Randomized Algorithms for Gaussian Process Regression

ANCHP Lab, EPFL

Matthias Zeller

June 10, 2022

1 Introduction

Gaussian process (GP) regression is a non-parametric supervised method that extends Bayesian linear regression [22, 20, 18, 8]. Considering the problem of inferring an unknown function f given some inputs and noisy outputs, the GP regression setting models the prior on f as a GP. Based on a-priori knowledge (e.g. the smoothness of f), one sets the GP mean and covariance functions and performs inference in the induced function space.

GP models are attractive due to their ability to approximate a wide range of functions with interpretable hyperparameters [20, 5], they provide uncertainty estimates [20, 24] and cast hyperparameter selection as an optimization problem [15, 20, 24]. However, GP inference is traditionally done with Cholesky decomposition and requires $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ time complexity, with n the number of samples in the training set, therefore suffering from scalability to large datasets. The GPyTorch framework [10] proposes a more efficient approach for GP inference that i) achieves sub-cubic time complexity for exact[1] inference under some suitable conditions[2], ii) requires the user to only provide blackbox matrix-matrix multiplication oracles with the kernel matrix and its derivative. As a consequence the framework disentangles the model from the inference engine, greatly easing implementation for the user, specifically for complex GP models (e.g., combining different kernels or using GP approximations for large datasets).

The core of GPyTorch is based on a combination of conjugate gradients, Lanczos quadrature and stochastic trace estimation to compute the likelihood of the model hyperparameters and its gradient. A specialized preconditioner is used to speedup CG convergence and reduce the overall complexity.

The rest of this introduction reviews GP regression and presents the computational challenges involved in determining unknown parameters (i.e., training the model). In section 2 we present the algorithms involved in GPyTorch and the underlying theory. Section 3 exposes a preconditioning approach suited for some kernel matrices. The need and the efficiency of preconditioning depends on the spectrum of the kernel matrix, which is discussed in section 4. Numerical experiments are eventually presented in section 5.

1.1 Gaussian Process Regression Background

Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and assuming there is an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ following the measurement model with Gaussian noise

[1] Exact refers to the fact that one can choose parameters of the algorithms to reach machine precision, although those parameters may be unknown apriori.

[2] In the paper, it is stated that GPyTorch reduces exact GP inference to $\mathcal{O}(n^2)$ time complexity, which may be true in practice depending on the dataset at hand. This is related to the data distribution and the spectrum of the kernel matrix, see the discussion in the results section.

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \ , \tag{1}$$

the typical machine learning task is to predict $f(\mathbf{x}^\star)$ for test samples $\mathbf{x}^\star$ that were not in the training set. Gaussian process methods achieve this by modeling the prior on f to be a Gaussian Process.

Definition 1.1 (see [28])**.** *A Gaussian Process (GP) is a stochastic process whose any finite number of points have a joint Gaussian distribution. We denote $f \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ for f being a Gaussian process with mean function $\mu(\mathbf{x}) := \mathbb{E}[f(\mathbf{x})]$ and covariance function $k(\mathbf{x}, \mathbf{x}') := \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$.*

GP models perform probabilistic inference in the function space $\mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. One can loosely think of f as an infinite vector $(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots)$ of function evaluations at all possible locations, and we impose a distribution of all finite subsets $(f(\mathbf{x}'_1), \ldots, f(\mathbf{x}'_n))$. Choosing the mean and covariance functions incorporate prior knowledge on the shape of f. One typically assumes a zero mean ($\mu(\mathbf{x}) := 0)$) as the data can be shifted to have a zero empirical mean, the true object of interest being the covariance function. Remaining consistent with the literature, we will call k the kernel function, in reference to the fact that GP models leverage the *kernel trick* to implicitly perform computations in high-dimensional spaces (see section 2.1.2 of [20] for details).

To see how the choice of the kernel function induces a distribution over function, we first define the matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{d \times n}$ collecting the training features and define the $n \times n$ kernel matrix

$$(K_{XX})_{ij} = (k(X, X))_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \quad 1 \leq i, j \leq n \ .$$

Similarly for a test sample $\mathbf{x}^\star$, we define $\mathbf{k}_{X,\mathbf{x}^\star} = k(X, \mathbf{x}^\star)$, $k_{\mathbf{x}^\star} = k(\mathbf{x}^\star, \mathbf{x}^\star)$. A hat symbol denotes addition of the identity matrix scaled by the noise variance, e.g. $\widehat{K}_{XX} = K_{XX} + \sigma^2\mathbb{I}$. From Equation 1, the joint distribution of the prediction $y^\star = f(\mathbf{x}^\star)$ with the training observations $\mathbf{y} = (y_1, \ldots, y_n)$ is

$$\begin{bmatrix} y^\star \\ \mathbf{y} \end{bmatrix} \mid X, \mathbf{x}^\star \ \sim \ \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k_{\mathbf{x}^\star} & \mathbf{k}_{X\mathbf{x}^\star}^\top \\ \mathbf{k}_{X\mathbf{x}^\star} & \widehat{K}_{XX} \end{bmatrix}\right) \ . \tag{2}$$

The distribution of interest is the *posterior* or *predictive* distribution, obtained by conditioning (2) on the training data. The resulting distribution is also Gaussian with a well-known expression (see e.g. theorem 3.3.4 of [26]),

$$y^\star \mid \mathbf{x}^\star, X, \mathbf{y} \sim \mathcal{N}(\mu, \Sigma) \ , \quad \begin{aligned} \mu &= \mathbf{k}_{X\mathbf{x}^\star}^\top \widehat{K}_{XX}^{-1} \mathbf{y} \\ \Sigma &= k_{\mathbf{x}^\star} - \mathbf{k}_{X\mathbf{x}^\star}^\top \widehat{K}_{XX}^{-1} \mathbf{k}_{X\mathbf{x}^\star} \end{aligned} \ . \tag{3}$$

As a brief interpretation of (3), notice that without any training data we have $\Sigma = k_{\mathbf{x}^\star}$ being the prior variance. By adding data we subtract the positive quadratic term, decreasing the uncertainty of the prediction. The predictive mean can be written

$$\mu = \sum_{i=1}^{n} \alpha_i y_i = \sum_{i=1}^{n} \beta_i k(\mathbf{x}_i, \mathbf{x}^\star) \ ,$$

with $\boldsymbol{\alpha} := \widehat{K}_{XX}^{-1} \mathbf{k}_{X\mathbf{x}^\star}$, $\boldsymbol{\beta} := \widehat{K}_{XX}^{-1} \mathbf{y}$. This provides two equivalent views for the prediction mean, either as a weighted sum of the training targets y_i, or a weighted sum of kernel functions.

1.2 Training a Gaussian Process & Hyperparameter Estimation

Gaussian Process models require as a first step to determine some unknown parameters (generically noted $\boldsymbol{\theta}$) before performing prediction with (3), this step is referred as *training* the model. Indeed, kernel functions are typically parameterized, e.g. the popular squared exponential kernel $k(\mathbf{x}_i, \mathbf{x}_j; \ \ell) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\ell}$ is characterized by the lengthscale ℓ. In addition, the noise variance σ^2 in (1) appearing in the kernel matrix $\widehat{K}_{XX}$ is also a free parameter.

Those quantities are unknown apriori and have be to estimated from the data. The typical approach is *maximum likelihood estimation* with an optimization algorithm using gradient information, obtaining an estimate $\widehat{\boldsymbol{\theta}}$ that can be plugged into equations. In this setting, the likelihood would be the density $p(\mathbf{f} \mid X, \boldsymbol{\theta})$, with $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))$ the function values. However, true function values $\mathbf{f}$ are unknown and we instead consider the *marginal* likelihood

$$p(\mathbf{y} \mid X, \boldsymbol{\theta}) = \int p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathbf{f}) p(\mathbf{f} \mid X, \boldsymbol{\theta}) d\mathbf{f} \ ,$$

where marginalization occurs over the latent function values. The function to optimize is then the marginal log likelihood, noted $\mathcal{L}(\boldsymbol{\theta} \mid X, \mathbf{y}) := \log p(\mathbf{y} \mid X, \boldsymbol{\theta})$. From (2), we have $\mathbf{y} \mid X, \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \widehat{K}_{XX})$, so that the likelihood and its gradient (see Appendix A for the derivation) read

$$\mathcal{L}(\boldsymbol{\theta} \mid X, \mathbf{y}) = -\frac{1}{2}\mathbf{y}^\top \widehat{K}_{XX}^{-1}\mathbf{y} - \frac{1}{2}\log\det\left(\widehat{K}_{XX}\right) - \frac{n}{2}\log(2\pi) \tag{4}$$

$$\frac{d\mathcal{L}}{d\theta_i}(\boldsymbol{\theta} \mid X, \mathbf{y}) = \frac{1}{2}\mathbf{y}^\top \widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta_i}\widehat{K}_{XX}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{Tr}\left(\widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta_i}\right) \tag{5}$$

Evaluation of the above quantities is challenging because of three dominant terms: the linear solve $\widehat{K}_{XX}^{-1}\mathbf{y}$, the log determinant $\log\det\widehat{K}_{XX}$ and the trace term $\operatorname{Tr}(\widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta_i})$. Those computations are traditionally addressed by the Cholesky decomposition $\widehat{K}_{XX} = LL^\top$. However, this decomposition has cubic time complexity $\mathcal{O}(n^3)$ with n the number of observations. In this project, we hence focus on approximating the likelihood and its gradient by using iterative methods and stochastic estimates, trading exactness for speed, as done in the GPyTorch inference engine [10].

2 Algorithms for Likelihood Evaluation

In this section, we present the approach of computing in sub-cubic time the three dominant terms $\widehat{K}_{XX}^{-1}\mathbf{y}$, $\log\det\widehat{K}_{XX}$, $\operatorname{Tr}(\widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta})$ required to train a GP. The structure is as follows: we first present in section 2.1 stochastic trace estimation for estimating the trace term (avoiding computing the matrix-matrix product) and the log-determinant. Section 2.2 introduces the Lanczos algorithm and the Lanczos quadrature to estimate quadratic forms arising from log-determinant estimation. We then present the CG algorithm for the linear solves with the kernel matrix, and discuss how to recycle CG computations to virtually run Lanczos tridiagonalization. Eventually, we briefly mention the core algorithm of GPyTorch, which combines the aforementioned algorithms to jointly compute the three quantities required for likelihood evaluation.

2.1 Stochastic Trace Estimation

For a matrix $A \in \mathbb{R}^{n\times n}$, consider the estimator $\mathbf{z}^\top A\mathbf{z}$ with a random probe vector $\mathbf{z} \sim \mathcal{D}$. It is easy to show that this estimator is unbiased for $\operatorname{Tr}(A)$ if $\mathbb{E}_{\mathcal{D}}[\mathbf{z}] = \mathbf{0}, \mathbb{V}\mathrm{ar}_{\mathcal{D}}[\mathbf{z}] = \mathbb{I}$. From this, we build the Monte Carlo estimator with N probe vectors

$$\operatorname{Tr}_N^{\mathcal{D}} := \frac{1}{N}\sum_{i=1}^N \mathbf{z}_i^\top A\mathbf{z}_i, \quad \mathbf{z}_i \overset{\text{iid}}{\sim} \mathcal{D} \ ,$$

having $\mathcal{O}(Nn^2)$ complexity, provided that sampling from $\mathcal{D}$ is not the bottleneck. We will restrict our discussion for $\mathcal{D}$ being the standard normal distribution and the symmetric case $A = A^\top$, we note $\operatorname{Tr}_N^G(A)$ the corresponding estimator and show its variance in the following proposition.

Proposition 2.1. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix, then* $\mathbb{V}\mathrm{ar}[\operatorname{Tr}_N^G] = 2\|A\|_F^2/N$.

