}\left(k_s(\boldsymbol{X}_{s+1:},\boldsymbol{X}_{s+1:}) + \sigma^2\boldsymbol{I}\right)\boldsymbol{D}^{-1}\boldsymbol{e} \mid \mathcal{F}_s\right] \quad (62)$$

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

$$= \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + 2\alpha\sum_{j=s+1}^{N}\mathbb{E}\left[\frac{(y_j - m_s(\boldsymbol{x}_j))^2}{k_s(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2}\middle|\mathcal{F}_s\right] - \alpha^2\sum_{j=s+1}^{N}\sum_{i=s+1}^{N}\mathbb{E}\left[\frac{(y_j - m_s(\boldsymbol{x}_j))(y_i - m_s(\boldsymbol{x}_i))k_s(\boldsymbol{x}_j,\boldsymbol{x}_i) + \delta_{ij}\sigma^2}{(k_s(\boldsymbol{x}_j,\boldsymbol{x}_j)+\sigma^2)(k_s(\boldsymbol{x}_i,\boldsymbol{x}_i)+\sigma^2)}\middle|\mathcal{F}_s\right] \quad (63)$$

// writing the vector products as sums

$$= \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + 2\alpha(N-s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2}\middle|\mathcal{F}_s\right] - \alpha^2(N-s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2}\middle|\mathcal{F}_s\right] - \alpha^2\left((N-s)^2 - (N-s)\right)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))(y_{s+2} - m_s(\boldsymbol{x}_{s+2}))k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+2})}{(k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2)(k_s(\boldsymbol{x}_{s+2},\boldsymbol{x}_{s+2})+\sigma^2)}\middle|\mathcal{F}_s\right] \quad (64)$$

// using Assumption 7, grouping variance and covariance terms separately

$$= \boldsymbol{y}_{:s}^\top \boldsymbol{K}_{:s,:s}^{-1}\boldsymbol{y}_{:s} + \alpha(2-\alpha)(N-s)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))^2}{k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2}\middle|\mathcal{F}_s\right] \quad (65)$$

$$- \alpha^2(N-s)(N-s-1)\mathbb{E}\left[\frac{(y_{s+1} - m_s(\boldsymbol{x}_{s+1}))(y_{s+2} - m_s(\boldsymbol{x}_{s+2}))k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+2})}{(k_s(\boldsymbol{x}_{s+1},\boldsymbol{x}_{s+1})+\sigma^2)(k_s(\boldsymbol{x}_{s+2},\boldsymbol{x}_{s+2})+\sigma^2)}\middle|\mathcal{F}_s\right] \quad (66)$$

// simplifying

□

K UTILITY PROOFS

Lemma 21 (Bounding the relative error (Lemma 15 in [ANONYMIZED FOR PEER REVIEW])). *Let $D, \hat{D} \in [\mathcal{L},\mathcal{U}]$, and assume $\mathrm{sign}(\mathcal{L}) = \mathrm{sign}(\mathcal{U}) \neq 0$. 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 $\operatorname{sign}(\mathcal{L}) = \operatorname{sign}(\mathcal{U})$ 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 22. *The* log*-determinant of a kernel matrix can be written as a sum of conditional variances.*

$$\log\det\left[\boldsymbol{K}\right] = \sum_{j=1}^{N} \log(k_{j-1}(\boldsymbol{x}_j, \boldsymbol{x}_j) + \sigma^2) \tag{67}$$

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

$$\begin{aligned}\log\det\left[\boldsymbol{K}\right] &= \log\det\left[\boldsymbol{L}\boldsymbol{L}^\top\right] && (68)\\ &\quad /\!/ \textit{ using } \boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top \\ &= \log\left(\det\left[\boldsymbol{L}\right]\det\left[\boldsymbol{L}^\top\right]\right) && (69)\\ &\quad /\!/ \textit{ for square matrices } \boldsymbol{B}, \boldsymbol{C}\textit{: } \det\left[\boldsymbol{B}\boldsymbol{C}\right] = \det\left[\boldsymbol{B}\right]\det\left[\boldsymbol{C}\right] \\ &= \log\left(\prod_{j=1}^{N} \boldsymbol{L}_{jj}^2\right) && (70)\\ &\quad /\!/ \textit{ for triangular matrices the determinant is the product of the diagonal elements} \\ &= \sum_{j=1}^{N} 2\log\boldsymbol{L}_{jj} && (71)\\ &\quad /\!/ \textit{ property of } \log\end{aligned}$$

