Bias-Free Scalable Gaussian Processes via Randomized Truncations

Andres Potapczynski [*1] **Luhuan Wu** [*2] **Dan Biderman** [*1] **Geoff Pleiss** [1] **John P. Cunningham** [1 2]

Abstract

Scalable Gaussian Process methods are computationally attractive, yet introduce modeling biases that require rigorous study. This paper analyzes two common techniques: early truncated conjugate gradients (CG) and random Fourier features (RFF). We find that both methods introduce a systematic bias on the learned hyperparameters: CG tends to underfit while RFF tends to overfit. We address these issues using randomized truncation estimators that eliminate bias in exchange for increased variance. In the case of RFF, we show that the bias-to-variance conversion is indeed a trade-off: the additional variance proves detrimental to optimization. However, in the case of CG, our unbiased learning procedure meaningfully outperforms its biased counterpart with minimal additional computation. Our code is available at `https://github.com/cunningham-lab/RTGPS`.

1. Introduction

Gaussian Processes (GP) are popular and expressive nonparametric models, and considerable effort has gone into alleviating their cubic runtime complexity. Notable successes include inducing point methods (e.g. Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013), finite-basis expansions (e.g. Rahimi & Recht, 2008; Mutnỳ & Krause, 2018; Wilson et al., 2020; Loper et al., 2020), nearest neighbor truncations (e.g. Datta et al., 2016; Katzfuss et al., 2021), and iterative numerical methods (e.g. Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018). Common to these techniques is the classic *speed-bias tradeoff*: coarser GP approximations afford faster but more biased solutions that in turn affect both the model's predictions and learned hyperparameters. While a few papers analyze the bias of inducing point methods (Bauer et al., 2016; Burt et al., 2019), the biases of other approximation techniques, and their subsequent impact on learned GP models, have not been rigorously studied.

Here we scrutinize the biases of two popular techniques – random Fourier features (RFF) (Rahimi & Recht, 2008) and conjugate gradients (CG) (e.g. Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018). These methods are notable due to their popularity and because they allow dynamic control of the speed-bias tradeoff: at any model evaluation, the user can adjust the number of CG iterations or RFF features to a desired level of approximation accuracy. In practice, it is common to truncate these methods to a fixed number of iterations/features that is deemed adequate. However, such truncation will stop short of an exact (machine precision) solution and potentially lead to biased optimization outcomes.

We provide a novel theoretical analysis of the biases resulting from RFF and CG on the GP log marginal likelihood objective. Specifically, we prove that CG is biased towards hyperparameters that underfit the data, while RFF is biased towards overfitting. In addition to yielding suboptimal hyperparameters, these biases hurt posterior predictions, regardless of the inference method used at test-time. Perhaps surprisingly, this effect is not subtle, as we will demonstrate. Our analysis suggests there is value in debiasing GP learning with CG and RFF. To do so, we turn to recent work that shows the merits of exchanging the speed-bias tradeoff for a *speed-variance tradeoff* (Beatson & Adams, 2019; Chen et al., 2019; Luo et al., 2020; Oktay et al., 2020). These works all introduce a randomization procedure that reweights elements of a fast truncated estimator, eliminating its bias at the cost of increasing its variance.

We thus develop bias-free versions of GP learning with CG and RFF using randomized truncation estimators. In short, we randomly truncate the number of CG iterations and RFF features, while reweighting intermediate solutions to maintain unbiasedness. Our variant of CG uses the **R**ussian **R**oulette estimator (Kahn, 1955), while our variant of RFF uses the **S**ingle **S**ample estimator of Lyne et al. (2015). We believe our **RR-CG** and **SS-RFF** methods to be the first to produce unbiased estimators of the GP log marginal likelihood with $< \mathcal{O}(N^3)$ computation.

[*]Equal contribution (randomly ordered). [1]Zuckerman Institute, Columbia University [2]Statistics Department, Columbia University. Correspondence to: Andres Potapczynski <ap3635@columbia.edu>, Luhuan Wu <lw2827@columbia.edu>, Dan Biderman <db3236@columbia.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Finally, through extensive empirical evaluation, we find our methods and their biased counterparts indeed constitute a bias-variance tradeoff. Both RR-CG and SS-RFF are unbiased, recovering nearly the same optimum as the exact GP method, while GP trained with CG and RFF often converge to solutions with worse likelihood. We note that bias elimination is not always practical. For SS-RFF, the optimization is slow, due to the large auxiliary variance needed to counteract the slowly decaying bias of RFF. On the other hand, RR-CG incurs a minimal variance penalty, likely due to the favorable convergence properties of CG. In a wide range of benchmark datasets, RR-CG demonstrates similar or better predictive performance compared to CG using the same expected computational time.

To summarize, this work offers three main contributions:

- theoretical analysis of the bias of CG- and RFF-based GP approximation methods (§3)
- RR-CG and SS-RFF: bias-free versions of these popular GP approximation methods (§4)
- results demonstrating the value of our RR-CG and SS-RFF methods (§3 and 5).

2. Background

We consider observed data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ for $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and the standard GP model:

$$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot)),$$
$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}\ 0, \sigma^2$$

where $k(\cdot,\cdot)$ is the covariance kernel, $\mu(\cdot)$ is set to zero without loss of generality, and hyperparameters are collected into the vector , which is optimized as:

$$\begin{aligned} &= \arg\min\ \mathcal{L}(\ ) \\ \mathcal{L}(\ ) &= -\log p(\mathbf{y} \mid \mathbf{X};\ ) \\ &= \frac{1}{2}\ \underbrace{\log|\mathbf{K}_{\mathbf{XX}}|}_{\text{model complexity}} + \underbrace{\mathbf{y}^\top \mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}}_{\text{data fit}} + N\log 2\pi \end{aligned} \tag{1}$$

where $\mathbf{K}_{\mathbf{XX}} \in \mathbb{R}^{N\ N}$ is the Gram matrix of all data points with diagonal observational noise:

$$\mathbf{K}_{\mathbf{XX}}[i,j] = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma^2 \mathbf{I}_{i=j}.$$

Following standard practice, we optimize with gradients:

$$\frac{\partial \mathcal{L}}{\partial} = \frac{1}{2}\ \operatorname{tr}\left\{\mathbf{K}_{\mathbf{XX}}^{-1}\frac{\partial \mathbf{K}_{\mathbf{XX}}}{\partial}\right\} - \mathbf{y}^\top \mathbf{K}_{\mathbf{XX}}^{-1}\frac{\partial \mathbf{K}_{\mathbf{XX}}}{\partial}\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}\ . \tag{2}$$

Three terms thus dominate the computational complexity: $\mathbf{y}^\top \mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}$, $\log|\mathbf{K}_{\mathbf{XX}}|$, and $\operatorname{tr}\{\mathbf{K}_{\mathbf{XX}}^{-1}\frac{\partial \mathbf{K}_{\mathbf{XX}}}{\partial}\}$. The common approach to computing this triad is the Cholesky factorization, requiring $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space.

Extensive literature has accelerated the inference and hyperparameter learning of GP. Two very popular strategies are using *conjugate gradients* (Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018; Wang et al., 2019) to approximate the linear solves in Eq. (2), and *random Fourier features* (Rahimi & Recht, 2008), which constructs a randomized finite-basis approximation of the kernel.

2.1. Conjugate Gradients

To apply conjugate gradients to GP learning, we begin by replacing the gradient in Eq. (2) with a stochastic estimate (Cutajar et al., 2016; Gardner et al., 2018):

$$\frac{\partial \mathcal{L}}{\partial} \approx \frac{1}{2}\ \mathbf{z}^\top \mathbf{K}_{\mathbf{XX}}^{-1}\frac{\partial \mathbf{K}_{\mathbf{XX}}}{\partial}\mathbf{z} - \mathbf{y}^\top \mathbf{K}_{\mathbf{XX}}^{-1}\frac{\partial \mathbf{K}_{\mathbf{XX}}}{\partial}\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}\ , \tag{3}$$

where $\mathbf{z}$ is a random variable such that $\mathbb{E}[\mathbf{z}] = 0$ and $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbf{I}$. Note that the first term constitutes a stochastic estimate of the trace term (Hutchinson, 1989). Thus, stochastic optimization of Eq. (1) can be reduced to computing the linear solves $\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{z}$.