Proof. Let $A = Q\Lambda Q^\top$ be the spectral decomposition of A, with $\mathbf{q}_i$ the columns of Q. The quadratic forms are written as $\mathbf{z}^\top A\mathbf{z} = \sum_{i=1}^n \lambda_i y_i^2$ with $y_i := \mathbf{z}^\top\mathbf{q}_i \sim \mathcal{N}(0,1)$ independent, so that y_i^2 is distributed as χ_1^2 with a well-known variance $\mathbb{V}\mathrm{ar}[y_i^2] = 2$. By independence of the y_i's,

$$\mathbb{V}\mathrm{ar}[\mathbf{z}^\top A\mathbf{z}] = \sum_{i=1}^{n} \lambda_i^2 \, \mathbb{V}\mathrm{ar}[y_i^2] = 2\sum_{i=1}^{n} \lambda_i^2 = 2\|A\|_F^2.$$

The statement follows by again using independence of the probe vectors $\mathbf{z}_i$. □

Applying trace estimation to $B := \widehat{K}_{XX}^{-1} \frac{d\widehat{K}_{XX}}{d\theta}$, we have

$$\mathrm{Tr}\,(B) \approx \mathrm{Tr}_N^G\,(B) = \frac{1}{N}\sum_{i=1}^{N} \left(\mathbf{z}_i^\top \widehat{K}_{XX}^{-1}\right)\left(\frac{d\widehat{K}_{XX}}{d\theta}\mathbf{z}_i\right) \; . \tag{6}$$

where the complexity is now dominated by the linear solves $\widehat{K}_{XX}^{-1}\mathbf{z}_i$, this remains to be addressed. For the log determinant, since $\widehat{K}_{XX}$ is SPD,

$$\log(\det \widehat{K}_{XX}) = \mathrm{Tr}(\log \widehat{K}_{XX}) \approx \mathrm{Tr}_N^G(\log \widehat{K}_{XX}) \; .$$

and the complexity is dominated by the matrix logarithm[3] of $\widehat{K}_{XX}$ involved in the quadratic forms. The next section discusses approximation of those quadratic forms and the convergence of the log-determinant estimator.

2.2 Stochastic Lanczos Quadrature

Computing the log determinant with stochastic trace estimation involves quadratic forms $\mathbf{z}^\top \log(\widehat{K}_{XX})\mathbf{z}$, which are expensive to compute and thus need to be approximated. We start by presenting the general idea and main results of Lanczos quadrature (see [13] for details) for such an approximation, show the Lanczos algorithm and its relationship with the Gauss quadrature rule and give convergence bounds when this approximation is paired with trace estimation.

The first step of Lanczos quadrature is to express the quadratic form as a Riemann-Stieljes integral with respect to an unknown measure α. Given a symmetric matrix A and a function f that is analytic in the spectrum interval $[\lambda_{\min}, \lambda_{\max}]$, we have,

$$\mathbf{x}^\top f(A)\mathbf{x} = \mathbf{x}^\top Q f(\Lambda) Q^\top \mathbf{x} = \sum_i f(\lambda_i)(Q^\top \mathbf{x})_i^2 = \int_{\lambda_{\min}}^{\lambda_{\max}} f(\lambda) d\alpha(\lambda) =: I \; ,$$

with the measure $\alpha(\lambda) = \sum_i w_i^2 I_{\{\lambda_i \le \lambda < \lambda_{i+1}\}}$, $w_i := (Q^\top \mathbf{x})_i$, i.e. it is piecewise constant with known jumps w_i^2 but unknown jump nodes λ_i. The idea is now to approximate the integral with an m-point quadrature rule,

$$I_m := \sum_{i=1}^{m} b_i f(c_i) \approx I \; , \tag{7}$$

with $m \ll n$. The Lanczos quadrature relies on running the Lanczos method to obtain the weights b_i and nodes c_i (Equation (7)) of the Gauss quadrature rule. We state below the Lanczos method (Algorithm 1) for solving linear systems, briefly explain how this algorithm is related to Krylov subspace methods, and cite the theorem relating this method with the Gauss quadrature.

[3]The matrix logarithm is a matrix function, which for diagonalizable matrices $A = X\Lambda X^{-1}$ is defined as $f(A) = X\mathrm{diag}(f(\lambda_1), \ldots, f(\lambda_n))X^{-1}$ when f is analytic in the spectrum of A. See chapter 9 of [14] for details.

Algorithm 1: Lanczos algorithm to solve linear system $\widehat{K}_{XX}\mathbf{x} = \mathbf{b}$ (Algorithm 6.16 of [21])

```
Input  : matrix-vector multiplication oracles $\mathbf{a} \mapsto \widehat{K}_{XX}\mathbf{a}$, right-hand side $\mathbf{b}$, starting vector $\mathbf{x}_0$,
         number of steps $m$
Output: approximate solution $\mathbf{x}_m \approx \widehat{K}_{XX}^{-1}\mathbf{b}$, orthonormal basis $V_m$, tridiagonal matrix $T_m$
$\mathbf{r}_0 \leftarrow \mathbf{b} - \widehat{K}_{XX}\mathbf{x}_0,\ \beta \leftarrow \|\mathbf{r}_0\|_2$
$\mathbf{v}_1 \leftarrow \mathbf{r}_0/\beta$
for $k = 1, \dots, m$ do
    $\tilde{\mathbf{w}} \leftarrow \widehat{K}_{XX}\mathbf{v}_k - \eta_k\mathbf{v}_{k-1}$        // if k=1, set $\eta_1\mathbf{v}_0 := \mathbf{0}$
    $\delta_k \leftarrow \mathbf{v}_k^\top\tilde{\mathbf{w}}$
    $\mathbf{w}_k \leftarrow \tilde{\mathbf{w}} - \delta_k\mathbf{v}_k$
    $\eta_k \leftarrow \|\mathbf{w}_k\|_2$
    $\mathbf{v}_{k+1} \leftarrow \mathbf{w}_l/\eta_k$
end
Note $V_m = \begin{bmatrix}\mathbf{v}_1 & \dots & \mathbf{v}_m\end{bmatrix}$, $T_m = \mathrm{tridiag}(\eta_{k-1}, \delta_k, \eta_k)$
Compute $\mathbf{y}_m = \beta T_m^{-1}\mathbf{e}_1$ and $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$
```

We briefly explain the rationale behind the algorithm, as this will be useful later, and refer to [21, 14] for the details. We replace $\widehat{K}_{XX}$ by a generic symmetric matrix A for notation simplicity. The algorithm builds an orthonormal basis $V_m \in \mathbb{R}^{n\times m}$ for the Krylov subspace

$$\mathcal{K}_m(A, \mathbf{r}_0) := \mathrm{span}\{\mathbf{r}_0, A\mathbf{r}_0, \dots, A^{m-1}\mathbf{r}_0\}\ .$$

The computed quantities can be arranged as the Arnoldi decomposition,

$$AV_m = V_mT_m + \eta_m\mathbf{v}_{m+1}\mathbf{e}_m^\top, \quad T_m = \begin{bmatrix}\delta_1 & \eta_1 & & & \\ \eta_1 & \delta_2 & \eta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_{m-2} & \delta_{m-1} & \eta_{m-1} \\ & & & \eta_{m-1} & \delta_m\end{bmatrix}\ . \tag{8}$$

The approximate solution $\mathbf{x}_m$ is extracted from the affine subspace $\mathbf{x}_0 + \mathcal{K}_m(A, \mathbf{r}_0)$, whose expression is found by imposing the Galerkin condition $\mathbf{r}_m \perp \mathcal{K}_m(A, \mathbf{r}_0)$. Reformulating this condition in terms of orthogonality with basis elements $\mathbf{v}_k$, and noting from (8) that $V_m^\top AV_m = T_m$, we have

$$\begin{aligned}\mathbf{r}_m \perp \mathcal{K}_m(A, \mathbf{r}_0) &\iff \mathbf{0} = V_m^\top\mathbf{r}_m = V_m^\top(\mathbf{b} - A\mathbf{x}_0 - AV_m\mathbf{y}_m) = V_m^\top\mathbf{r}_0 - V_m^\top AV_m\mathbf{y}_m \\ &\iff \mathbf{y}_m = T_m^{-1}\mathbf{e}_1\|\mathbf{r}_0\|_2\ .\end{aligned} \tag{9}$$

From this, we relate residuals and basis elements of the Krylov subspace, showing that residuals are orthogonal to each other and that $\|\mathbf{r}_k\|_2$ can be cheaply evaluated at each step k,

$$\begin{aligned}\mathbf{r}_k &:= \mathbf{b} - A\mathbf{x}_k = \mathbf{r}_0 - AV_k\mathbf{y}_k \\ &\overset{(8)}{=} \mathbf{r}_0 - V_kT_k\mathbf{y}_k - \eta_k\mathbf{v}_{k+1}\mathbf{e}_k^\top\mathbf{y}_k \\ &\overset{(9)}{=} -(\eta_k\mathbf{e}_k^\top\mathbf{y}_k)\mathbf{v}_{k+1}\ .\end{aligned} \tag{10}$$

The next theorem relates this algorithm with the Gauss quadrature.

Theorem 2.2 (Theorem 6.2 of [13])**.** *Consider the Arnoldi decomposition* (8) *of the symmetric matrix $A \in \mathbb{R}^{n\times n}$ after m steps of the Lanczos iteration. Let $(\mu_i, \mathbf{u}_i)$ be the Ritz pairs of A, i.e. the eigenpairs of T_m, such that $\|\mathbf{u}_i\|_2 = 1$. Then the m-point Gauss quadrature formula approximating $\mathbf{v}_1^\top f(A)\mathbf{v}_1$ is given by the nodes $c_i = \mu_i$ and the weights $b_i = (\mathbf{e}_1^\top\mathbf{u}_i)^2$.*

Putting things together, running Algorithm 1 on the matrix $\widehat{K}_{XX}$ with right-hand size $\mathbf{z}$ and starting vector $\mathbf{x}_0 = \mathbf{0}$ will give us the tridiagonal matrix T_m. We can then efficiently compute its spectral decomposition $T_m = UMU^\top$ (since $m \ll n$) and approximate any quadratic form $\mathbf{z}^\top f(\widehat{K}_{XX})\mathbf{z}$,

$$I_m = \sum_{i=1}^{m} b_i f(c_i) = \|\mathbf{z}\|_2^2 \sum_{i=1}^{m} f(\mu_i)(\mathbf{e}_1^\top \mathbf{u}_i)^2 = \|\mathbf{z}\|_2^2\, \mathbf{e}_1^\top U f(M) U^\top \mathbf{e}_1 = \|\mathbf{z}\|_2^2\, \mathbf{e}_1^\top f(T_m)\mathbf{e}_1 \; . \tag{11}$$

Combining this approximation with stochastic trace estimation yields the stochastic Lanczos quadrature, which we use to approximate the log determinant. Noting $T_m^{(i)}$ the tridiagonal matrix corresponding to the right-hand side $\mathbf{z}_i$, we note $\Gamma_{N,m}$ the estimate defined below and we state its convergence properties in the following theorem,

$$\Gamma_{N,m} := \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{z}_i\|_2^2\, \mathbf{e}_1^\top \log(T_m^{(i)})\mathbf{e}_1 \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i^\top \log(\widehat{K}_{XX})\mathbf{z}_i \approx \log(\det \widehat{K}_{XX}) \; .$$

Theorem 2.3 (Theorem 19 of [7]). *Let $A \in \mathbb{R}^{n\times n}$ be an SPD matrix with $n \geq 2$. Suppose that*

- $N \geq \frac{16}{\epsilon^2}(\|\log A\|_F^2 + \epsilon\|\log A\|_2)\log\frac{4}{\delta}$*, and* $N \leq \frac{\delta}{2}e^{n^2/16}$
- $m \geq \frac{\sqrt{\kappa(A)+1}}{4}\log\left(\frac{4}{\epsilon}n^2\log(2\kappa(A))(\sqrt{\kappa(A)+1}+1)\right)$

Then the log-determinant approximation satisfies the absolute error bound $\mathbb{P}(|\Gamma_{N,m} - \log\det A| \geq \epsilon) \leq \delta$.

Turning the above theorem into a relative error bound (provided a non-zero log-determinant), for a fixed ϵ, δ we need the number of probe vectors N to roughly scale with $\|\log \widehat{K}_{XX}\|_F^2 / \operatorname{Tr}(\log \widehat{K}_{XX})^2$. Unlike Theorem 2 of [10], we have a matrix-dependent bound for the number of probe vectors N. However, it is unclear at this point whether Theorem 2.3 improves the bound on N over [10] as $\log \widehat{K}_{XX}$ is not necessarily PSD. Preconditioning will later address this issue, as well as reducing the required number of Lanczos steps m.

2.3 Conjugate Gradients Algorithm

We now discuss approximation of the linear solve $\widehat{K}_{XX}^{-1}\mathbf{y}$ required for the likelihood and the solves $\widehat{K}_{XX}^{-1}\mathbf{z}_i$ involved in estimation of the trace term, Equation (6). We will use the well-known conjugate gradients method, given that we are working with $\widehat{K}_{XX}$ SPD and because it is related to the Lanczos iteration, which will allow us to save computations as discussed later. For now, we state the algorithm and its convergence rate (just assume for now that the preconditioner is $P = \mathbb{I}$).