With Lemma 27 the result follows. □

Lemma 23 (The f_js are decreasing in expectation (Lemma 7 in [ANONYMIZED FOR PEER REVIEW]))**.** *Assume $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{X}$ are independent and identically distributed. Denote with $\mathbb{P}$ the law of the $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and with $\boldsymbol{L}$ the Cholesky decomposition of $\boldsymbol{K}$. Define the probability space $(\mathbb{X}, \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N), \mathbb{P})$ and the canonical filtration $\mathcal{F}_j := \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j)$ for $j = 1, \ldots, N$. Then the v_j decrease in conditional expectation, that is,*

$$\mathbb{E}[\log v_{j+1} \mid \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j)] \leq \mathbb{E}[\log v_j \mid \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1})]\,,$$

where

$$\begin{aligned}\boldsymbol{k}_n(\boldsymbol{x}) &:= \left[k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n)\right]^\top \in \mathbb{R}^n\,, && (72)\\ \boldsymbol{k}_{n+1} &:= \boldsymbol{k}_n(\boldsymbol{x}_{n+1}) \in \mathbb{R}^n \textit{ and} && (73)\\ v_n &:= k(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2 - \boldsymbol{k}_n^\top(\boldsymbol{K}_{n-1} + \sigma^2\boldsymbol{I}_{n-1})^{-1}\boldsymbol{k}_n\,. && (74)\end{aligned}$$

Proof. Denote with $\mathbb{Q}_j(\,\mathrm{d}\boldsymbol{x}) := \mathbb{P}(\,\mathrm{d}\boldsymbol{x} \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_j)$, the regular conditional probability. Define the shorthand $q_j(\boldsymbol{x}) := \boldsymbol{k}_j(\boldsymbol{x})^\top (\boldsymbol{K}_j + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_j(\boldsymbol{x})$. We will show later in the proof, in Eq. (93), that $q_j(\boldsymbol{x}) = q_{j-1}(\boldsymbol{x}) + r_j(\boldsymbol{x})$ where $r_j(\boldsymbol{x}) \geq 0$. Taking Eq. (93) as granted for now, we can show the claim as follows.

$$
\begin{aligned}
\mathbb{E}[\log v_{j+1} \mid \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j)] &= \mathbb{E}[\log \boldsymbol{L}^2_{j+1,j+1} \mid \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j)] && (75)\\
&\quad /\!/ \textit{ definition of } f_j \\
&= \int \log\left(k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - \boldsymbol{k}_j(\boldsymbol{x})^\top (\boldsymbol{K}_j + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_j(\boldsymbol{x})\right)\, \mathbb{Q}_j(\,\mathrm{d}\boldsymbol{x}) && (76)\\
&\quad /\!/ \textit{ property of conditional expectation} \\
&= \int \log\left(k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - q_j(\boldsymbol{x})\right)\, \mathbb{Q}_j(\,\mathrm{d}\boldsymbol{x}) && (77)\\
&\quad /\!/ \textit{ definition of } q_j(\boldsymbol{x}) \\
&= \int \log\left(k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - q_{j-1}(\boldsymbol{x}) - r_j(\boldsymbol{x})\right)\, \mathbb{Q}_j(\,\mathrm{d}\boldsymbol{x}) && (78)\\
&\quad /\!/ \textit{ using Eq. } (92) \\
&\leq \int \log\left(k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - q_{j-1}(\boldsymbol{x})\right)\, \mathbb{Q}_j(\,\mathrm{d}\boldsymbol{x}) && (79)\\
&\quad /\!/ \textit{ using Eq. } (93) \textit{ and monotonicity of the logarithm} \\
&= \int \log\left(k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - q_{j-1}(\boldsymbol{x})\right)\, \mathbb{Q}_{j-1}(\,\mathrm{d}\boldsymbol{x}) && (80)\\
&\quad /\!/ \textit{ with Fubini's theorem} \\
&= \mathbb{E}[\log v_j \mid \sigma(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1})] && (81)\\
&\quad /\!/ \textit{ property of conditional expectation}
\end{aligned}
$$