Conjugate gradients (CG) (Hestenes et al., 1952) is an iterative algorithm for solving positive definite linear systems $\mathbf{A}^{-1}\mathbf{b}$. It consists of a three-term recurrence, where each new term requires only a matrix-vector multiplication with $\mathbf{A}$. More formally, each CG iteration computes a new term of the following summation:

$$\mathbf{A}^{-1}\mathbf{b} = \sum_{i=1}^{N} \gamma_i \mathbf{d}_i, \tag{4}$$

where the γ_i are coefficients and the $\mathbf{d}_i$ are conjugate search directions (Golub & Van Loan, 2012). N iterations of CG produce all N summation terms and recover the exact solution. In practice, exact convergence may require more than N iterations due to inaccuracies of floating point arithmetic. However, the summation converges exponentially, and so $J \ll N$ iterations may suffice to achieve high accuracy.

CG is an appealing method for computing $\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{z}$ due to its computational complexity and its potential for GPU-accelerated matrix products. J iterations takes at most $\mathcal{O}(JN^2)$ time and $\mathcal{O}(N)$ space if the matrix-vector products are performed in a map-reduce fashion (Wang et al., 2019). However, ill-conditioned kernel matrices hinder the convergence rate (Cutajar et al., 2016), and so the J^{th} CG iteration may yield an inaccurate approximation of Eq. (3).

2.2. Random Fourier Features

Rahimi & Recht (2008) introduce a randomized finite-basis approximation to stationary kernels:

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') \approx\ (\mathbf{x})^\top\ (\mathbf{x}') \tag{5}$$

where $(\mathsf{x}) \in \mathsf{R}^J$ and $J \ll N$. The RFF approximation relies on Bochner's theorem (Bochner et al., 1959): letting $\tau = \mathsf{x} - \mathsf{x}'$, all stationary kernels $k(\tau)$ on R^d can be exactly expressed as the Fourier dual of a nonnegative measure $\mathsf{P}(\,!\,)$: $k(\tau) = \int_{\mathsf{R}} \mathsf{P}(\,!\,)\exp(i\,!\,\tau)\,d\,!\,$. A Monte Carlo approximation of this Fourier transform yields:

$$k(\tau) \approx \frac{2}{J}\sum_{j=1}^{J/2} \exp(i\,!_j\tau), \quad !_j \sim \mathsf{P}(\,!\,),$$

which simplifies to a finite-basis approximation:

$$\mathsf{K}_{\mathsf{x}\mathsf{x}} \approx [\ (\mathsf{x}_1)\ldots\ (\mathsf{x}_n)][\ (\mathsf{x}_1)\ldots\ (\mathsf{x}_n)]^\top,$$
$$(\mathsf{x}) = [\cos\ !_i^\top \mathsf{x}\ ,\ \sin\ !_i^\top \mathsf{x}\ ]_{i=1}^{J=2}, \quad !_i \sim \mathsf{P}(\,!\,).$$

For many common kernels, $\mathsf{P}(\,!\,)$ can be computed in closed-form (e.g. RBF kernels have zero-mean Gaussian spectral densities). The approximated log likelihood can be computed in $\mathcal{O}(J^3+N)$ time and $\mathcal{O}(JN)$ space using the Woodbury inversion lemma and the matrix determinant lemma, respectively. The number of random features $J/2$ is a user choice, with typical values between 100-1000. More features lead to more accurate kernel approximations.

2.3. Unbiased Randomized Truncation

We will now briefly introduce Randomized Truncation Estimators, which are the primary tool we use to unbias the CG and RFF log marginal likelihood estimates. At a high level, assume that we wish to estimate some quantity ψ that can be expressed as a (potentially-infinite) series:

$$\psi = \sum_{j=1}^{H} \Delta_j, \quad H \in \mathsf{N} \cup \{\infty\}.$$

Here and in the following sections, Δ_j can either be random or deterministic. To avoid the expensive evaluation of the full summation, a randomized truncation estimator chooses a random term $J \in \{1,\ldots,H\}$ with probability mass function $\mathsf{P}(J) = \mathsf{P}(\mathcal{J} = J)$ after which to truncate computation. In the following, we introduce two means of deriving unbiased estimators by upweighting the summation terms.

The Russian Roulette estimator (Kahn, 1955) obtains an unbiased estimator $\bar{\psi}_J$ by truncating the sum after $J \sim \mathsf{P}(J)$ terms and dividing the surviving terms by their survival probabilities:

$$\bar{\psi}_J = \sum_{j=1}^{J} \frac{\Delta_j}{\mathsf{P}(\mathcal{J} \geq j)} = \sum_{j=1}^{J} \frac{\mathsf{I}_{J\ \ j}}{\mathsf{P}(\mathcal{J} \geq j)}\ \ \Delta_j, \qquad (6)$$

and, $\mathsf{E}[\bar{}_J] = \sum_{i=1}^{H} \Delta_j = \psi$. (See appendix for further derivation.) The choice of $\mathsf{P}(J)$ determines both the computational efficiency and the variance of $\bar{\psi}_J$. A thin-tailed $\mathsf{P}(J)$ will often truncate sums after only a few terms ($J \ll H$). However, tail events ($J \approx H$) are upweighted inversely to their low survival probability, and so thin-tailed truncation distributions may lead to high variance.

The Single Sample estimator (Lyne et al., 2015) implements an alternative reweighting scheme. After drawing $J \sim \mathsf{P}(J)$, it computes a single summation term Δ_J, which it upweights by $1/\mathsf{P}(J)$:

$$\bar{\psi}_J = \frac{\Delta_J}{\mathsf{P}(J)} = \sum_{j=1}^{H} \frac{\mathsf{I}_{J=j}}{\mathsf{P}(\mathcal{J}=j)}\ \ \Delta_j. \qquad (7)$$

This procedure is unbiased, and it amounts to estimating ψ using a single importance-weighted sample from $\mathsf{P}(J)$ (see appendix). Again, $\mathsf{P}(J)$ controls the speed/variance trade-off. We refer the reader to (Beatson & Adams, 2019) for a detailed comparison of these two estimators. We emphasize that both estimators remain unbiased even if Δ_j is a random variable, as long as it is independent from the random truncation integer J.

3. GP Learning with CG and RFF is Biased

Here we prove that early truncated CG and RFF provide biased approximations to the terms comprising the GP log marginal likelihood (Eq. 1). We also derive the bias decay rates for each method. We then empirically demonstrate these biases and show they affect the hyperparameters learned through optimization. Remarkably, we find that the above biases are *highly systematic*: CG-based GP learning favors underfitting hyperparameters while RFF-based learning favors overfitting hyperparameters.

3.1. CG Biases GP Towards Underfitting

In the GP literature, CG has often been considered an "exact" method for computing the log marginal likelihood (Cunningham et al., 2008; Cutajar et al., 2016), as the iterations are only truncated after reaching a pre-specified residual error threshold (e.g. 10^{-10}). However, as CG is applied to ever-larger kernel matrices it is common to truncate the CG iterations before reaching this convergence (Wang et al., 2019). While this accelerates the hyperparameter learning process, the resulting solves and gradients can no longer be considered "exact." In what follows, we show that the early-truncated CG optimization objective is not only approximate but also systematically biased towards underfitting.

To analyze the early-truncation bias, we adopt the analysis of Gardner et al. (2018) that recovers the GP log marginal likelihood (Eq. 1) from the stochastic gradient's CG estimates of $\mathsf{k}_{\mathsf{x}\mathsf{x}}^{-1}\mathsf{y}$ and $\mathsf{k}_{\mathsf{x}\mathsf{x}}^{-1}\mathsf{z}$ (Eq. 3). Recall the two terms in the log marginal likelihood are the "data fit" term $\mathsf{y}^\top \mathsf{k}_{\mathsf{x}\mathsf{x}}^{-1}\mathsf{y}$ and the "model complexity" term $\log|\mathsf{k}_{\mathsf{x}\mathsf{x}}|$. The first term falls directly out of the CG estimate of $\mathsf{k}_{\mathsf{x}\mathsf{x}}^{-1}\mathsf{y}$, while a

Figure 1. CG (left) systematically overestimates $\log|K_{XX}|$ and underestimates $y^\top K_{XX}^{-1} y$ whereas RFF (right) does the opposite. The dashed orange line shows the exact values computed by Cholesky. Our unbiased methods, RR-CG (left) and SS-RFF (right), recover the true $\log|K_{XX}|$ and $y^\top K_{XX}^{-1} y$ values. For these two methods, the x-axis indicates the expected number of iterations/features.