Algorithm 2: Preconditioned Conjugate Gradients to solve $\widehat{K}_{XX}\mathbf{u} = \mathbf{b}$ (Algorithm 9.1 of [21])

```
Input  : matrix-vector multiplication oracles $\mathbf{a} \mapsto \widehat{K}_{XX}\mathbf{a}$ and $\mathbf{a} \mapsto P^{-1}\mathbf{a}$, right-hand side $\mathbf{b}$,
         number of steps $m$
Output: approximate solution $\mathbf{u}_m \approx \widehat{K}_{XX}^{-1}\mathbf{b}$
// Initialize solution and residuals
$\mathbf{u}_0 \leftarrow \mathbf{0},\ \mathbf{r}_0 \leftarrow \mathbf{b}$
// Initialize preconditioned residuals and first search direction
$\mathbf{z}_0 \leftarrow P^{-1}\mathbf{r}_0$
$\mathbf{d}_1 \leftarrow \mathbf{r}_0$
for $k = 1, \dots, m$ do
    // Line search:  optimal step size in direction $\mathbf{d}_k$
    $\alpha_k \leftarrow \mathbf{r}_k^\top \mathbf{z}_k \,/\, \mathbf{d}_k^\top(\widehat{K}_{XX}\mathbf{d}_k)$
    // Update solution and residuals
    $\mathbf{u}_k \leftarrow \mathbf{u}_{k-1} + \alpha_k \mathbf{d}_k$
    $\mathbf{r}_k \leftarrow \mathbf{r}_{k-1} - \alpha_k \widehat{K}_{XX}\mathbf{d}_k$
    $\mathbf{z}_k \leftarrow P^{-1}\mathbf{r}_k$
    // Conjugate Gram Schmidt and compute next search direction
    $\beta_k \leftarrow \mathbf{r}_k^\top \mathbf{z}_k \,/\, \mathbf{r}_{k-1}^\top \mathbf{z}_{k-1}$
    $\mathbf{d}_{k+1} \leftarrow \mathbf{z}_k + \beta_k \mathbf{d}_k$
end
```

Theorem 2.4 (Convergence of Conjugate Gradients [21, 14])**.** *After m steps of conjugate gradients for solving the linear system $\widehat{K}_{XX}\mathbf{u} = \mathbf{b}$, the approximate $\mathbf{u}_m$ has the error*

$$\|\mathbf{u} - \mathbf{u}_m\|_{\widehat{K}_{XX}} \le 2\|\mathbf{u} - \mathbf{u}_0\|_{\widehat{K}_{XX}} \left(\frac{\sqrt{\kappa(\widehat{K}_{XX})} - 1}{\sqrt{\kappa(\widehat{K}_{XX})} + 1} \right)^m ,$$

with $\mathbf{u}_0$ the starting vector and the norm $\|\mathbf{u}\|_A := (\mathbf{u}^\top A\mathbf{u})^{1/2}$.

Similarly to Theorem 2.3, convergence depends on the square root of the condition number, we will later introduce a specialized preconditionner to improve convergence.

We now discuss the relationship between the Lanczos iteration and CG. In fact, CG can be entirely derived from the Lanczos algorithm for linear systems, and CG is often introduced in this way (see e.g. section 6.7.1 of [21]). However, in this case, we are only interested in the Lanczos tridiagonal matrices $T_m^{(i)}$ of Equation (8), which can also be derived from CG. One first notices that Lanczos algorithm performs computations with basis elements $\mathbf{v}_{k+1}$, whereas CG manipulates residuals $\mathbf{r}_k$, and Equation (10) relates the two. Exploiting this and the fact that CG search directions are A-conjugate (i.e., $\mathbf{d}_k^\top A\mathbf{d}_k = 0$), one starts from the definition of η_k, δ_k in Algorithm 1 and obtains after a few manipulations

$$T_m = \begin{bmatrix} \frac{1}{\alpha_1} & \frac{\sqrt{\beta_1}}{\alpha_1} & & & \\ \frac{\sqrt{\beta_1}}{\alpha_1} & \frac{1}{\alpha_2} + \frac{\beta_1}{\alpha_1} & \frac{\sqrt{\beta_2}}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{\sqrt{\beta_{m-1}}}{\alpha_{m-1}} \\ & & & \frac{\sqrt{\beta_{m-1}}}{\alpha_{m-1}} & \frac{1}{\alpha_m} + \frac{\beta_{m-1}}{\alpha_{m-1}} \end{bmatrix}, \tag{12}$$

see section 6.7.3 of [21] for the full derivation. This means one can run m steps of CG for a right-hand side $\mathbf{z}_i$ to obtain $\mathbf{u}_m \approx \widehat{K}_{XX}^{-1}\mathbf{z}_i$ and then retrieve $T_m^{(i)}$ in $\mathcal{O}(m)$ time.

2.4 GPyTorch Engine

The core of the GPyTorch inference engine [10] is based on a modified batched version of conjugate gradients (mBCG). This combines all the aforementioned methods, hence we let the corresponding algorithm in the Appendix B. mBCG relies on running CG with an SPD preconditioner P to approximate the matrix equation

$$\widehat{K}_{XX}\begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \dots & \mathbf{u}_N\end{bmatrix} = \begin{bmatrix}\mathbf{y} & \mathbf{z}_1 & \dots & \mathbf{z}_N\end{bmatrix} ,$$

with $\mathbf{y}$ the observations and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, P)$ probe vectors that must be sampled before running the algorithm. Recycling CG computations, mBCG also returns partial Lanczos tridiagonalizations $T_m^{(i)}$ using Equation 12. GPyTorch is build on top of the PyTorch library[4], a library handling tensor operations on GPUs. Using several right-hand sides indeed exploits parallel hardware better. Moreover, iterative methods being intrinsically sequential, mBCG relies on preconditioning to trade fewer steps against more effort per step.

3 Preconditioning

The motivation for preconditioning has been discussed in Theorem 2.3 and Theorem 2.4. Both linear solves and log-determinant estimations depend on $\sqrt{\kappa(\widehat{K}_{XX})}$. A large condition number will force us to run a large number of CG iterations ($m \to n$) for each of the N probe vectors, and our approximations may compare unfavorably to the Cholesky decomposition for computing the likelihood exactly. We start by assuming we have a good preconditioner P for CG and explain how our random estimators must be modified. We then present the specialized preconditioner used in [10] and some related theoretical results.

3.1 Modified Algorithms for Preconditioning

Given an SPD preconditioner $P \approx \widehat{K}_{XX}$, we precondition the system $\widehat{K}_{XX}\mathbf{u} = \mathbf{z}$ as

$$\left(P^{-\frac{1}{2}}\widehat{K}_{XX}P^{-\frac{1}{2}}\right)\left(P^{\frac{1}{2}}\mathbf{u}\right) = P^{-\frac{1}{2}}\mathbf{z} . \tag{13}$$

The preconditioned CG (pCG) Algorithm 2 will compute the solution $\mathbf{u}$ while only requiring access to P^{-1}. Let us denote the preconditioned matrix $M := P^{-\frac{1}{2}}\widehat{K}_{XX}P^{-\frac{1}{2}}$. Then pCG convergence is as in Theorem 2.4, replacing the condition number by $\kappa(M)$. Moreover, the tridiagonal matrix T_m of Equation (12) is now a partial tridiagonalization of M with starting vector $P^{-\frac{1}{2}}\mathbf{z}$. In turn, Lanczos quadrature will return the approximate

$$\|P^{-\frac{1}{2}}\ \mathbf{z}\|_2^2\mathbf{e}_1^\top \log(T_m)\mathbf{e}_1 \approx \mathbf{z}^\top P^{-\frac{1}{2}}\log(M)P^{-\frac{1}{2}}\mathbf{z} .$$

This can be made an estimator of $\log(\det M)$ if $P^{-\frac{1}{2}}\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. As a consequence, we change the distribution of probe vectors to $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, P)$. In order to recover the original log-determinant, one simply exploits

$$\log\det\widehat{K}_{XX} = \log\det P + \log\det\left(P^{-\frac{1}{2}}\widehat{K}_{XX}P^{-\frac{1}{2}}\right) , \tag{14}$$

where $\log(\det P)$ is computed exactly as discussed later. Because of the new distribution of probe vectors, we change the estimate of $\mathrm{Tr}(\widehat{K}_{XX}^{-1}(d\widehat{K}_{XX}/d\theta))$ in Equation 6 by first noticing that

$$\mathrm{Tr}(A) = \mathrm{Tr}\left(A\,\mathbb{E}_{\mathcal{N}(\mathbf{0},P)}\left[P^{-1}\mathbf{z}\mathbf{z}^\top\right]\right) = \mathbb{E}_{\mathcal{N}(\mathbf{0},P)}\left[\mathrm{Tr}(AP^{-1}\mathbf{z}\mathbf{z}^\top)\right] = \mathbb{E}_{\mathcal{N}(\mathbf{0},P)}\left[\mathbf{z}^\top AP^{-1}\mathbf{z}\right] ,$$

by linearity of expectation and the trace and by the cyclic property of the trace. The new estimator is thus

$$\mathrm{Tr}\left(\widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta}\right) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{z}_i^\top\widehat{K}_{XX}^{-1}\right)\left(\frac{d\widehat{K}_{XX}}{d\theta}P^{-1}\mathbf{z}_i\right) .$$

[4]`https://pytorch.org/`

Given the above considerations, there are three important constraints on the preconditioner: it must be easy and efficient to i) solve $P^{-1}\mathbf{z}$, ii) compute the log-determinant $\log(\det P)$ exactly, iii) sample from $\mathcal{N}(\mathbf{0}, P)$. The preconditioner presented in the following section is a low-rank perturbation to the identity matrix, so the constraints are addressed by leveraging i) the Woodbury formula, ii) the matrix determinant lemma, iii) the reparametrization trick, see [10] for details.

3.2 Pivoted Cholesky as a Low-Rank Approximation

The proposed preconditionner in [10] is the matrix

$$\widehat{P}_k = L_k L_k^\top + \sigma^2 \mathbb{I} \tag{15}$$

with $L_k \in \mathbb{R}^{n\times k}$ the partial pivoted Cholesky factor [16]. The algorithm iteratively builds the matrix L_k up to the desired rank k, by adding a rank-one update matrix $\boldsymbol{\ell}_i\boldsymbol{\ell}_i^\top$ at each step i, so that $L_k = \sum_{i=1}^k \boldsymbol{\ell}_i\boldsymbol{\ell}_i^\top$. In order to obtain a good low-rank approximation $L_k L_k^\top \approx K_{XX}$, the heuristic selects at each step the largest pivot of the remaining Schur complement and perform pivoting.

The convergence of the algorithm is assessed by monitoring the trace of the error matrix $E_k := K_{XX} - L_k L_k^\top$, which is computed exactly in only $\mathcal{O}(n)$ time. This might serves as an early stopping criterion to select a good rank k.

We now present some theoretical results. We start showing an important property of the spectrum of the preconditioned matrix, which is not discussed in [10]. We then upper bound the condition number of the preconditioned matrix, following the derivation in [10] but extending results to arbitrary SPD matrices $\widehat{K}_{XX}$. We will note M_k the preconditioned matrix

$$M_k := \widehat{P}_k^{-1/2}\widehat{K}_{XX}\widehat{P}_k^{-1/2} , \tag{16}$$

and notice that [10] refers to $\widehat{P}_k^{-1}\widehat{K}_{XX}$ but those two matrices have the same spectrum and the same log-determinant, and their condition number is related as follows.