It remains to show $q_j(\boldsymbol{x}) = q_{j-1}(\boldsymbol{x}) + r_j(\boldsymbol{x})$ where $r_j(\boldsymbol{x}) \geq 0$. For readability, we define $\boldsymbol{v}_{\boldsymbol{x}} := (\boldsymbol{K}_{j-1} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{j-1}(\boldsymbol{x})$ and $c := v_j^{-1}$. First note, that using block-matrix inversion we can write

$$
(\boldsymbol{K}_j + \sigma^2 \boldsymbol{I}_j)^{-1} = \begin{bmatrix} (\boldsymbol{K}_{j-1} + \sigma^2 \boldsymbol{I}_{j-1})^{-1} + \boldsymbol{v}_{\boldsymbol{x}_j} c \boldsymbol{v}_{\boldsymbol{x}_j}^\top & -\boldsymbol{v}_{\boldsymbol{x}_j} c \\ -\boldsymbol{v}_{\boldsymbol{x}_j}^\top c & c \end{bmatrix}. \tag{82}
$$

Using above observation, we can transform $q_j(\boldsymbol{x})$.

$$
\begin{aligned}
q_j(\boldsymbol{x}) &= \begin{bmatrix} \boldsymbol{k}_{j-1}(\boldsymbol{x})^\top & k(\boldsymbol{x}_j, \boldsymbol{x}) \end{bmatrix} && (83)\\
&\quad \cdot \begin{bmatrix} (\boldsymbol{K}_{j-1} + \sigma^2 \boldsymbol{I})^{-1} + \boldsymbol{v}_{\boldsymbol{x}_j} c \boldsymbol{v}_{\boldsymbol{x}_j}^\top & -\boldsymbol{v}_{\boldsymbol{x}_j} c \\ -\boldsymbol{v}_{\boldsymbol{x}_j}^\top c & c \end{bmatrix} && (84)\\
&\quad \cdot \begin{bmatrix} \boldsymbol{k}_{j-1}(\boldsymbol{x}) \\ k(\boldsymbol{x}_j, \boldsymbol{x}) \end{bmatrix} && (85)\\
&\quad /\!/ \textit{ definition of } q_j(\boldsymbol{x}) \textit{ and using above observation} \\
&= \begin{bmatrix} \boldsymbol{k}_{j-1}(\boldsymbol{x})^\top & k(\boldsymbol{x}, \boldsymbol{x}_j) \end{bmatrix} && (86)\\
&\quad \cdot \begin{bmatrix} \boldsymbol{v}_{\boldsymbol{x}} + \boldsymbol{v}_{\boldsymbol{x}_j} c \boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}) - \boldsymbol{v}_{\boldsymbol{x}_j} c k(\boldsymbol{x}, \boldsymbol{x}_j) \\ -\boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}) c + c k(\boldsymbol{x}, \boldsymbol{x}_j) \end{bmatrix} && (87)\\
&\quad /\!/ \textit{ evaluating the RHS matrix-vector multiplication} \\
&= \boldsymbol{k}_{j-1}(\boldsymbol{x})^\top \boldsymbol{v}_{\boldsymbol{x}} + c(\boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}))^2 && (88)\\
&\quad - 2\boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}) c k(\boldsymbol{x}, \boldsymbol{x}_j) + c k(\boldsymbol{x}, \boldsymbol{x}_j)^2 && (89)\\
&\quad /\!/ \textit{ evaluating the vector product} \\
&= \boldsymbol{k}_{j-1}(\boldsymbol{x})^\top \boldsymbol{v}_{\boldsymbol{x}} + c(k(\boldsymbol{x}, \boldsymbol{x}_j) - \boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}))^2 && (90)\\
&\quad /\!/ \textit{ rearranging terms into a quadratic} \\
&= q_{j-1}(\boldsymbol{x}) + c(k(\boldsymbol{x}, \boldsymbol{x}_j) - \boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}))^2 && (91)\\
&\quad /\!/ \textit{ definition of } q_{j-1}(\boldsymbol{x})
\end{aligned}
$$

This shows that

$$q_j(\boldsymbol{x}) = q_{j-1}(\boldsymbol{x}) + r_j(\boldsymbol{x}) \text{ , where} \tag{92}$$
$$r_j(\boldsymbol{x}) := c(k(\boldsymbol{x}, \boldsymbol{x}_j) - \boldsymbol{v}_{\boldsymbol{x}_j}^\top \boldsymbol{k}_{j-1}(\boldsymbol{x}))^2 \geq 0. \tag{93}$$

□

Lemma 24. *The term $\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}$ can be written as*

$$\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} = \sum_{n=1}^{N} \frac{(y_n - m_{n-1}(\boldsymbol{x}_n))}{k_{n-1}(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2}. \tag{94}$$

Proof. Define

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

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

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

This allows to write

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

Lemma 25. *For all $t, m \in \mathbb{N}$ with $1 \leq t + m \leq N$*

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

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

Proof.