Figure 2. Kernel lengthscale values learned by optimizing (biased) CG and RFF log marginal likelihood approximations. CG overestimates the optimal kernel lengthscale whereas RFF underestimates it. We plot the divergence (in log-ratio scale) between the learned and true lengthscales as a function of the number of CG iterations (left) and of the number of RFF samples (right).

stochastic estimate of $\log|K_{XX}|$ can be obtained through the byproducts of CG's computation for $K_{XX}^{-1} z$. Gardner et al. (2018) show that the CG coefficients in Eq. (4) can be manipulated to produce a partial tridiagonalization: $T_z^{(J)} = Q_z^{(J)\top} K_{XX} Q_z^{(J)}$; where $T_z^{(J)} \in \mathbb{R}^{J \times J}$ is tridiagonal. $T_z^{(J)}$ can compute the Stochastic Lanczos Quadrature estimate of K_{XX} (Ubaru et al., 2017; Dong et al., 2017):

$$\log|K_{XX}| = \mathbb{E}\left[z^\top (\log K_{XX}) z\right] \approx \|z\|^2 e_1^\top \log T_z^{(J)} e_1, \tag{8}$$

where $\log(\cdot)$ is the matrix logarithm and e_1 is the first row of the identity matrix. The following theorem analyzes the bias of these $y^\top K_{XX}^{-1} y$ and $\log|K_{XX}|$ estimates:

Theorem 1. Let u_J and v_J be the estimates of $y^\top K_{XX}^{-1} y$ and $\log|K_{XX}|$ respectively after J iterations of CG; i.e.:

$$u_J = y^\top \sum_{i=1}^{J} \alpha_i d_i \;; \quad v_J = \|z\|^2 e_1^\top \log T_z^{(J)} e_1 :$$

If $J < N$, CG underestimates the inverse quadratic term and overestimates the log determinant in expectation:

$$u_J \le y^\top K_{XX}^{-1} y \;; \quad \mathbb{E}_z[v_J] \ge \log|K_{XX}| : \tag{9}$$

The biases of both terms decay at a rate of $O(C^{-2J})$, where C is a constant that depends on the conditioning of K_{XX}.

Proof sketch. The direction of the biases can be proved using a connection between CG and numeric quadrature. u_J and v_J are exactly equal to the J-point Gauss quadrature approximation of $y^\top K_{XX}^{-1} y$ and $\|z\|^2 e_1^\top (\log T_z^{(N)}) e_1$ represented as Riemann-Stieltjes integrals. The sign of the CG approximation bias follows from the standard Gauss quadrature error bound, which is negative for $y^\top K_{XX}^{-1} y$ and positive for $\log|K_{XX}|$. The convergence rates are from standard bounds on CG (Golub & Van Loan, 2012) and the analysis of Ubaru et al. (2017). See appendix for a full proof.

Fig. 1 confirms our theoretical analysis and demonstrates the systematic biases of CG. We plot the log marginal likelihood terms for a subset of the PoleTele UCI dataset, varying the number of CG iterations (J) used to produce the estimates. Compared against the exact terms (computed with Cholesky), we see an overestimation of $\log|K_{XX}|$ and an underestimation of $y^\top K_{XX}^{-1} y$. These biases are most prominent when using few CG iterations.

We turn to study the effect of CG learning on hyperparameters. Since the log marginal likelihood is a nonconvex function of θ, it is not possible to directly prove how the bias affects θ. Nevertheless, we know intuitively that underestimating $y^\top K_{XX}^{-1} y$ de-prioritizes model fit while overestimating $\log|K_{XX}|$ over-penalizes model complexity. Thus, the learned hyperparameters will likely underfit the data. Underfitting may manifest in an overestimation of the learned lengthscale, as low values of ℓ increase the flexibility and the complexity of the model. This hypothesis is empirically confirmed in Fig. 2 (left panel). We train a GP regression model on a toy dataset $y = x \sin(5x) + \varepsilon$

and" N (0; 0:01). We x all hyperparameters other than the lengthscale, which is learned using both CG-based optimization and (exact) Cholesky-based optimization. The overestimation of decays with the number of CG iterations.

3.2. RFF Biases GP Towards Over tting

Previous work has studied the accuracy of RFF's approximation to the entries of the Gram matrix $k(x; x^0)$ $(x)^\top (x^0)$ (Rahimi & Recht, 2008; Sutherland & Schneider, 2015). However, to the best of our knowledge there has been little analysis of nonlinear functions of this approximate Gram matrix, such as $y^\top \hat{K}_{XX}^{-1} y$ and $\log|\hat{K}_{XX}|$ appearing in the GP objective. Interestingly, we nd that RFF systematically biases these terms:

Theorem 2. Let $\hat{K}_J$ be the RFF approximation with $J=2$ random features. In expectation $\hat{K}_J$ overestimates the inverse quadratic and underestimates the log determinant:

$$E_{P(!)}\left[y^\top \hat{K}_J^{-1} y\right] \quad y^\top \hat{K}_{XX}^{-1} y \tag{10}$$

$$E_{P(!)}\left[\log|\hat{K}_J|\right] \quad \log|\hat{K}_{XX}|: \tag{11}$$

The biases of both terms decay at a rate $O(1=J)$.

Proof sketch. The direction of the biases is a straightforward application of Jensen's inequality, noting that $\hat{K}_{XX}^{-1}$ is a convex function and $\log|\hat{K}_{XX}|$ is a concave function. The magnitude of the bias is derived from a second-order Taylor expansion that closely resembles the analysis of Nowozin (2018). See appendix for full proof.

Again, Fig. 1 con rms the systematic biases of RFF, which decay at a rate proportional to the number of features, as predicted by Thm. 2. Hence, RFF should affect the learned hyperparameters in a manner opposite to CG. Overestimating $y^\top \hat{K}_{XX}^{-1} y$ emphasizes data tting while underestimating $\log|\hat{K}_{XX}|$ reduces the model complexity penalty, overall resulting in over tting behavior. Following the intuition presented in Sec. 3.1, we expect the lengthscale to be underestimated, as empirically con rmed by Fig. 2 (right panel). The gure also illustrates the slow decay of the RFF bias.

4. Bias-free Scalable Gaussian Processes

We debias the estimates of both the GP training objective in Eq. (1) and its gradient in Eq. (2) (as approximated by CG and RFF) using unbiased randomized truncation estimators. To see how such estimators apply to GP hyperparameter learning, we note that both CG and RFF recover the true log marginal likelihood (or an unbiased estimate thereof) in their limits:

Observation 1. CG recovers the exact log marginal likelihood in expectation in at most N iterations:

$$y^\top \hat{K}_{XX}^{-1} y = y^\top \sum_{j=1}^{N} {}_j d_j \quad ; \tag{12}$$

$$\log|\hat{K}_{XX}| = E_z\left[kzk^2 e_1^\top (\log T_z^{(N)}) e_1\right]: \tag{13}$$

By the law of large numbers, RFF converges almost surely to the exact log marginal likelihood as the number of random features goes to in nity:

$$y^\top \hat{K}_{XX}^{-1} y = \lim_{J!1} \; y^\top \hat{K}_J^{-1} y; \tag{14}$$

$$\log|\hat{K}_{XX}| = \lim_{J!1} \; \log|\hat{K}_J|: \tag{15}$$

To maintain the scalability of CG and RFFs while eliminating bias, we express the log marginal likelihood terms in Eqs. (12) to (15) as summations amenable to randomized truncation. We then apply the Russian Roulette and Single Sample estimators of Sec. 2.3 to avoid computing all summation terms while obtaining the same result in expectation.