Proposition 3.1. *The condition numbers of the matrices M_k (Equation (16)) and $\widehat{P}_k^{-1}\widehat{K}_{XX}$ satisfy*

$$\kappa(M_k) \leq \kappa(\widehat{P}_k^{-1}\widehat{K}_{XX}) .$$

Proof. Note first the inequality relating singular values and eigenvalues, for a matrix $A \in \mathbb{R}^{n\times n}$, $|\lambda_{\max}(A)| \leq \sigma_{\max}(A)$ (see section 7.1.6 of [14]). This is shown by considering an eigen-pair $(\lambda, \mathbf{u})$ of A and the sub-multiplicative property of matrix norms:

$$|\lambda|\|\mathbf{u}\|_2 = \|\lambda\mathbf{u}\|_2 = \|A\mathbf{u}\| \leq \|A\|_2\|\mathbf{u}\|_2 = \sigma_{\max}(A)\|\mathbf{u}\|_2 .$$

Using this property and equality of spectra, the statement follows,

$$\begin{aligned}\kappa(M_k) = \lambda_{\max}(M_k)\lambda_{\max}(M_k^{-1}) &= \lambda_{\max}(\widehat{P}_k^{-1}\widehat{K}_{XX})\lambda_{\max}(\widehat{K}_{XX}^{-1}\widehat{P}_k)\\ &\leq \sigma_{\max}(\widehat{P}_k^{-1}\widehat{K}_{XX})\sigma_{\max}(\widehat{K}_{XX}^{-1}\widehat{P}_k) = \kappa(\widehat{K}_{XX}^{-1}\widehat{P}_k) .\end{aligned}$$

□

Proposition 3.2. *Eigenvalues of the preconditioned matrix M_k from Equation (16) are lower bounded by one.*

Proof. Since the decomposition is exact after n steps, $K_{XX} = L_n L_n^\top = \sum_{i=1}^n \boldsymbol{\ell}_i\boldsymbol{\ell}_i^\top \succcurlyeq L_k L_k^\top$, we have

$$\begin{aligned}\lambda_{\min}(\widehat{P}_k^{-1/2}\widehat{K}_{XX}\widehat{P}_k^{-1/2}) &= \min_{\mathbf{v}\neq\mathbf{0}} \frac{\mathbf{v}^\top\widehat{P}_k^{-1/2}\widehat{K}_{XX}\widehat{P}_k^{-1/2}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}\\ &= \min_{\mathbf{w}\neq\mathbf{0}} \frac{\mathbf{w}^\top\widehat{K}_{XX}\mathbf{w}}{\mathbf{w}^\top\widehat{P}_k\mathbf{w}}\\ &= \min_{\mathbf{w}\neq\mathbf{0}} \frac{\mathbf{w}^\top(\sum_{i=1}^n \boldsymbol{\ell}_i\boldsymbol{\ell}_i^\top + \sigma^2\mathbb{I})\mathbf{w}}{\mathbf{w}^\top(\sum_{i=1}^k \boldsymbol{\ell}_i\boldsymbol{\ell}_i^\top + \sigma^2\mathbb{I})\mathbf{w}}\\ &\geq 1 .\end{aligned}$$

□

Remark. *Proposition 3.2 has important consequences for log-determinant estimation. First, this implies* $\log M_k$ *is PSD. This shows that Theorem 2 of [10] is valid, since it is based on Theorem 4.1 of [27], which requires* log *to not cross zero inside the spectral interval of* M_k*. It also shows that Theorem 2.3 improves over [10] since we roughly need the number of probe vectors* N *to scale with* $\|\log M_k\|_F^2/\operatorname{Tr}(\log M_k)^2 \leq 1$.

Theorem 3.3 (Theorem 3.2 of [16]). *Let* $L_k \in \mathbb{R}^{n\times k}$ *be the pivoted Cholesky factor of* $K_{XX} \in \mathbb{R}^{n\times n}$ *after* k *steps. Denote* $\boldsymbol{\pi}^{(k)} \in \mathbb{R}^n$ *the vector of permutations of the algorithm after* k *steps and* $\Pi_k \in \mathbb{R}^{n\times n}$ *the identity matrix with rows permuted according to* $\boldsymbol{\pi}^{(k)}$*. Consider the following partitioning,*

$$\Pi_n K_{XX} \Pi_n^\top = \begin{bmatrix} K_{11} & K_{2,1}^\top \\ K_{2,1} & K_{22} \end{bmatrix}, \quad \Pi_k L_k = \begin{bmatrix} L_{11} \\ L_{12} \end{bmatrix} ,$$

with $K_{11}, L_{11} \in \mathbb{R}^{k\times k}$ *and define* $\Gamma_k := \|L_{11}^{-1}\|_2^2\,(L_{11})_{k,k}^2$*. Then the trace of the low-rank approximation error satisfies*

$$\operatorname{Tr}(K_{XX} - L_k L_k^\top) \leq n\Gamma_k \lambda_k(K_{XX}) , \tag{17}$$

with $\lambda_k(A)$ *the kth largest eigenvalue of* A*. Moreover, in the worst case,* $\Gamma_k = \mathcal{O}(4^k)$.

Proof. We generalize theorem 3.2 in [16], by rephrasing the bounds in terms of Γ_k instead of the worst case scenario, and by introducing pivoting.

Note that pivoted Cholesky starts with $\boldsymbol{\pi}^{(0)} = (1, 2, \ldots, n)$ and performs greedy updates such that $\boldsymbol{\pi}_{1:k}^{(k+1)} = \boldsymbol{\pi}_{1:k}^{(k)}$ for $k \geq 1$. With the above partitioning, L_{11} is lower-triangular and is the Cholesky factor of K_{11}. Also remark that $1/\|L_{11}^{-1}\|_2^2 = \lambda_k(K_{11})$, whereas $(L_{11})_{k,k}^2$ is simply the kth pivot element selected at the kth step of pivoted Cholesky. By construction of the algorithm, the trace error at step k is bounded by $n - m$ times the pivot element, then using the Courant-Fisher theorem we upper bound eigenvalues of the submatrix K_{11},

$$\operatorname{Tr}(E_k) \leq n(L_{11})_{k,k}^2 = n\Gamma_k\lambda_k(K_{11}) \leq n\Gamma_k\lambda_k(\Pi_n K_{XX}\Pi_n^\top) = n\Gamma_k\lambda_k(K_{XX}) .$$

The worst case $\Gamma_k = \mathcal{O}(4^k)$ is achieved when L_{11} has its subdiagonal full of -1 (see remarks of theorem 6.1 in [17]). □

Lemma 3.4 (Lemma 1 of [10]). *Let* M_k*,* $k < n$ *be the preconditioned matrix of Equation* (16)*, then*

$$\kappa\left(M_k\right) \leq \left(1 + c\operatorname{Tr}(E_k)\right)^2 ,$$

with $E_k = K_{XX} - L_k L_k^\top$ *the error matrix and* $c = 1/\sigma^2$.

Proof. The beginning of proof is exactly the same as lemma 1 of [10]. We simply do not include bounds on the trace yet and give an expression for the constant c. Using proposition 3.1 and with some substitutions $K_{XX} = E_k + L_k L_k^\top$, $L_k L_k^\top = K_{XX} - E_k$, we obtain after a few manipulations

$$\kappa(M_k) \leq \kappa(\widehat{P}_k^{-1}\widehat{K}_{XX}) := \|\widehat{P}_k^{-1}\widehat{K}_{XX}\|_2 \|\widehat{K}_{XX}^{-1}\widehat{P}_k\|_2 = \|\mathbb{I} + \widehat{P}_k^{-1}E_k\|_2\|\mathbb{I} - \widehat{K}_{XX}^{-1}E_k\|_2 .$$

Applying the Cauchy-Schwarz and triangle inequalities,

$$\kappa(M_k) \leq (1 + \|\widehat{P}_k^{-1}\|_2\|E_k\|_2)(1 + \|\widehat{K}_{XX}^{-1}\|_2\|E_k\|_2) \leq (1 + c\operatorname{Tr}(E_k))^2 ,$$

where we use that E_k is PSD to upper-bound the spectral norm by the trace, and since $L_k L_k^\top$ is rank-deficient for $k < n$, it follows

$$\begin{aligned} c :=& \max\left\{\|\widehat{P}_k^{-1}\|_2, \|\widehat{K}_{XX}^{-1}\|_2\right\} \\ =& \max\left\{1/\lambda_{\min}(\widehat{P}_k), 1/\lambda_{\min}(\widehat{K}_{XX})\right\} \\ =& \max\left\{\frac{1}{\sigma^2}, \frac{1}{\sigma^2 + \lambda_{\min}(K_{XX})}\right\} \\ =& \frac{1}{\sigma^2} . \end{aligned}$$

□

Theorem 3.5. *With notation of Theorem 2.4, Theorem 3.3 and Lemma 3.4, we can upper bound the condition number as*

$$\kappa\left(M_k\right) \leq \left(1+\mathcal{O}(n\Gamma_k\lambda_k)\right)^2 ,$$

and pCG after m steps converges as

$$\|\mathbf{u}-\mathbf{u}_m\|_{\widehat{K}_{XX}} \leq 2\|\mathbf{u}-\mathbf{u}_0\|_{\widehat{K}_{XX}} \left(\frac{1}{1+\mathcal{O}((n\lambda_k\Gamma_k)^{-1})}\right)^m$$

Proof. We bound the condition number by directly plugging Theorem 3.3 into bounds of Lemma 3.4 as done in proof of Lemma 1 of [10]. We drop the constant c since the noise ϵ_i of Equation (1) is independent of the data points $\mathbf{x}_i$. We then plug the condition number in CG convergence bounds, as done in proof of Lemma 1 of [10],

$$\frac{1}{2}\frac{\|\mathbf{u}-\mathbf{u}_m\|_{\widehat{K}_{XX}}}{\|\mathbf{u}-\mathbf{u}_0\|_{\widehat{K}_{XX}}} \leq \left(\frac{\sqrt{\kappa(M_k)}-1}{\sqrt{\kappa(M_k)}+1}\right)^m \leq \left(\frac{1+\mathcal{O}(n\Gamma_k\lambda_k)-1}{1+\mathcal{O}(n\Gamma_k\lambda_k)+1}\right)^m \leq \left(\frac{1}{1+\mathcal{O}((n\lambda_k\Gamma_k)^{-1})}\right)^m .$$

□

Remark. *As mentioned, those results are essentially a reformulation of the cited papers for an arbitrary kernel matrix. Garnder et al. [10] only covers the case $\lambda_k = \mathcal{O}(4^{-k})$, which as they show holds with 1D uniform data points and the squared exponential kernel. In this case (since $\Gamma_k = \mathcal{O}(4^k)$), we are guaranteed that M_k converges exponentially fast to the identity with respect to k. In the next section, we extend results of [10] for other kernels and distributions of the data, but we only achieve a sub-exponential spectral decay. Truly exploiting Theorem 3.5 would require to also characterize the factor Γ_k.*

4 Spectrum of Kernel Matrices

In this section, we aim to characterize the spectral decay of kernel matrices, focusing on the squared exponential (SE) and Matern kernels. Both are radial kernels, i.e. $k(\mathbf{x}_i, \mathbf{x}_j)$ only depends on $r_{ij} := \|\mathbf{x}_i - \mathbf{x}_j\|_2$. They are defined as $k_{\mathrm{SE}}(r;\, \ell) = e^{-r^2/\ell}$ and

$$k_{\mathrm{Matern}}(r;\, \ell, \nu) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{\ell}r\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{\ell}r\right) ,$$

with Γ the gamma function and K_ν the modified Bessel function of the second kind. Note that the SE kernel is a special case of the Matern kernel in the limit $\nu \to \infty$ (see e.g. [6], [11], [20]).

Gardner et al. [10] treat the squared exponential with 1D data in the interval $[0,1]$, casting the spectral analysis as a polynomial approximation problem (lemma 4 of [10]). They use Chebyshev expansions (lemma 5 of [10]) to eventually show a super-exponential spectral decay.

We wish to develop a set of tools to extend those results for more general cases (higher dimensions and other kernels). We start with Chebyshev expansions for the SE kernel, including the missing proof of lemma 5 of [10] for completeness. In passing, we mention an alternative method that is useful in the case the error of the truncated Mercer expansion would be known. We then cast the spectral analysis of the kernel matrix as the spectral analysis of an integral operator associated to the kernel function, exploiting the rich literature of such operators.

4.0.1 Chebyshev approximation of Kernel Functions

We briefly recall the Chebyshev polynomials of the first kind, see e.g. chapter 3 of [12] for details. Those polynomials are defined as $T_k(x) := \cos(k \arccos(x)), x \in [-1,1]$. Using trigonometric identities, the following recurrence relation can be obtained:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - 2T_{k-2}(x), \quad k \geq 2,\, x \in [-1, 1] \;.$$

Chebyshev polynomials are near-minimax approximations [9], meaning they are almost optimal for minimizing the uniform norm of the polynomial approximation error. One first considers the Chebyshev expansion of some continuous function $f : [-1; 1] \to \mathbb{R}$:

$$f(x) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} a_k T_k(x) \;, \tag{18}$$

and sets the degree d Chebyshev approximation as the first d terms of the above sum. The error is upper bounded by:

$$\|f - p_l\| = \left\|\textstyle\sum_{i=l+1}^{\infty} a_k T_k\right\| \leq \sum_{i=l+1}^{\infty} |a_k| \;,$$

since $|T_k(x)| \leq 1$. Chebyshev polynomials are orthogonal with respect to the weight function $w(x) = 1/\sqrt{1-x^2}$. Knowing that $\langle T_k, T_k\rangle_w = \pi/2$ for $k > 0$, the analytical expression of a_k coefficients is derived from (18):

$$a_k = \frac{2}{\pi}\int_{-1}^{1} f(x) T_k(x) \frac{1}{\sqrt{1-x^2}} dx \;, \quad k \geq 1 \;. \tag{19}$$

With a change of variable $x = \cos\theta$, this can be rewritten as

$$a_k = \frac{2}{\pi}\int_{0}^{\pi} f(\cos\theta)\cos(k\theta) d\theta \;, \quad k \geq 1. \tag{20}$$

Lemma 4.1 (Lemma 5 of [10]). *Let p_{2d} be the Chebyshev expansion of the squared exponential of degree $2d$, then the error is*

$$|e^{-\gamma x^2} - p_{2d}(x)| \leq 2e^{-\gamma/4} I_{d+1}(\gamma/4) \;, \quad |x| \leq 1 \;,$$

with I_d the modified Bessel function of the first kind of order d.