$$
\begin{aligned}
&k_{t+m}(\boldsymbol{x}_a, \boldsymbol{x}_b) \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - k(\boldsymbol{x}_a, \boldsymbol{X}_{t+m}) \boldsymbol{A}_{t+m}^{-1} k(\boldsymbol{X}_{t+m}, \boldsymbol{x}_b) \\
&\text{// by definition} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \begin{bmatrix} k(\boldsymbol{X}_t) + \sigma^2 \boldsymbol{I}_t & k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) & k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \end{bmatrix}^{-1} \begin{bmatrix} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// in block notation} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_t & k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) & k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \end{bmatrix}^{-1} \begin{bmatrix} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// using the definition of } \boldsymbol{A}_t \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \cdot \\
&\begin{bmatrix} \boldsymbol{A}_t^{-1} + \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} & -\boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \right)^{-1} \\ - \left(k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} & \left(k(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \right)^{-1} \end{bmatrix} \cdot \\
&\begin{bmatrix} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// applying block-matrix inversion} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \cdot \\
&\begin{bmatrix} \boldsymbol{A}_t^{-1} + \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} & -\boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} \\ - \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} & \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} \end{bmatrix} \cdot \\
&\begin{bmatrix} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \\ k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// applying the definition of } k_t \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \cdot \\
&\begin{bmatrix} \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) + \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) - \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \\ - \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) + \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// evaluating multiplication with right-most vector} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \cdot \\
&\begin{bmatrix} \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) - \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} \left(k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \right) \\ \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} \left(k(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) - k(\overline{\boldsymbol{X}}, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \right) \end{bmatrix} \\
&\text{// rearranging} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - \begin{bmatrix} k(\boldsymbol{x}_a, \boldsymbol{X}_t) & k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \end{bmatrix} \cdot \\
&\begin{bmatrix} \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) - \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \\ \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \end{bmatrix} \\
&\text{// applying the definition of } k_t \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - k(\boldsymbol{x}_a, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) \\
&\quad + k(\boldsymbol{x}_a, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) - k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \\
&\text{// evaluating the vector product} \\
&= k(\boldsymbol{x}_a, \boldsymbol{x}_b) - k(\boldsymbol{x}_a, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \boldsymbol{x}_b) - \left(k(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) - k(\boldsymbol{x}_a, \boldsymbol{X}_t) \boldsymbol{A}_t^{-1} k(\boldsymbol{X}_t, \overline{\boldsymbol{X}}) \right) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \\
&\text{// rearranging} \\
&= k_t(\boldsymbol{x}_a, \boldsymbol{x}_b) - k_t(\boldsymbol{x}_a, \overline{\boldsymbol{X}}) \left(k_t(\overline{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}_t \right)^{-1} k_t(\overline{\boldsymbol{X}}, \boldsymbol{x}_b) \\
&\text{// applying the definition of } k_t
\end{aligned}
$$

□

Lemma 26.

$$\sum_{j=t+1}^{n}\sum_{i=t_0+1}^{j-1} 1 = (n-t)\left(\frac{n+t-1}{2} - t_0\right) \tag{95}$$

Proof.

$$\begin{aligned}
\sum_{j=t+1}^{n}\sum_{i=t_0+1}^{j-1} 1 &= \sum_{j=t+1}^{n}(j-1-t_0) && (96)\\
&= \sum_{j=0}^{n-t-1}(j-1-t_0+t+1) && (97)\\
&= \sum_{j=0}^{n-t-1}(j+t-t_0) && (98)\\
&= (t-t_0)(n-t) + \sum_{j=0}^{n-t-1} j && (99)\\
&= (t-t_0)(n-t) + \frac{(n-t-1)(n-t)}{2} && (100)\\
&= (n-t)\left(\frac{n-t-1}{2} + t - t_0\right) && (101)\\
&= (n-t)\left(\frac{n+t-1}{2} - t_0\right) && (102)
\end{aligned}$$

□

Lemma 27 (Link between the Cholesky and Gaussian process regression)**.** *Denote with $\boldsymbol{C}_N$ the Cholesky decomposition of $\boldsymbol{K}$, so that $\boldsymbol{C}_N\boldsymbol{C}_N^\top = \boldsymbol{K}$. The n-th diagonal element of $\boldsymbol{C}_N$, squared, is equivalent to $k_{n-1}(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma^2$:*

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

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

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

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

To remain within the text margins, define

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

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

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

Also x can be simplified further.

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

□