4.1. Russian Roulette-Truncated CG (RR-CG)

The stochastic gradient in Eq. (3) requires performing two solves: $\hat{K}_{XX}^{-1} y$ and $\hat{K}_{XX}^{-1} z$. Using the summation formulation of CG (Eq. 4), we can write these two solves as series:

$$\hat{K}_{XX}^{-1} y = \sum_{i=1}^{N} {}_i d_i; \quad \hat{K}_{XX}^{-1} z = \sum_{i=1}^{N} {}_i {}^0 d_i^0;$$

where each CG iteration computes a new term of the summation. By applying the Russian Roulette estimator from Eq. (6), we obtain the following unbiased estimates:

$$\begin{aligned} \hat{K}_{XX}^{-1} y \quad & \sum_{j=1}^{J} ({}_j d_j)/P(J \quad j); \quad J \quad P(J) \\ \hat{K}_{XX}^{-1} z \quad & \sum_{j=1}^{J^0} ({}_j {}^0 d_j^0)/P(J \quad j): \quad J^0 \quad P(J^0); \end{aligned} \tag{16}$$

These unbiased solves produce an unbiased optimization gradient in Eq. (3); we refer to this approach as Russian Roulette CG (RR-CG). With the appropriate truncation distribution $P(J)$, this estimate affords the same computational complexity of standard CG without its bias.

We must compute two independent estimates of $\hat{K}_{XX}^{-1} y$ with different J $P(J)$ in order for the $y^\top \hat{K}_{XX}^{-1} \frac{@\hat{K}_{XX}}{@} \hat{K}_{XX}^{-1} y$ term in Eq. (3) to be unbiased. Thus, the unbiased gradient requires 3 calls to RR-CG, as opposed to the 2 CG calls needed for the biased gradient. Nevertheless, RR-CG has the same $O(JN^2)$ complexity as standard CG – and the additional solve can be computed in parallel with the others.

We can also use the Russian Roulette estimator to compute the log marginal likelihood itself, though this is not strictly necessary for gradient-based optimization. (See appendix.)

Choosing the truncation distribution. Since the Russian Roulette estimator is unbiased for any choice of $P(J)$,

we wish to choose a truncation distribution that balances computational cost and variance.[1] Beatson & Adams (2019) propose the relative optimization efficiency (ROE) metric, which is the ratio of the expected improvement of taking an optimization step with our gradient estimate to its computational cost. A critical requirement of the ROE analysis is the expected rate of decay of our approximations in terms of the number of CG iterations J. We summarize our estimates and choices of distribution as follows:

Theorem 3. The approximation to $\log|\hat{K}_{XX}|$ and to $y^\top \hat{K}_{XX}^{-1} y$ using RR-CG decays at a rate of $\mathcal{O}(C^{-2J})$. Therefore the truncation distribution that maximizes the ROE is $P(J) \propto C^{-2J}$, where C is a constant that depends on the conditioning of $\hat{K}_{XX}$. The expected computation and variance of $P(J)$ is finite.

Proof sketch. Beatson & Adams (2019) show that the truncation distribution that maximizes the ROE is proportional to the rate of decay of our approximation divided by its computational cost. The error of CG decays as $\mathcal{O}(C^{-2J})$, and the cost of each summation term is constant with respect to J. In practice, we vary the exponential decaying rate to control the expectation and variance of $P(J)$. To further reduce the variance of our estimator, we set a minimum number of CG iterations to be computed, as in (Luo et al., 2020). See appendix for full proof.

In practice, however, we do not have access to C since computing the conditioning of $\hat{K}_{XX}$ is impractical. Yet, we can change the base of the exponential a and add a temperature parameter b to rescale the function and control the rate of decay of the truncation distribution as a sensible alternative. Thus, we follow a more general exponential decay distribution:

$$P(J) \propto e^{-aJ}; \quad J = J_{\min}, \ldots, N \tag{17}$$

where $J_{\min}$ is the minimum truncation number. By varying the values of a and $J_{\min}$, we can control the expectation and standard deviation of $P(J)$. In practice we found that having the standard deviation value between 10 and 20 achieves stable GP learning process, which can be obtained by tuning a between 0.05 and 0.1 for sufficiently large datasets (e.g. $N \geq 500$). We also noticed that the method is not sensitive to these choices of hyperparameters and that they work well across all the experiments. The expected truncation number can be further tuned by varying $J_{\min}$. We emphasize that these choices impact the speed-variance tradeoff. By setting a larger $J_{\min}$ we decrease the speed by requiring more baseline computations but also decrease the variance (since the minimum approximations have the largest deviations from the ground truth).

[1] Solely minimizing variance is not appealing, as this is achieved by $P(J) = \mathbb{I}_{J=N}$ which has no computational savings.

Toy problem. In Fig. 1 we plot the empirical mean of the RR-CG estimator using 10^4 samples from an exponential truncation distribution. We find that RR-CG produces unbiased estimates of the $y^\top \hat{K}_{XX}^{-1} y$ and $\log|\hat{K}_{XX}|$ terms that are indistinguishable from the exact values computed with Cholesky. Reducing the expected truncation iteration $E(J)$ (x-axis) increases the standard error of empirical means, demonstrating the speed-variance trade-off.

4.2. Single Sample-Truncated RFF (SS-RFF)

Denoting $\hat{K}_j$ as the kernel matrix estimated by j random Fourier features, we can write $\log|\hat{K}_{XX}|$ as the following telescoping series:

$$\log|\hat{K}_{XX}| = \log|\hat{K}_1| + \sum_{j=2}^{N-2} \log|\hat{K}_j| - \log|\hat{K}_{j-1}| + \log|\hat{K}_{XX}| - \log|\hat{K}_{N-2-1}| \tag{18}$$

$$= \log|\hat{K}_1| + \sum_{j=2}^{N-2} \Delta_j + \Delta_{N-2}; \tag{19}$$

where Δ_j is defined as $\log|\hat{K}_j| - \log|\hat{K}_{j-1}|$ for all $j < N-2$, and Δ_{N-2} is defined as $\log|\hat{K}_{XX}| - \log|\hat{K}_{N-2-1}|$. Note that each Δ_j is now a random variable, since it depends on the random Fourier frequencies. Crucially, we only include $N-2$ terms in the series so that no term requires more than $\mathcal{O}(N^3)$ computation in expectation. (For any $j > N-2$, $\hat{K}_j$ is a full-rank matrix and thus is as computationally expensive as the true $\hat{K}_{XX}$ matrix.) We construct a similar telescoping series for $y^\top \hat{K}_{XX}^{-1} y$.

As with Eq. (16), we approximate the series in Eq. (19) with a randomized truncation estimator, this time using the Single Sample estimator (7):

$$\log|\hat{K}_{XX}| \approx \log|\hat{K}_1| + \Delta_J / P(J). \tag{20}$$

where J is drawn from the truncation distribution $P(J)$ with support over $\{2, 3, \ldots, N-2\}$. Note that the Single Sample estimate requires computing 3 log determinants $\log|\hat{K}_1|$, $\log|\hat{K}_{J-1}|$, and $\log|\hat{K}_J|$) for a total of $\mathcal{O}(J^3 + NJ^2)$ computations and $\mathcal{O}(NJ)$ memory. This is asymptotically the same requirement as standard RFF. The Russian Roulette estimator, on the other hand, incurs a computational cost of $\mathcal{O}(NJ^3 + J^4)$ as it requires computing $\mathcal{O}(J)$ $\log|\hat{K}_1|$ through $\log|\hat{K}_J|$) which quickly becomes impractical for large J.

A similar Single Sample estimator constructs an unbiased estimate of $y^\top \hat{K}_{XX}^{-1} y$. Backpropagating through these Single Sample estimates produces unbiased estimates of the log marginal likelihood gradient in Eq. (2).

Choosing the truncation distribution. For the Single Sample estimator we do not have to optimize the ROE since

minimizing the variance of this estimator does not result in a degenerate distribution.

Theorem 4. The truncation distribution that minimizes the variance of the SS-RFF estimators for $\log|\hat{K}_{XX}|$ and $y^\top \hat{K}_{XX}^{-1} y$ is $P(J) \propto 1/J$. The expected variance and computation of $P(J)$ is finite.

Proof sketch. The minimum variance distribution can be found by solving a constrained optimization problem. In practice, we can further decrease the variance of our estimator by fixing a minimum value of RFF features to be used in Eq. (19) and by increasing the step size $c \geq 2$ between the elements at each $\Delta_j = \log|\hat{K}_{cJ}| - \log|\hat{K}_{c(J-1)}|$. See appendix for full proof.