Proof. The proof of lemma 5 of [10] simply states the error of the Chebyshev approximation in terms of an infinite sum of Bessel function, without proof. We fill in the gap by proving this step.

It is easy to show by induction that $T_k(-x) = (-1)^k T_k(x)$, i.e. T_k is odd for k odd. Since $e^{-\gamma x^2}$ is even, we see from (19) that odd coefficients a_k vanish. We use equation (20) to compute the coefficients, with the trigonometric identity $\cos^2\theta = (1 + \cos(2\theta))/2$,

$$a_k = \frac{2}{\pi}\int_0^{\pi} e^{-\gamma\cos^2\theta}\cos(\theta k) d\theta = \frac{2}{\pi}e^{-\gamma/2}\int_0^{\pi} e^{-\gamma\cos(2\theta)/2} cos(\theta k) d\theta \;.$$

We recall the series expansion and integral form of the modified Bessel function of the first kind for integer order n (see e.g. formulas 9.6.10 and 9.6.19 of [1], respectively),

$$I_n(x) = (\tfrac{1}{2}x)^n \sum_{m=0}^{\infty} \frac{(\frac{1}{2}x)^{2m}}{m!(n+m)!} = \frac{1}{\pi}\int_0^{\pi}\cos(n\theta)e^{x\cos\theta} d\theta \;.$$

Computing only even coefficients a_{2k}, we retrieve the Bessel functions with a change of variable $\alpha = 2\theta$,

$$a_{2k} = \frac{1}{\pi}e^{-\gamma/2}\int_0^{2\pi} e^{-\gamma\cos(\alpha)/2}\cos(k\alpha) d\alpha = \frac{2}{\pi}e^{-\gamma/2}\int_0^{\pi} e^{-\gamma\cos(\alpha)/2}\cos(k\alpha) d\alpha = 2e^{-\gamma/2} I_k(-\gamma/2) \;,$$

where the second inequality holds due to symmetry of the cosine. From the series expansion, we see that $|I_n(-x)| = I_n(x)$, so the error is

$$|e^{-\gamma x^2} - p_{2d}| \leq 2e^{-\gamma/2}\sum_{l=d+1}^{\infty} I_l(\gamma/2) \;.$$

Using lemma 6 of [10] to bound the sum of Bessel functions concludes the proof. □

4.0.2 Mercer expansion

An alternative approach to Chebyshev approximation relies on eigen-decomposition of the kernel function. If $k : D \times D \to \mathbb{R}$ with $D \subset \mathbb{R}^d$ compact, then by the Mercer theorem (see e.g. theorem 4.2 of [20]),

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{\infty} \lambda_p \phi_p(\mathbf{x}_i)\phi_p(\mathbf{x}_j) \ ,$$

with $\{\lambda_p\}_{p=1}^{\infty}$ the absolutely summable eigenvalues and ϕ_p the corresponding eigenfunctions. Then we define the rank m matrix $\tilde{K}_m = \Phi^\top \mathrm{diag}(\lambda_1, \ldots, \lambda_m)\Phi$ corresponding to the truncated Mercer expansion with the m first terms, with $\Phi_{ij} = \phi_i(\mathbf{x}_j)$.

Proposition 4.2 (Spectral norm of matrix with bounded entries)**.** *Let $A \in \mathbb{R}^{n\times n}$ be any square matrix with bounded entries $|a_{ij}| \leq \epsilon$, then $\|A\|_2 \leq n\epsilon$.*

Proof. This proposition is inspired by Lemma 4 of [10], which mentions this property without proof. We apply the Gershorogin theorem on $A^\top A$. By assumption of bounded entries,

$$(A^\top A)_{ij} = \textstyle\sum_{k=1}^n a_{ki}a_{kj} \leq n\epsilon^2 \ ,$$

and the Gershogorin theorem states, for some row r,

$$\lambda_{\max}(A^\top A) \leq \textstyle\sum_{k=1}^n |(A^\top A)_{rk}| \ ,$$

so that $\|A\|_2^2 = \lambda_{\max}(A^\top A) \leq n^2\epsilon^2$. □

Proposition 4.3. *Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in D$ be d-dimensional data in $D \subset \mathbb{R}^d$ compact. If k_m is the truncated Mercer expansion of $k : D \to D$ with m terms and satisfies*

$$\|k - k_m\|_{L_\infty(D^2)} \leq \epsilon_m \ ,$$

then $\lambda_{m+1}(K_{XX}) \leq n\epsilon_m$.

Proof. The proof is similar to the one of lemma 4 in [10]. Consider the matrix $\tilde{K}_m$ corresponding to the truncated Mercer expansion with m terms and the error matrix $E_m = K_{XX} - \tilde{K}_m$, by assumption $\|E_m\|_\infty = \epsilon_m$. All $K_{XX}, \tilde{K}_m, E_m$ being PSD, by the min-max principle,

$$\lambda_k(K_{XX}) = \max_{\substack{\mathcal{U}\subset\mathbb{R}^n \\ \dim\mathcal{U}=k}} \min_{\substack{\mathbf{v}\in\mathcal{U} \\ \|\mathbf{v}\|_2=1}} \mathbf{v}^\top(\tilde{K}_m + E_m)\mathbf{v} \leq \max_{\substack{\mathcal{U}\subset\mathbb{R}^n \\ \dim\mathcal{U}=k}} \min_{\substack{\mathbf{v}\in\mathcal{U} \\ \|\mathbf{v}\|_2=1}} \mathbf{v}^\top\tilde{K}_m\mathbf{v} + \|E_m\|_2 = \lambda_k(\tilde{K}_m) + \|E_m\|_2 \ .$$

Since $\tilde{K}_m$ has rank at most m, we have $\lambda_{m+1}(\tilde{K}_m) = 0$ and hence $\lambda_{m+1}(K_{XX}) \leq \|E_m\|_2$. Using the proposition 4.2 concludes the proof. □

4.0.3 Eigenanalysis of Hilbert-Schmidt Operator

An alternative analysis is based on characterization of the eigenvalues of the integral operator associated with the kernel function [4, 19, 2]. That is, we define the Hilbert-Schmidt operator $T_k : L_2(D) \to L_2(D)$ with $D \subset \mathbb{R}^d$ compact,

$$(T_k f)(\mathbf{y}) = \int_D k(\mathbf{x}, \mathbf{y})f(\mathbf{x})d\mathbf{x} \ , \tag{21}$$

and are interested in the eigen-problem

$$T_k\phi_i = \lambda_i\phi_i \ .$$

We can relate the spectrum of the kernel matrix with the spectrum of the operator T_k with the Nyström method. We approximate the integral (21) with the quadrature rule

$$\lambda_i \phi_i(\mathbf{x}_j) \approx \frac{1}{n} \sum_{k=1}^{n} k(\mathbf{x}_k, \mathbf{x}_j)\phi_i(\mathbf{x}_k) \ , \quad j = 1, \dots, n \ , \tag{22}$$

so that we have a matrix eigenproblem, with $\tilde{\lambda}_i$ eigenvalues of the kernel matrix and $\mathbf{\Phi}_i = (\phi_i(\mathbf{x}_1), \dots, \phi_i(\mathbf{x}_n))$ its corresponding eigenvector:

$$K_{XX}\mathbf{\Phi}_i = \lambda_i(K_{XX})\mathbf{\Phi}_i \ .$$

This method always underestimate the eigenvalues of the operator T_k (see proof of theorem 4 in [4] and [3]), i.e. $\lambda_i(K_{XX}) \leq n\lambda_i(T_k)$. The goal is now to characterize the decay of the eigenvalues of this operator. The following theorem states results for the squared exponential kernel using the above method:

Theorem 4.4 (Theorem 4 of [4])**.** *Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in D \subset \mathbb{R}^d$ be the d-dimensional data in D compact. Then the eigenvalues of the kernel matrix K_{XX} for the squared exponential kernel function $k : D \to D$, $k(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|_2^2}$ satisfies, for some $C > 0$,*

$$\lambda_m(K_{XX}) \leq Cn\frac{\gamma^{m^{1/d}}}{\Gamma(m^{1/d}/2)}$$

For the one-dimensional case $d = 1$, we retrieve the super-exponential spectral decay of lemma 4.1. In a higher-dimensional case $d > 1$, the decay is sub-exponential because the dth square root accounts for possible eigenvalue multiplicity (see remark 2.20 of [23]). For the Matern kernels, we can characterize eigenvalues for the 1D uniform case, with a bound that explicitly depends on the smoothness parameter ν.

Theorem 4.5. *Let $x_1, \dots, x_n \in \mathcal{U}([0, 1])$ be one-dimensional data. Then the eigenvalues of the kernel matrix K_{XX} for the Matern kernel with smoothness parameter ν satisfiy, for some $C > 0$,*

$$\lambda_m(K_{XX}) \leq Cnm^{-2\lceil\nu\rceil} \ .$$

Proof. The stochastic process f is $\lceil\nu\rceil - 1$ times differentiable in the mean-square sense (see [20] section 4.2.1). In the 1D case with uniformly distributed data, if a stochastic process is r times MS-differentiable, eigenvalues of T_k decay algebraically as m^{-2r-2} (see section 4.3 of [20]). Upper bounding eigenvalues of the kernel matrix by those of T_k as explained before gives the desired statement. □

For non-compact domain D, the operator T_k can be generalized by assuming a probability density function $\mathbf{x}_1, \dots, \mathbf{x}_n \sim p$ and introducing the weight function $p(\mathbf{x})$ in (21). The approximation (22) is then interpreted as an estimator of the expectation $\mathbb{E}_{\mathbf{x}\sim p}[k(\mathbf{x}, \cdot)f(\mathbf{x})]$. Some results then relate the eigenvalue decay to the decay of the Fourier transform of k and to the decay of p (see appendix C.2 of [2] for the 1D case with Gaussian distribution).

In order to extend results for Matern kernels in the multidimensional setting on a compact domain, possible direction could be to explore whether the Matern kernel is Sobolev smooth and then use results of eigenvalue decay for such smoothness conditions (see proposition 2.21 of [23] or proposition 3.2 of [25]). If this is the case, a decay of $m^{-\alpha/d}$ could be obtained, for some $\alpha > 0$.

To the best of my knowledge, there is no result for a non-compact domain in $d > 1$ dimensions.

5 Numerical Experiments

In the following numerical experiments, we will generate synthetic data points $\mathbf{x}_i \in \mathbb{R}^d$ with independent components in dimension $d = 1, 2, 3, 4$. This setting is far from real-world applications in which $d > 10$ may be common, and such cases are presented in experiments of Gardner et al. [10]. However, our experiments cannot be compared with those of Gardner et al. [10] as i) they do not provide a breakdown of the computations, ii) their code provides no obvious entry point to do so (e.g. compute the logdet), iii) their implementation heavily relies on techniques[5] not even mentionned in the paper to increase efficiency and accuracy.

[5]See an example list on `https://docs.gpytorch.ai/en/stable/examples/02_Scalable_Exact_GPs/index.html`

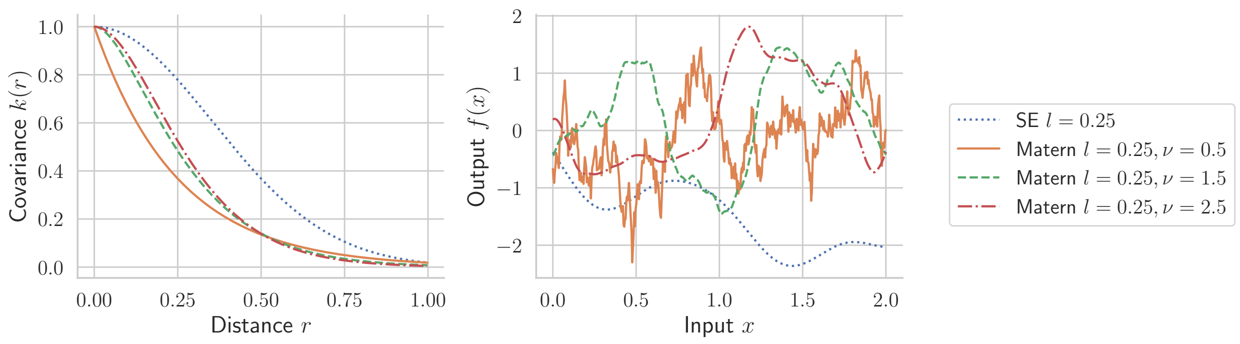


Figure 1: Overview of different kernel functions. SE is squared exponential, l is lengthscale, ν is the smoothness parameter. **Left**: covariance functions as function of distance. **Right**: one prior function $f \sim \mathcal{N}(\mathbf{0}, K_{XX})$ sampled for each kernel.

The experiments are organized as follows. We first show the spectra of different kernel matrices, and the efficiency of pivoted Cholesky on those matrices. We then analyze errors of Lanczos quadrature and stochastic Lanczos quadrature for logdet estimation. Eventually, we show the errors of the likelihood and its gradient in function of the preconditioner rank.

5.1 Overview of Covariance Functions and Kernel Matrices

We start by showing the shape of the different kernels functions and random functions drawn from the prior induced by the choice of the kernel in Figure 1. For Matern kernels, one sees how the smoothness of the function increases with increasing ν parameter. For the extreme cases $\nu = 1/2$ and $\nu \to \infty$ (which corresponds to SE kernel).

In Figure 2, we show the eigenvalues of the kernel matrices K_{XX} for the same kernel functions, using normally- and uniformly-distributed data in different dimensions. For the SE kernel in 1D, eigenvalues exhibit fast exponential decay, as the slope is linear on a semilog scale. Increasing the dimension greatly reduces the rate of decay. For the 1D Matern uniform case, the linear part on a log-log scale (between index 10 and 100) indeed has slope $-2\lceil \nu \rceil = -4$ as stated in theorem 4.5 with corresponding constant $C = 233$.

5.2 Pivoted Cholesky on Kernel Matrices

We use 1000 data points $\mathbf{x}_i \sim \mathcal{D}$ for the uniform and Gaussian distribution in several dimensions with the SE and Matern kernels. We run up to $k_{\max} = 300$ steps of pivoted Cholesky, early stopping whenever $\mathrm{Tr}(E_k)$ drops below the tolerance 10^{-10}.

The growth factor Γ_k of Theorem 3.3 is shown in Figure 3. Recall that the theoretical worst case is $\Gamma_k = \mathcal{O}(4^k)$. For the kernel matrices we consider here, we have linear growth at worst. Comparing the balance between spectral decay and the growth of the gamma factor, we show in Figure 4 the factor $\Gamma_k \lambda_k(K_{XX})$ appearing in the upper bound of $\mathrm{Tr}(E_k)$ in Theorem 3.3. For the cases with slow spectral decay (see Figure 2), the upper bound might not even decay with the rank k, giving no guarantees on the efficiency of pivoted Cholesky as a preconditioner. Eventually, the condition number of the kernel matrix preconditioned with pivoted Cholesky in Figure 5. For Gaussian data in dimension $d > 1$, the preconditioner has little effect, but the matrices are well conditioned in the first place.

5.3 Lanczos quadrature and Log Determinant Estimation

We compare the bounds of Theorem 2.3 with empirical results. We use custom matrices to have control on the spectra. Recall for a fixed relative error ϵ, the estimated number of probe vectors N is mainly driven by the ratio $r = \|\log A\|_F^2 / \mathrm{Tr}(\log A)^2$. In Figure 6, we compare log-det estimation for a matrix with ratio $r \ll 1$ (left panel) and with $r \lesssim 1$ (right panel). The case $r \ll 1$ requires very few probe vectors and iterations,

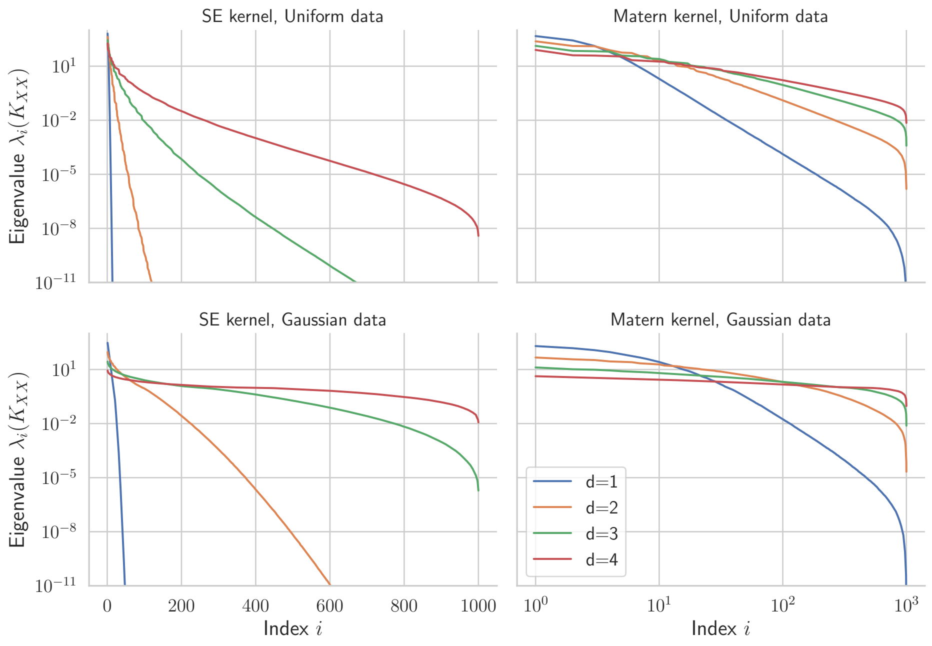


Figure 2: Eigenvalues of the kernel matrix on 1000 data points $\mathbf{x}_i \sim \mathcal{D}$ for different distribution $\mathcal{D}$ in dimension $d = 1, 2, 3, 4$. Values shown are averaged over 5 runs. **Upper row**: Uniform data $\mathcal{D} = \mathcal{U}([0, 1]^d)$, **bottom row**: Gaussian data $\mathcal{D} = \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$. **Left column**: squared exponential kernel with lengthscale $l = 0.25$, **right column**: Matern kernel with lengthscale $l = 0.25$ and smoothness $\nu = 1.5$.

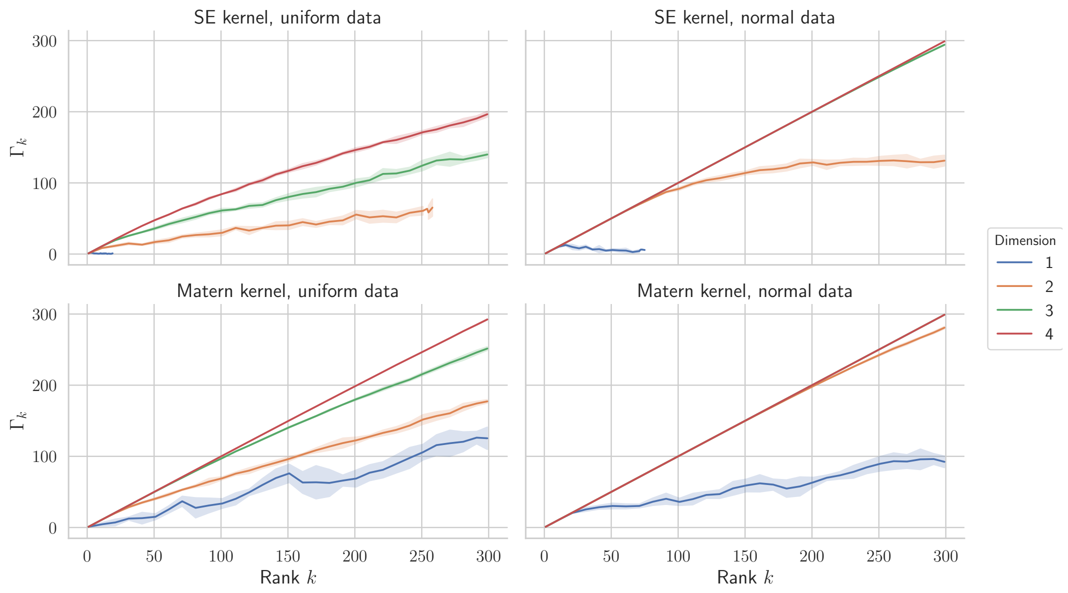


Figure 3: Factor Γ_k of Theorem 3.3 for pivoted Cholesky run on K_{XX} with 1000 data points $\mathbf{x}_i \sim \mathcal{D}$. Values are averaged over five runs, shaded areas represent $\pm$ standard deviation. **Columns**: uniform (left) and Gaussian (right) data. **Rows**: squared exponential (top) and Matern (bottom) kernels.

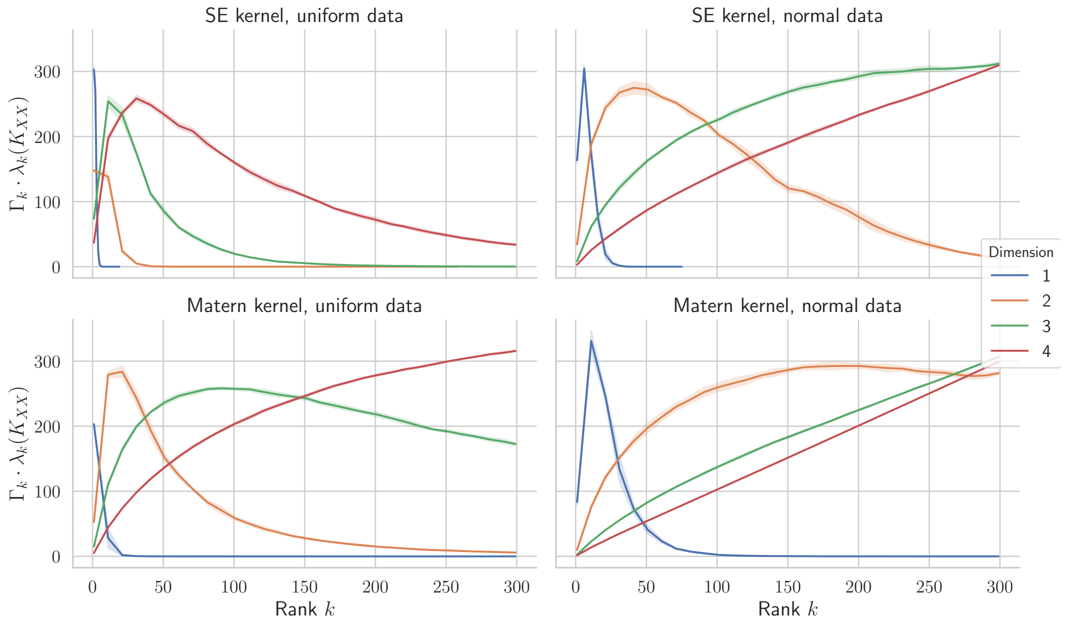


Figure 4: Upperbound $\Gamma_k \lambda_k(K_{XX})$ of Theorem 3.3 for pivoted Cholesky run on K_{XX} with 1000 data points $\mathbf{x}_i \sim \mathcal{D}$. Values are averaged over five runs, shaded areas represent $\pm$ standard deviation. **Columns**: uniform (left) and Gaussian (right) data. **Rows**: squared exponential (top) and Matern (bottom) kernels.