For the experiments we started with 500 features and also tried various step sizes $c \in \{1, 10, 100\}$. The variance of the estimator decreases as we increase c since the probability weights will decrease in magnitude. Yet, despite of using the optimal truncation distribution, setting a high number of features and taking long steps $c = 100$, the variance of the estimator still requires taking several steps before converging, making SS-RFF computationally impractical.

Toy problem. Similar to RR-CG, in Fig. 1 we plot the empirical mean of the SS-RFF estimator using 10^4 samples. We find that SS-RFF produces unbiased estimates of the $y^\top \hat{K}_{XX}^{-1} y$ and $\log|\hat{K}_{XX}|$ terms. However, these estimates have a higher variance when compared to the estimates of RR-CG. Reducing the expected truncation iteration $E(J)$ (x-axis) increases the standard error of the empirical means, demonstrating the speed-variance trade-off.

4.3. Analysis of the Bias-free Methods

Randomized truncations and conjugate gradients have existed for many decades (Hestenes et al., 1952; Kahn, 1955), but have rarely been used in conjunction. Filippone & Engler (2015) proposed a closely related method, combining randomized early-turncatation with CG to draw GP posterior samples over covariance parameters. We differ by providing the first theoretical analysis of the biases on GP learning incurred by CG and RFF, and proceed to remedy these biases using the Russian Roulette and the Simple Sample estimators.

To some extent, randomized truncation methods are antithetical to the original intention of CG: producing deterministic and nearly exact solves. For large-scale applications, where early truncation is necessary for computational tractability, the ability to trade bias for variance is beneficial. This fact is especially true in the context of GP learning, where the bias of early truncation is systematic and cannot be simply explained away as numerical imprecision.

Randomized truncation estimates are often used to estimate in finite series, where it is challenging to design truncation distributions with finite expected computation and/or variance. We avoid such issues since CG and the telescoping RFF summations are both finite.

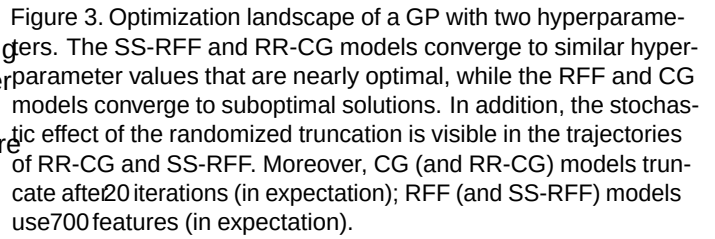

Figure 3. Optimization landscape of a GP with two hyperparameters. The SS-RFF and RR-CG models converge to similar hyperparameter values that are nearly optimal, while the RFF and CG models converge to suboptimal solutions. In addition, the stochastic effect of the randomized truncation is visible in the trajectories of RR-CG and SS-RFF. Moreover, CG (and RR-CG) models truncate after 20 iterations (in expectation); RFF (and SS-RFF) models use 700 features (in expectation).

5. Results

First, we show that our bias-free methods recover nearly the same hyperparameters as exact methods (i.e. Cholesky-based optimization), whereas models that use CG and RFF converge to suboptimal hyperparameters. Since RR-CG and SS-RFF eliminate bias at the cost of increased variance, we then demonstrate the optimization convergence rate and draw conclusions on our methods' applicability. Finally, we compare models optimized with RR-CG against a host of approximate GP methods across a wide range of UCI datasets (Asuncion & Newman, 2007). All experiments are implemented in GPyTorch (Gardner et al., 2018)

Optimization trajectories of bias-free GP. Fig. 3 displays the optimization landscape – the log marginal likelihood of the PoleTele dataset – as a function of (RBF) kernel lengthscale $\grave{e}$ and noise ^{2}. As expected, an exact GP (optimized using Cholesky, Fig. 3 upper left) recovers the optimum. Notably, the GP trained with standard CG and RFF converges to suboptimal hyperparameters (upper right/lower right). RR-CG and SS-RFF models (trained with 20 iterations and 700 features in expectation, respectively) successfully eliminate this bias, and recover nearly

Figure 4. The GP optimization objective for models trained with RFF and SS-RFF. (PoleTele dataset, RBF kernel, Adam optimizer.) RFF models converge to sub-optimal log marginal likelihoods. SS-RFF models converge to (near) optimum values, yet require more than 100 as many optimization steps.

Figure 5. The GP optimization objective for models trained with CG and RR-CG. (Bike dataset, RBF kernel, Adam optimizer.) RR-CG models converge to optimal solutions, while the (biased) CG models diverge. Increasing the expected truncation of RR-CG only slightly improves optimization convergence; models converge in < 100 steps of Adam.

the same parameters as the exact model (upper center/lower left). These plots also show the speed-variance tradeoff of randomized truncation. SS-RFF and RR-CG have noisy optimization trajectories due to auxiliary truncation variance.

Convergence of GP hyperparameter optimization. Figs. 4 and 5 plot the exact GP log marginal likelihood of the parameters learned by each method during optimization. Each trajectory corresponds to a RBF-kernel GP trained on the PoleTele dataset (Fig. 4) and the Bike dataset (Fig. 5).

Fig. 4 shows that RFF models converge to solutions with worse log likelihoods, and more RFF features slow the rate of optimization. Additionally, we see the cost of the auxiliary variance needed to debias SS-RFF: while SS-RFF models achieve better optima than their biased counterparts, they take 2-3 orders of magnitude longer to converge, despite using a truncation distribution that minimizes variance. We thus conclude that SS-RFF has too much variance to be practical for GP hyperparameter learning.

Fig. 5 on the other hand shows that RR-CG is minimally affected by its auxiliary variance. The GP trained with RR-CG converges in roughly 100 iterations, nearly matching Cholesky-based optimization. Decreasing the expected truncation value from $\mathbb{E}[J] = 40$ to 20 slightly slows this convergence. We note that the bias induced by standard CG can be especially detrimental to GP learning. On this dataset, the biased models deviate from their Cholesky counterparts and eventually diverge away from the optimum.

Predictive performance of bias-free GP. Lastly, we compare the predictive performance of GPs that use RR-CG, CG, and Cholesky for hyperparameter optimization. We emphasize that the RR-CG and CG methods only make use of early truncation approximations during training. At test time, we compute the predictive posterior by running CG to a tolerance of 10^{-4}, which we believe can be considered "exact" for practical intents and purposes. Additionally, we include four other (biased) scalable GP approximations methods as baselines: RFF, Stochastic Variational Gaussian Processes (SVGP) (Hensman et al., 2013), generalized Product of Expert Gaussian Processes (POE) (Cao & Fleet, 2014; Deisenroth & Ng, 2015), and stochastic gradient-based Gaussian Processes (sgGP) (Chen et al., 2020). We note that the RFF, SVGP, and sgGP methods introduce both bias and variance, as these methods rely on randomization and approximation.

We use CG with $J = 100$ iterations, and RR-CG with $\mathbb{E}[J] = 100$ expected iterations; both methods use the preconditioner of Gardner et al. (2018). All RFF models use 700 random features. For SVGP, we use 1,024 inducing points and minibatches of size 1,024 as in (Wang et al., 2019). The POE models are comprised of GP experts that are each trained on 1,024 data point subsets. For sgGP, the subsampled datasets are constructed by selecting a random point x, y and its 15 nearest neighbors as in (Chen et al., 2020). Each dataset is randomly split to 64% training, 16% validation and 20% testing sets. All kernels are RBF with a separate lengthscale per dimension. See appendix for more details.

We report prediction accuracy (RMSE) and negative log likelihood (NLL) in Fig. 6 (see appendix for full tables on predictive performance and training time). We make two key observations: (i) RR-CG meaningfully debiases CG. When the bias of CG is not detrimental to optimization (e.g. CG with 100 iterations is close to convergence for the Elevators dataset), RR-CG has similar performance. However, when the CG bias is more significant (e.g. the KEGG dataset), the bias-free RR-CG improves the GP predictive RMSE and NLL. We also include a figure displaying the predictive performance of RR-CG and CG with increasing number of (expected) CG iterations in appendix. (ii) RR-CG recovers the same optimum as the "ground-truth" method

Figure 6. Root-mean-square-error (RMSE) and negative log likelihood (NLL) of GP trained with CG (light purple), RR-CG (dark purple) and various approximate methods (grey). Dashed red lines indicates Cholesky-based GP performance (when applicable). Results are averaged over 3 dataset splits. Missing RFF and sgGP results correspond to (very high) outlier NLL / RMSE values. In almost all experiments, GP learning with RR-CG achieves similar or better performance compared to that with CG at the same computational cost.