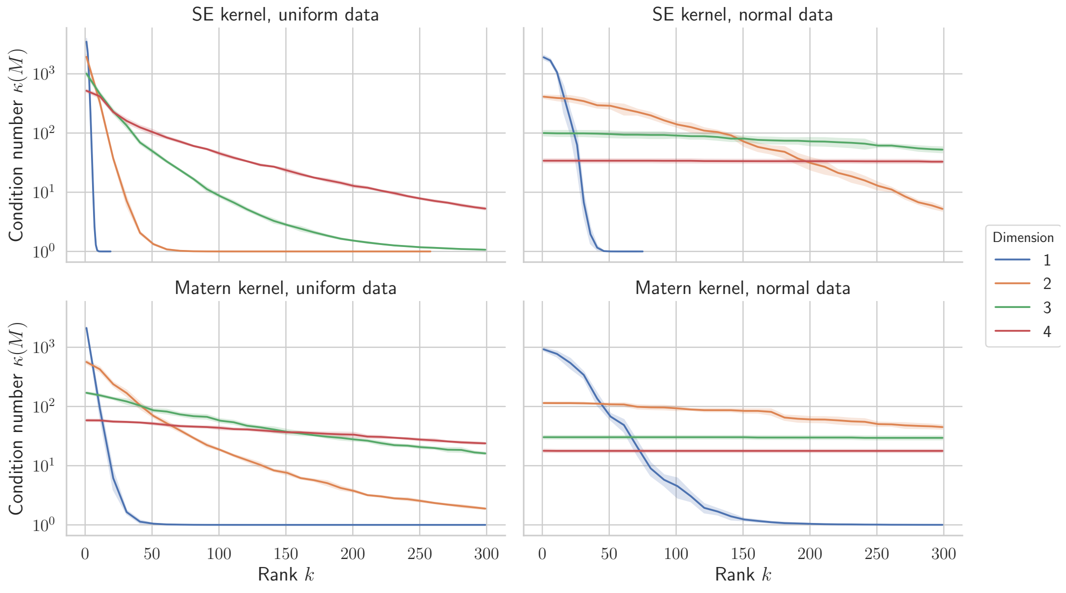


Figure 5: Condition number of the preconditioned matrix $M := \widehat{P}_k^{-1/2} \widehat{K}_{XX} \widehat{P}_k^{-1/2}$ with 1000 data points $\mathbf{x}_i \sim \mathcal{D}$. Values are averaged over five runs, shaded areas represent $\pm$ standard deviation. **Columns**: uniform (left) and Gaussian (right) data. **Rows**: squared exponential (top) and Matern (bottom) kernels.

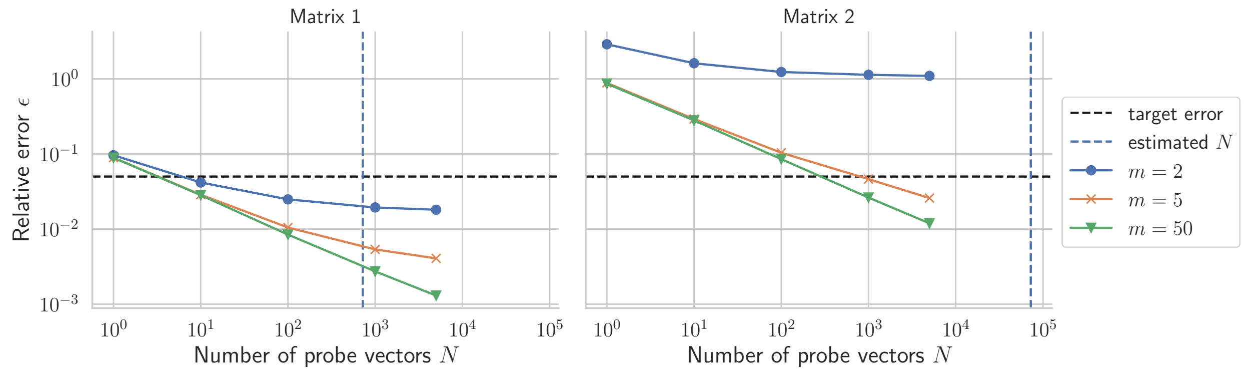


Figure 6: Log determinant estimation with stochastic Lanczos quadrature where partial tridiagonal matrices were computed with mBCG algorithm, no preconditioning used. The matrices are $A = Q\Lambda Q^\top \in \mathbb{R}^{n\times n}$ with $n = 1000$, Q orthogonal and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. The 95th error quantiles are plotted. **Left**: $\lambda_i = i$, $1 \le i \le n$, the estimated m value is 51. **Right**: $\lambda_i = 10i$, $1 \le i \le 10$, $\lambda_i = 1$, $10 < i \le n$, so that $\mathrm{rank}(\log K_{XX}) = 10$. The estimated m value is 132.

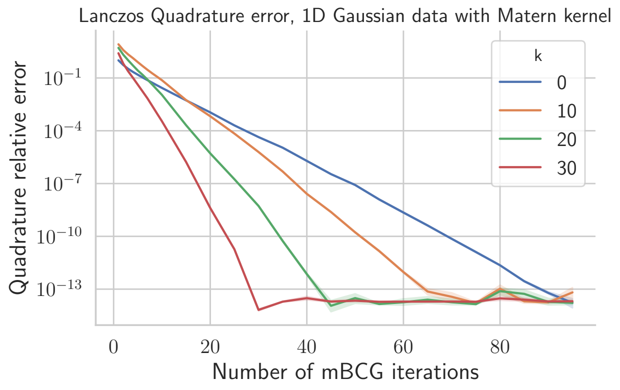


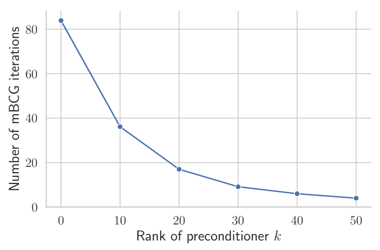
Figure 7: Error of Lanczos quadrature for the estimate $\|\mathbf{z}\|_2^2 \mathbf{e}_1^\top \log(T_m)\mathbf{e}_1 \approx \mathbf{z}^\top \log(M)\mathbf{e}_1$, with T_m the tridiagonalization of the preconditioned matrix $M = \widehat{P}_k^{-1/2}\widehat{K}_{XX}\widehat{P}_k^{-1/2}$ for $k > 0$ and $M = K_{XX}$ for $k = 0$. We used 1D Gaussian data with a Matern kernel ($\sigma^2 = 0.1$, $\ell = 0.1$).

as the variance of the estimator and the error of Lanczos are small compared to the trace. The case $r \lesssim 1$ compares unfavorably to using n probe vectors with the canonical basis.

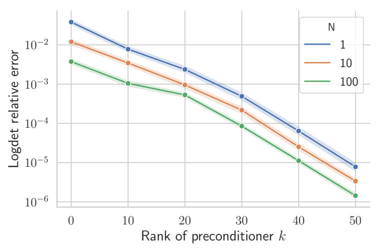
We show the error of Lanczos quadrature with preconditioning for an example kernel matrix in Figure 7. Each curve represents a different rank k of the preconditioner, and hence represent estimation of quadratures with different matrices. Comparing the slopes shows preconditioning allows to decrease the number of iterations.

5.4 Computation of Likelihood

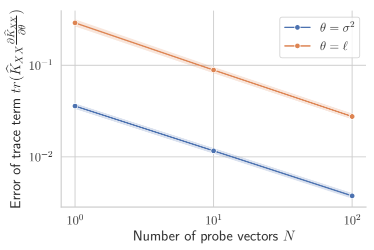
We test computation of likelihood on 2D uniform data with SE kernel. We run as many steps as required for mBCG to reach tolerance 10^{-10}, as shown in Figure 8(a). For a fixed preconditioner rank k, the logdet error achieves a Monte Carlo rate $\mathcal{O}(N^{-1/2})$ in Figure 8(b), whereas increasing k makes the preconditioner capture most of the logdet (see Equation 14). The error on the trace term is independent of k and achieves a Monte Carlo rate as well, see Figure 8(c). The final errors on the likelihood and the gradient (Equations 4, 5) are shown in Figures 8(d), 8(e).



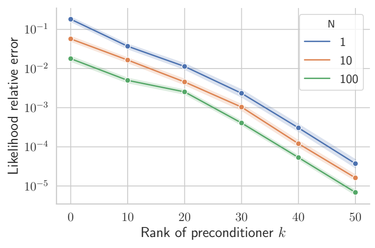
(a)



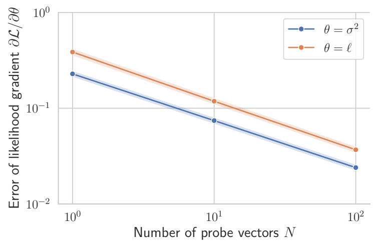
(b)



(c)

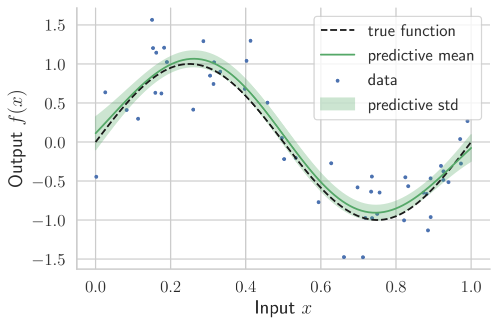


(d)

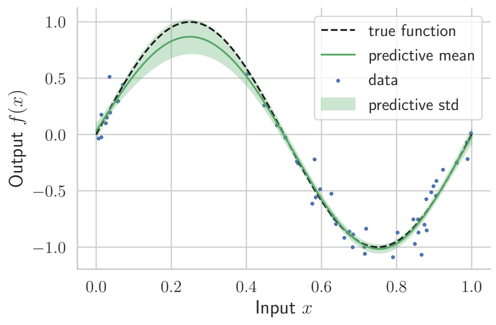


(e)

Figure 8: Likelihood computation for 2D uniform data with SE kernel ($\sigma^2 = 0.1, \ell = 0.2$).



(a)



(b)

Figure 9: Example of Gaussian process regression with SE kernel on the sinus function. **(a)** Uniform data in $[0, 1]$. **(b)** Uniform data in $[0, 0.1] \cup [0.4, 1]$.

5.5 Inference on Toy Example

As a last example, we show how GP models can be used to approximate the sine function with a simple dataset. We generate $x_1, \ldots, x_{50} \sim \mathcal{U}([0, 1])$ and $y_i = \sin(2\pi x_i) + \epsilon_i$, $\epsilon \sim \mathcal{N}(0, 0.1)$. We fit a squared exponential kernel, starting with initial guesses $\hat{\sigma}_0^2 = 0.01, \hat{\ell}_0 = 0.5$. Estimated values after optimization are $\sigma^2 = 0.123, \hat{\ell} = 0.156$. The result is shown in Figure 9(a). We show how confidence intervals widen at regions with few data in Figure 9(b), where no data is generated between the interval $[0.1, 0.4]$.

Conclusion

Combining stochastic estimates with conjugate gradients and Lanczos algorithms was shown to provide accurate estimates of the likelihood and its gradient as required to train a Gaussian process. In particular, one can avoid the Cholesky decomposition and its cubic-time complexity. An attractive advantage of this GPyTorch framework is its ease of implementation (e.g., adding new kernels) and modularity (e.g., combine Gaussian processes with deep learning models). A significant limitation is a-priori selection of mBCG parameters (number of steps, number of probe vectors, tolerance) and the rank of the specialized preconditioner, so that one can truly use this inference engine as a blackbox. Further investigation may analyze the speedup provided by other techniques implemented in the GPyTorch library that are not mentioned in the paper.

References

[1] Milton Abramowitz, Irene A. Stegun, and United States. National Bureau of Standards. *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* In collab. with Internet Archive. New York : Wiley, 1972. 1078 pp. ISBN: 978-0-486-61272-0 978-0-471-80007-1. URL: `http://archive.org/details/handbookofmathe000abra`.

[2] Francis R. Bach and Michael I. Jordan. "Kernel Independent Component Analysis". In: *Journal of Machine Learning Research* 3 (Jul 2002), pp. 1–48. ISSN: ISSN 1533-7928. URL: `https://www.jmlr.org/papers/v3/bach02a`.

[3] Christopher T. H. Baker. *The numerical treatment of integral equations.* Monographs on numerical analysis. Oxford: Clarendon Press, 1977. xiv+1034. ISBN: 978-0-19-853406-8.

[4] Anjishnu Banerjee, Joshua Vogelstein, and David Dunson. *Parallel inversion of huge covariance matrices.* Dec. 6, 2013.

[5] Mikhail Belyaev, Evgeny Burnaev, and Yermek Kapushev. *Exact Inference for Gaussian Process Regression in case of Big Data with the Cartesian Product Structure.* Number: arXiv:1403.6573. July 3, 2014. arXiv: `1403.6573[math,stat]`. URL: `http://arxiv.org/abs/1403.6573`.

[6] Viacheslav Borovitskiy et al. "Matérn Gaussian processes on Riemannian manifolds". In: (), p. 12.

[7] Alice Cortinovis and Daniel Kressner. "On randomized trace estimates for indefinite matrices with an application to determinants". In: *arXiv:2005.10009 [cs, math]* (May 25, 2021). arXiv: `2005.10009`. URL: `http://arxiv.org/abs/2005.10009`.