(i.e. Cholesky) does, as indicated by the red-dashed line in Fig. 6. This result provides additional evidence that RR-CG achieves unbiased optimization.

While RR-CG obtains the lowest RMSE on all but 2 datasets, we note that the (biased) GP approximations sometimes achieve lower NLL. For example, SVGP has a lower NLL than that of RR-CG on the Bike dataset, despite having a higher RMSE. We emphasize that this is not a failing of RR-CG inference. The SVGP NLL is even better than that of the exact (Cholesky) GP, suggesting a potential model misspeci cation for this particular dataset. Since SVGP overestimates the observational noise (σ^2) (Bauer et al., 2016), it may obtain a better NLL when outliers are abundant. Though we cannot compare against the Cholesky posterior on larger datasets, we hypothesize that the NLL/RMSE discrepancy on these datasets is due to a similar modeling issue.

6. Conclusion

We prove that CG and RFF introduce systematic biases to the GP log marginal likelihood objective: CG-based training will favor under tting models, while RFF-based training will promote over tting. Modifying these methods with randomized truncation converts these biases into variance, enabling unbiased stochastic optimization. Our results show that this bias-to-variance exchange indeed constitutes a trade-off. The convergence of SS-RFF is impractically slow, likely due to the truncation variance needed to eliminate RFF's slowly-decaying bias. However, for CG-based training, we nd that variance is almost always preferable to bias. Models trained with RR-CG achieve better performance than those trained with standard CG, and tend to recover the hyperparameters learned with exact methods. Though models trained with CG do not always exhibit noticeable bias, RR-CG's negligible computational overhead is justi able to counteract cases where the bias is signi cant.

We reported experiments with at most 300K observations for our methods and baselines, which is substantial for GPs. We emphasize that RR-CG can be extended to datasets with over one million data points as in Wang et al. (2019). However, the computational cost is much higher, requiring multiple GPUs for training and testing.

We note that the RR-CG algorithm is not limited to GP applications. Future work should explore applying RR-CG to other optimization problems with large-scale solves.

Acknowledgements

This work was supported by the Simons Foundation, McKnight Foundation, the Grossman Center, and the Gatsby Charitable Trust.

References

Angelova, J. A. On moments of samples mean and variance. In Int. J. Pure Appl. Math, pp. 79:67–85, 2012.

Asuncion, A. and Newman, D. Uci machine learning repository, 2007.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, 2016.

Beatson, A. and Adams, R. P. Efficient optimization of loops and limits with randomized telescoping sums. In *International Conference on Machine Learning*, 2019.

Bochner, S. et al. *Lectures on Fourier integrals*, volume 42. Princeton University Press, 1959.

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pp. 862–871, 2019.

Cao, Y. and Fleet, D. J. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in Neural Information Processing Systems*, 33, 2020.

Chen, R. T. Q., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 2019.

Cunningham, J. P., Shenoy, K. V., and Sahani, M. Fast gaussian process methods for point process intensity estimation. In *International Conference on Machine learning*, pp. 192–199, 2008.

Cutajar, K., Osborne, M. A., Cunningham, J. P., and Filippone, M. Preconditioning kernel matrices. In *International Conference on Machine Learning*, 2016.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

Deisenroth, M. and Ng, J. W. Distributed gaussian processes. In *International Conference on Machine Learning*, pp. 1481–1490. PMLR, 2015.

Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. G. Scalable log determinants for gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, pp. 6327–6337, 2017.

Filippone, M. and Engler, R. Enabling scalable stochastic gradient-based inference for gaussian processes by employing the unbiased linear system solver (ulisse). In *International Conference on Machine Learning*, pp. 1015–1024. PMLR, 2015.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems*, pp. 7576–7586, 2018.

Golub, G. H. and Meurant, G. *Matrices, moments and quadrature with applications*, volume 30. Princeton University Press, 2009.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. The Johns Hopkins University Press, 4th Edition, 2012.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *Uncertainty in Artificial Intelligence*, 2013.

Hestenes, M. R., Stiefel, E., et al. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(1), 1952.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Kahn, H. Use of different monte carlo sampling techniques. *Rand Corporation*, 1955.

Katzfuss, M., Guinness, J., et al. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 36(1):124–141, 2021.

Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4), 1950.

Loper, J., Blei, D., Cunningham, J. P., and Paninski, L. General linear-time inference for gaussian processes on one dimension. *arXiv preprint arXiv:2003.05554*, 2020.

Luo, Y., Beatson, A., Norouzi, M., Zhu, J., Duvenaud, D., Adams, R. P., and Chen, R. T. Q. Sumo: Unbiased estimation of log marginal probability for latent variable models. In *International Conference on Learned Representations*, 2020.

Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. On russian roulette estimates for bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

Mutny, M. and Krause, A. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, pp. 9005–9016, 2018.

Nowozin, S. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *International Conference on Learned Representations*, 2018.

Oktay, D., McGreivy, N., Aduol, J., Beatson, A., and Adams, R. P. Randomized automatic differentiation. *arXiv preprint arXiv:2007.10412*, 2020.

Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. Constant-time predictive distributions for gaussian processes. In *International Conference on Machine learning*, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2008.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2006.

Sutherland, D. J. and Schneider, J. G. On the error of random fourier feaures. *Conference on Uncertainty in Arti cial Intelligence*, 2015.

Titsias, M. K. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Arti cial Intelligence and Statistics*, pp. 567–574, 2009.

Ubaru, S., Chen, J., and Saad, Y. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pp. 14648–14659, 2019.

Wilson, J., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. Ef ciently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pp. 10292–10302, 2020.

A. Proof of Biases for CG and RFF

A.1. Proof of Theorem 1

Proof. To prove the bias of the CG log marginal likelihood terms, we rely on a connection between conjugate gradients, the Lanczos algorithm (Lanczos, 1950), and Gauss quadrature. In particular, we demonstrate that u_J and v_J – CG's estimates for $y^\top \hat{K}_{XX}^{-1} y$ and $\log|\hat{K}_{XX}|$ terms – are equivalent to Gauss quadrature approximations of two Riemann-Stieljes integrals. We then use quadrature error analysis to prove the biases of these terms.

Expressing $y^\top \hat{K}_{XX}^{-1} y$ and $z^\top (\log \hat{K}_{XX}) z$ as Riemann-Stieljes integrals. First, we note that $y^\top \hat{K}_{XX}^{-1} y$ and $z^\top (\log \hat{K}_{XX}) z$ (our stochastic trace estimate of $\log|\hat{K}_{XX}|$) can both be expressed by the quadratic form $w^\top f(A) w$, where $f(A)$ denotes a matrix function of the matrix A, and w is a vector. Letting $A = P \Lambda P^\top$ be the eigendecomposition of A, we can write this quadratic form as

$$w^\top f(A) w = w^\top P f(\Lambda) P^\top w = \|w\|^2 \sum_{i=1}^{N} f(\lambda_i)\, \gamma_i^2,$$

where $\lambda_{\min} \le \lambda_i \le \lambda_{\max}$ are the diagonal elements of Λ (i.e. the eigenvalues) – ordered from smallest to largest, and γ_i are the components of $P^\top w / \|w\|$. The summation in the above equation can be expressed as a Riemann-Stieltjes integral:

$$I[f] := \|w\|^2 \sum_{j=1}^{N} f(\lambda_i)\, \gamma_i^2 = \|w\|^2 \int_{\lambda_{\min}}^{\lambda_{\max}} f(t)\, d\alpha_A(t); \qquad \text{(S1)}$$

where the measure $\alpha_A(t)$ is a piecewise constant function

$$\alpha_A(t) = \begin{cases} 0; & \text{if } t < \lambda_{\min}; \\ \sum_{j=1}^{i} \gamma_j^2; & \text{if } \lambda_i \le t < \lambda_{i+1} \\ \sum_{j=1}^{N} \gamma_j^2; & \text{if } \lambda_{\max} \le t. \end{cases} \qquad \text{(S2)}$$

See (Golub & Meurant, 2009) for more details.

The Lanczos algorithm approximates these integrals with Gauss quadrature. The Lanczos algorithm (Lanczos, 1950), which is briefly described in Sec. 3.1, iteratively expresses a symmetric matrix A via the partial tridiagonalization $T_w^{(J)} = Q_w^{(J)\top} A Q_w^{(J)}$. $Q_w^{(J)} \in \mathbb{R}^{N \times J}$ is a orthonormal matrix with $w/\|w\|$ as its first column, and $T_w^{(J)}$ is a $J \times J$ tridiagonal matrix. Briefly, the columns of $Q^{(J)}$ matrices are computed by performing Gram-Schmidt orthogonalization on the Krylov subspace $[w; Aw; A^2 w; \dots; A^{J-1} w]$, and storing the orthogonalization constants in $T_w^{(J)}$.

The Lanczos algorithm is commonly used to estimate quadratic forms:

$$w^\top f(A) w \approx w^\top Q_w^{(J)} f\left(T_w^{(J)}\right) Q_w^{(J)\top} w = \|w\|^2 e_1^\top f\left(T_w^{(J)}\right) e_1; \qquad \text{(S3)}$$

where e_1 is the first unit vector. Note that Eq. (S3) holds because the columns of $Q_w^{(J)}$ are orthonormal.

There is a well-established connection between Eq. (S3) and numeric quadrature (e.g. Golub & Meurant, 2009). More specifically, Eq. (S3) is exactly equivalent to J-term Gauss quadrature rule applied to the Riemann-Stieltjes integral in Eq. (S1). We can thus analyze Lanczos estimates of $w^\top f(A) w$ using standard Gauss quadrature error analysis.

Equivalence between CG and the Lanczos algorithm. We will now show an equivalence between our estimates $u_J \approx y^\top \hat{K}_{XX}^{-1} y$ and $v_J \approx z^\top (\log \hat{K}_{XX}) z$ and Lanczos algorithm approximations. Note that we have already established $v_J \approx z^\top (\log \hat{K}_{XX}) z$ as a Lanczos algorithm approximation in Eq. (8):

$$z^\top \approx \|z\|^2 e_1^\top (\log T_z^{(J)}) e_1.$$

For $u_J \approx y^\top \hat{K}_{XX}^{-1} y$, we exploit a connection between CG and Lanczos (Golub & Van Loan, 2012, Ch. 11.3):

$$\sum_{j=1}^{J} \gamma_j d_j = \|y\| Q_y^{(J)} (T_y^{(J)})^{-1} e_1; \qquad \text{(S4)}$$

where $\sum_{j=1}^{J} \gamma_j d_j$ (see Eq. 4) is the J^{th} CG approximation to $\hat{K}_{XX}^{-1} y$. Multiplying Eq. (S4) by $y^\top$, we have

$$u_J = y^\top \left(\sum_{j=1}^{J} \gamma_j d_j \right) = \|y\| y^\top Q_y^{(J)} (T_y^{(J)})^{-1} e_1 = \|y\|^2 e_1^\top (T_y^{(J)})^{-1} e.$$

Therefore, the CG approximations of $y^\top \hat{K}_{XX}^{-1} y$ and $z^\top (\log \hat{K}_{XX}) z$ are both Lanczos approximations. Putting all the pieces together, we have just shown that u_J and v_J are equivalent to J-term Gauss quadrature rules applied to the following integrals:

$$y^\top \hat{K}_{XX}^{-1} y = \|y\|^2 \int_{\lambda_{\min}}^{\lambda_{\max}} t^{-1}\, d\alpha_{\hat{K}_{XX}}(t); \qquad \text{(S5)}$$

$$z^\top \log \hat{K}_{XX}^{-1} z = \|z\|^2 \int_{\lambda_{\min}}^{\lambda_{\max}} \log(t)\, d\alpha_{\hat{K}_{XX}}(t); \qquad \text{(S6)}$$

where the measure $\alpha_{\hat{K}_{XX}}(t)$ is defined by Eq. (S2).

Applying Gauss quadrature error analysis to u_J and v_J. To analyze the bias of teh CG estimates, we make use of standard error Guass quadrature error analysis. If $L_G^{(J)}[f]$ is the J-term Gaussian quadrature approximation of the Riemann-Stieltjes integral $I[f]$ (see Eq. S1), the error can be exactly expressed as:

$$I[f] - L_G^{(J)}[f] = (\beta_1 \cdots \beta_J)^2 \frac{f^{(2J)}(\xi)}{(2J)!}, \qquad \text{(S7)}$$

for some $\xi \in [\lambda_{\min}, \lambda_{\max}]$, where $f^{(2J)}$ is the $2J^{\text{th}}$ derivative of f and $\{\beta_i\}$ are some quantities that depend on the spectrum of the matrix $\tilde{K}_{XX}$ (Golub & Meurant, 2009, Ch. 6).

Turning back to u_J, the corresponding function $f(t) = t^{-1}$ has even derivatives of the form $f^{(2J)} = (2J)!\, t^{-(2J+1)} > 0, \forall J, \forall t \in (\lambda_{\min}, \lambda_{\max})$. Replacing $I[f]$ with the integral defined by Eq. (S5), we have that $y^\top \tilde{K}_{XX}^{-1} y - u_J > 0$, which proves that CG underestimates $y^\top \tilde{K}_{XX}^{-1} y$. Similarly for v_J, the corresponding $f(t) = \log t$ has even derivatives of the form $f^{(2J)} = -(2J-1)!\, t^{-2J} < 0, \forall J, \forall t \in (\lambda_{\min}, \lambda_{\max})$. Replacing $I[f]$ with the integral defined by Eq. (S6), we have that $z^\top (\log \tilde{K}_{XX}) z - v_J > 0$.

The convergence rates of u_J and v_J follow from Eq. 4.4 of Ubaru et al. (2017). Let $\gamma = (\sqrt{\kappa}+1)/(\sqrt{\kappa}-1)$, where κ is the condition number of $\tilde{K}_{XX}$ (i.e. the ratio of its maximum and minimum singular values). Then

$$|y^\top \tilde{K}_{XX}^{-1} y - u_J| \leq C \cdot \gamma^{-2J},$$
$$|z^\top (\log \tilde{K}_{XX}) z - v_J| \leq C \cdot \gamma^{-2J},$$

where C is a constant that depends on the extremal eigenvalues of $\tilde{K}_{XX}$. □

A.2. Proof of Theorem 2

Proof. The inequalities are a result of applying Jensen's inequality to the inverse of a positive definite matrix (which is a convex function) and the log determinant of a positive definite matrix (which is a concave function). For example, for the $y^\top \tilde{K}_{XX}^{-1} y$ term we have that

$$\mathop{\mathbb{E}}_{P(\omega)}\left[y^\top \hat{K}_J^{-1} y\right] = y^\top \mathop{\mathbb{E}}_{P(\omega)}\left[\hat{K}_J^{-1}\right] y \geq y^\top \left[\mathop{\mathbb{E}}_{P(\omega)} \hat{K}_J\right]^{-1} y = y^\top \tilde{K}_{XX}^{-1} y$$

A similar procedure, but in the opposite direction applies to the $\log|\tilde{K}_{XX}|$.

To estimate the rate of decay of the biases, we rely on two keys ideas. The first idea is to translate the central moments of the kernel matrix being approximated by the random Fourier features into the central moments of these features. This strategy resembles the analysis in (Nowozin, 2018) which uses (Angelova, 2012, Theorem 1). The second idea is to use an approximation to two matrix functions. For the inverse function we use a Neumann series and for the logarithm we use a Taylor series. These series require that the eigenvalues of the approximation residual $I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}$ are less than 1. We will make this assumption as we know that for a large enough J, the kernel approximation will be close to $\tilde{K}_{XX}$. Below is a formal argument.