[8] Volker L. Deringer et al. "Gaussian Process Regression for Materials and Molecules". In: *Chemical Reviews* 121.16 (Aug. 25, 2021). Publisher: American Chemical Society, pp. 10073–10141. ISSN: 0009-2665. DOI: `10.1021/acs.chemrev.1c00022`. URL: `https://doi.org/10.1021/acs.chemrev.1c00022`.

[9] D. Elliott et al. "Error of truncated Chebyshev series and other near minimax polynomial approximations". In: *Journal of Approximation Theory* 50.1 (May 1987), pp. 49–57. ISSN: 00219045. DOI: `10.1016/0021-9045(87)90065-7`. URL: `https://linkinghub.elsevier.com/retrieve/pii/0021904587900657`.

[10] Jacob R. Gardner et al. "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *arXiv:1809.11165 [cs, stat]* (June 29, 2021). arXiv: `1809.11165`. URL: `http://arxiv.org/abs/1809.11165`.

[11] Marc G. Genton. "Classes of kernels for machine learning: a statistics perspective". In: *The Journal of Machine Learning Research* 2 (Mar. 1, 2002), pp. 299–312. ISSN: 1532-4435.

[12] Amparo Gil, Javier Segura, and Nico Temme. *Numerical Methods for Special Functions.* Jan. 1, 2007. ISBN: 978-0-89871-634-4. DOI: `10.1137/1.9780898717822`.

[13] Gene H. Golub. *Matrices, moments and quadrature with applications.* In collab. with Gene Howard Golub et al. Princeton series in applied mathematics. Princeton, N.J: Princeton University Press, 2010. ISBN: 978-0-691-14341-5.

[14] Gene H. Golub. *Matrix computations.* In collab. with Gene Howard Golub, Charles F. van Loan, and Charles F. Van Loan. 4th [rev.] ed. Johns Hopkins studies in the mathematical sciences. Baltimore: Johns Hopkins University Press, 2013. 756 pp. ISBN: 978-1-4214-0794-4.

[15] Nico S. Gorbach et al. "Model Selection for Gaussian Process Regression". In: *Pattern Recognition.* Ed. by Volker Roth and Thomas Vetter. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 306–318. ISBN: 978-3-319-66709-6. DOI: `10.1007/978-3-319-66709-6_25`.

[16] Helmut Harbrecht, Michael Peters, and Reinhold Schneider. "On the low-rank approximation by the pivoted Cholesky decomposition". In: *Applied Numerical Mathematics.* Third Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2010) 62.4 (Apr. 1, 2012), pp. 428–440. ISSN: 0168-9274. DOI: `10.1016/j.apnum.2011.10.001`. URL: `https://www.sciencedirect.com/science/article/pii/S0168927411001814`.

[17] Nicholas J. Higham. "A Survey of Condition Number Estimation for Triangular Matrices". In: *SIAM Review* 29.4 (1987). Number: 4, pp. 575–596. ISSN: 0036-1445. URL: `http://siamdl.aip.org/sirev`.

[18] Haitao Liu et al. *When Gaussian Process Meets Big Data: A Review of Scalable GPs.* Number: arXiv:1807.01065. Apr. 9, 2019. arXiv: 1807.01065[cs,stat]. URL: http://arxiv.org/abs/1807.01065.

[19] Binbin Pan, Jianhuang Lai, and Pong C. Yuen. "Learning low-rank Mercer kernels with fast-decaying spectrum". In: *Neurocomputing* 74.17 (Oct. 2011), pp. 3028–3035. ISSN: 09252312. DOI: 10.1016/j.neucom.2011.04.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S092523121100302X.

[20] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* Red. by Francis Bach. Adaptive Computation and Machine Learning series. Cambridge, MA, USA: MIT Press, Nov. 23, 2005. 272 pp. ISBN: 978-0-262-18253-9.

[21] Youcef Saad and Yousef Saad. *Iterative methods for sparse linear systems.* 2nd ed. Philadelphia: SIAM, 2003. xiii+528. ISBN: 978-0-89871-534-7.

[22] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions". In: *Journal of Mathematical Psychology* 85 (Aug. 1, 2018), pp. 1–16. ISSN: 0022-2496. DOI: 10.1016/j.jmp.2018.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0022249617302158.

[23] Christoph Schwab and Radu Alexandru Todor. "Karhunen–Loève approximation of random fields by generalized fast multipole methods". In: *Journal of Computational Physics.* Uncertainty Quantification in Simulation Science 217.1 (Sept. 1, 2006), pp. 100–122. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2006.01.048. URL: https://www.sciencedirect.com/science/article/pii/S0021999106000349.

[24] Matthias Seeger. "Gaussian processes for machine learning". In: *International Journal of Neural Systems* 14.2 (Apr. 2004). Publisher: World Scientific Publishing Co., pp. 69–106. ISSN: 0129-0657. DOI: 10.1142/S0129065704001899. URL: https://www.worldscientific.com/doi/abs/10.1142/S0129065704001899.

[25] Radu Alexandru Todor. "Robust Eigenvalue Computation for Smoothing Operators". In: *SIAM Journal on Numerical Analysis* 44.2 (Jan. 2006), pp. 865–878. ISSN: 0036-1429, 1095-7170. DOI: 10.1137/040616449. URL: http://epubs.siam.org/doi/10.1137/040616449.

[26] Yung L. Tong. *The multivariate normal distribution.* Springer series in statistics. New York Berlin Heidelberg: Springer, 1990. 271 pp. ISBN: 978-1-4613-9655-0 978-1-4613-9657-4 978-0-387-97062-2 978-3-540-97062-0.

[27] Shashanka Ubaru, Jie Chen, and Yousef Saad. "Fast Estimation of tr(f(A)) via Stochastic Lanczos Quadrature". In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (Jan. 2017). Number: 4, pp. 1075–1099. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/16M1104974. URL: https://epubs.siam.org/doi/10.1137/16M1104974.

[28] Christopher Williams and Carl Rasmussen. "Gaussian Processes for Regression". In: *Advances in Neural Information Processing Systems.* Vol. 8. MIT Press, 1995. URL: https://proceedings.neurips.cc/paper/1995/hash/7cce53cf90577442771720a370c3c723-Abstract.html.

Appendices

A Appendix - Gradient of Marginal Log-Likelihood

Starting from the marginal log-likelihood (4), we wish to calculate the expression of its gradient with respect to hyperparameters θ_i. We first compute the derivative of the inverse of a matrix using the chain rule,

$$\mathbb{0} = \frac{d}{d\theta}\mathbb{I} = \frac{d}{d\theta}\left(A^{-1}(\theta)A(\theta)\right) = \frac{dA^{-1}(\theta)}{d\theta}A(\theta) + A^{-1}(\theta)\frac{dA(\theta)}{d\theta} \ ,$$

which yields

$$\frac{dA^{-1}(\theta)}{d\theta} = -A^{-1}(\theta)\frac{dA(\theta)}{d\theta}A^{-1}(\theta) \ .$$

The derivative of the determinant can be obtained with the Jacobi's formula ,

$$\frac{d}{d\theta}\det A(\theta) = \det(A(\theta)) \cdot \mathrm{Tr}\left(A^{-1}(\theta)\frac{dA(\theta)}{d\theta}\right) \ .$$

Applying it on the log determinant will cancel the $\det A(\theta)$ term. We retrieve Equation (5) by substitution of the above quantities for $A(\theta) := \widehat{K}_{XX}$ where the dependence of the kernel on hyperparameters is left implicit for notation simplicity.

B Appendix — mBCG Algorithm

We present below the modified batched conjugate gradients (mBCG) algorithm, which combines CG and Lanczos iteration for linear systems while handling several right hand-sides simultaneously. Modified refers to computing tridiagonal matrices $T_m^{(i)}$ as explained in section 2.2. Batched refers to handling multiple right-hand sides, we detail this extension in the following discussion. Matrix-vector multiplication $\mathbf{a}_i \mapsto \widehat{K}_{XX}\mathbf{a}_i$ is trivially extended to matrix-matrix multiplication $A \mapsto \widehat{K}_{XX}A$ since

$$\widehat{K}_{XX}A = \widehat{K}_{XX}\begin{bmatrix}\mathbf{a}_1 & \dots & \mathbf{a}_p\end{bmatrix} = \begin{bmatrix}\widehat{K}_{XX}\mathbf{a}_1 & \dots & \widehat{K}_{XX}\mathbf{a}_p\end{bmatrix} \ .$$

Vectors $\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k \in \mathbb{R}^p$ store the coefficients at each step k for the p right-hand sides. Their computation involves the function φ which simply generalizes dot products:

$$\varphi : \mathbb{R}^{n\times p} \times \mathbb{R}^{n\times p} \to \mathbb{R}^p, \quad \varphi(\begin{bmatrix}\mathbf{a}_1 & \dots & \mathbf{a}_p\end{bmatrix}, \begin{bmatrix}\mathbf{b}_1 & \dots & \mathbf{b}_p\end{bmatrix}) = \begin{bmatrix}\mathbf{a}_1^\top\mathbf{b}_1 & \dots & \mathbf{a}_p^\top\mathbf{b}_p\end{bmatrix}^\top \ ,$$

which is programatically implemented as element-wise multiplication of A and B and then summing across the first dimension. The ./ symbol then refers to element-wise division. Finally, note that

$$A\,\mathrm{diag}(\boldsymbol{\alpha}) = \begin{bmatrix}\alpha_1\mathbf{a}_1 & \dots & \alpha_p\mathbf{a}_p\end{bmatrix}$$

simply weights columns of A by coefficients in $\boldsymbol{\alpha}$. Explicit computation of the diagonal matrix is straightforward to avoid using Numpy's broadcasting rules[6], e.g. `alpha.T * A`.

[6]See `https://numpy.org/doc/stable/user/basics.broadcasting.html`

Algorithm 3: mBCG (Algorithm 2 of [10])

```
Input  : matrix-matrix multiplication oracles $A \mapsto \widehat{K}_{XX} A$ and $A \mapsto P^{-1} A$, right-hand side
         $B = \begin{bmatrix} \mathbf{y} & \mathbf{z}_1 & \dots & \mathbf{z}_N \end{bmatrix}$, max number of steps $m$, tolerance tol
Output: approximate solutions $U_m \approx \widehat{K}_{XX}^{-1} B$, partial Lanczos tridiagonalizations $T_m^{(i)}$
// Initialize solutions and residuals
$U_0 \leftarrow \mathbb{0}_{n \times (N+1)}$
$R_0 \leftarrow B$
// Initialize preconditioned residuals and search directions
$Z_0 \leftarrow P^{-1} R_0$
$D_1 \leftarrow R_0$
// Initialize tridigonal matrices
$T_m^{(1)}, \dots, T_m^{(N)} \leftarrow \mathbb{0}_{m \times m}$
for $k = 1, \dots, m$ do
    // Line search:  optimal step sizes in directions $D_k$
    $\boldsymbol{\alpha}_k \leftarrow \varphi(R_k, Z_k) \,./\, \varphi(D_k, \widehat{K}_{XX} D_k)$
    // Update solutions and residuals
    $U_k \leftarrow U_{k-1} + D_k \operatorname{diag}(\boldsymbol{\alpha}_k)$
    $R_k \leftarrow R_{k-1} - \widehat{K}_{XX} D_k \operatorname{diag}(\boldsymbol{\alpha}_k)$
    $Z_k \leftarrow P^{-1} R_k$
    // Conjugate Gram Schmidt and compute next search directions
    $\boldsymbol{\beta}_k \leftarrow \varphi(R_k, Z_k) \,./\, \varphi(R_{k-1}, Z_{k-1})$
    $D_{k+1} \leftarrow Z_k + D_k \operatorname{diag}(\boldsymbol{\beta}_k)$
    // Compute tridiagonal matrices
    $[T_m^{(i)}]_{kk} \leftarrow 1/[\boldsymbol{\alpha}_k]_i + [\boldsymbol{\beta}_{k-1}]_i / [\boldsymbol{\alpha}_{k-1}]_i, \quad i = 1, \dots, N$
    $[T_m^{(i)}]_{k-1,k}, [T_m^{(i)}]_{k,k-1} \leftarrow \sqrt{[\boldsymbol{\beta}_{k-1}]_i} / [\boldsymbol{\alpha}_k]_i, \quad i = 1, \dots, N$
    // Check for convergence
    if $\max_i \|(R_k)_{:,i}\|_2 / \|B_{:,i}\|_2 < tol$ then
        Stop the algorithm
    end
end
```