For a fixed ω we can write

$$\begin{aligned} y^\top \hat{K}_J^{-1} y &= y^\top \left[\tilde{K}_{XX}^{1/2} \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2} \tilde{K}_{XX}^{1/2}\right]^{-1} y \\ &= y^\top \tilde{K}_{XX}^{-1/2} \left[I - I + \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right]^{-1} \tilde{K}_{XX}^{-1/2} y \\ &= y^\top \tilde{K}_{XX}^{-1/2} \left[I - \left(I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right)\right]^{-1} \tilde{K}_{XX}^{-1/2} y \end{aligned} \qquad \text{(S8)}$$

this last form allow us to use a Neumann series to expand the inner inverse matrix. Hence, we have that

$$\left[I - \left(I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right)\right]^{-1} = \sum_{t=0}^{\infty} \left(I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right)^t$$

Combining this with Eq. (S8), we have that

$$\mathop{\mathbb{E}}_{P(\omega)}\left[y^\top \hat{K}_J^{-1} y\right] = y^\top \tilde{K}_{XX}^{-1} y + \sum_{t=2}^{\infty} y^\top \tilde{K}_{XX}^{-1/2} \mathop{\mathbb{E}}_{P(\omega)}\left(I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right)^t \tilde{K}_{XX}^{-1/2} y \qquad \text{(S9)}$$

where the first term of the series ($t = 0$) is the data-fit term being approximated and the second term of the series cancels out ($t = 1$). Following the analysis in (Nowozin, 2018) we translate the central moments of the random variable $\tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}$ to the central moments of $\tilde{K}_{XX}^{-1/2} \varphi(\omega) \varphi(\omega)^\top \tilde{K}_{XX}^{-1/2}$ denoted as α_i for $i \geq 2$. Since the term ($t = 2$) will dominate, we will only focus on it (as the others would be of a higher order in J, see (Angelova, 2012)). We have that

$$\mathop{\mathbb{E}}_{P(\omega)}\left(I - \tilde{K}_{XX}^{-1/2} \hat{K}_J \tilde{K}_{XX}^{-1/2}\right)^2 = \frac{\alpha_2}{J} \qquad \text{(S10)}$$

Therefore, incorporating the previous result into Eq. (S9), allows us to conclude that

$$\mathop{\mathbb{E}}_{P(\omega)}\left[y^\top \hat{K}_J^{-1} y\right] - y^\top \tilde{K}_{XX}^{-1} y = O(1/J)$$

For the model complexity term $\log|K_{XX}|$ we follow a similar procedure as before. For a fixed

$$\log|K_J| = \log|K_{XX}| - \log|K_{XX}| + \log|K_J|$$
$$= \log|K_{XX}| - \log|I + K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2}|$$

Focusing on the last term we have that

$$\log|I + K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2}|$$
$$= \mathrm{tr}\log(I + K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2})$$

We can rewrite the term in the R.H.S as follows

$$\log(I + K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2})$$
$$= \sum_{t=1}^{\infty}\frac{1}{t}(-1)^{t+1}(K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2})^t$$

Again, following the analysis of (Nowozin, 2018), we can express the central moments of the random variable $K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2}$ to the central moments of $K_{XX}^{-1/2}(\hat{\theta}(\omega) - \theta)(\hat{\theta}(\omega) - \theta)^\top K_{XX}^{-1/2}$ denoted as α_i for $i \geq 2$. Also, as explained before, the dominant term will be ($t = 2$), therefore we have that thus, using Eq. (S10) we have that

$$\mathrm{tr}\left(\mathbb{E}_{P(\omega)}\left[\log(I + K_{XX}^{-1/2}(K_J - K_{XX})K_{XX}^{-1/2})\right]\right)$$
$$= O(1/J)$$

Therefore, we can conclude that

$$\mathbb{E}_{P(\omega)}\left[\log|K_J| - \log|K_{XX}|\right] = O(1/J).$$

□

B. Further Derivations of Randomized Truncation Estimators

Here we prove the unbiasedness of both Russian Roulette (RR) Eq. (6) and Single Sample (SS) Eq. (7) estimators. Recall from Sec. 2.3 that we wish to estimate the expensive

$$= \sum_{j=1}^{H} \quad ; \quad H \sim N[\{1\} \quad ;$$

by computing just the first J H terms, where $J \in \{1, \ldots, H\}$ is randomly drawn from the truncation distribution $P(J = J)$. RR and SS estimators, denoted as , offer two strategies for up-weighting the J computed terms, which, as we will prove below, yield an unbiased estimator $\mathbb{E}_J[\ ] = $.

B.1. Unbiasedness of the RR Estimator

The RR estimator of Eq. (6) is unbiased, i.e. $\mathbb{E}_{J,\ J} = $.

Proof.

$$\mathbb{E}_{J\ J} = \mathbb{E}_J\left[\sum_{j=1}^{J}\frac{\ j}{P(J\ \ j)}\right]$$
$$= \mathbb{E}_J\left[\sum_{j=1}^{H}\frac{\ j}{P(J\ \ j)}\ I_{j\ \ J}\right]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J\ \ j)}\mathbb{E}_J[I_{j\ \ J}]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J\ \ j)}\left[\sum_{J=1}^{H}P(J = J)\ I_{J\ \ j}\right]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J\ \ j)}P(J\ \ j) =$$

□

B.2. Unbiasedness of the SS Estimator

The SS estimator of Eq. (7) is unbiased, i.e. $\mathbb{E}_{J,\ J} = $.

Proof.

$$\mathbb{E}_{J\ J} = \mathbb{E}_J\left[\sum_{j=1}^{H}\frac{\ j}{P(J = j)}\ I_J\, 1_{j=J}\right]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J = j)}\mathbb{E}_J[I_{j=J}]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J = j)}\left[\sum_{J=1}^{H}P(J = J)\ I_{J=j}\right]$$
$$= \sum_{j=1}^{H}\frac{\ j}{P(J = j)}P(J = j) =$$

□

B.3. Minimizing the Variance of the SS Estimator

Below we will derive the optimal distribution that minimizes the variance of our SS estimator. Note that for a given truncation distribution, we have that

$$V_J\left[\ \right] = \sum_{j=1}^{H}\frac{\ j^2}{P(J = j)^2}V_J(I_{J=j})$$
$$= \sum_{j=1}^{H}\ \ j^2\ \frac{1 - P(J = j)}{P(J = j)}$$

since $\mathbb{I}_{J=j}$ is a Bernoulli random variable with probability $P(J=j)$, we can plug-in its variance and derive the second equality. Hence, to find the truncation distribution that minimizes the variance of the SS estimator we can solve the following constraint optimization problem.

$$\min_{p} \sum_{j=1}^{H} \frac{1}{p_j} \left(\frac{\alpha_j}{\beta_j}\right)^2 - 1 \quad \text{s.t.} \quad \sum_{j=1}^{H} p_j = 1; \quad p_j \geq 0$$

where p_j is acting as a shorthand for $P(J=j)$. The Lagrangian of the problem is

$$L(p, \lambda) = \sum_{j=1}^{H} \frac{1}{p_j} \left(\frac{\alpha_j}{\beta_j}\right)^2 + \lambda \left(1 - \sum_{j=1}^{H} p_j\right)$$

where we can ignore the nonnegativity constraints as long as $\alpha_j > 0$ (see solution below). Hence, the first order conditions dictate that

$$\frac{\partial L}{\partial p}\left(p_j^\star, \lambda^\star\right) = -\frac{\left(\frac{\alpha_j}{\beta_j}\right)^2}{p_j^{\star 2}} + \lambda^\star = 0$$

therefore, if we take the ratio of the probabilities with respect to the first we get that $p_j^\star = \frac{\alpha_j}{\alpha_1} p_1^\star$. If we substitute this expression in the equality constraint we get that

$$P^\star(J=j) = \frac{\frac{\alpha_j}{\beta_j}}{\sum_{i=1}^{H} \frac{\alpha_i}{\beta_i}} \Big/ \beta_j \qquad \text{(S11)}$$

We emphasize that this result is a guide for practical choices of the truncation distribution. It is impractical to compute it as it will require evaluating all the α_j for $j = 1, \ldots, H$. However, if we posses an estimate or a theoretical bound on rate of decay of each α_j then our unnormalized truncation distribution should also decay at this rate to minimize variance.

B.4. SS estimator as Importance Sampling

Here we derive the SS estimator by importance sampling the quantity $\sum_{J=1}^{H} \alpha_J$ with $P(J=J)$ as our proposal distribution.

$$= \sum_{J=1}^{H} \alpha_J$$

Next, re-write the summation above as an expectation over the discrete uniform distribution $J \sim U[1, H] = \frac{1}{H}, \forall J$:

$$= H \sum_{J=1}^{H} \frac{1}{H} \alpha_J;$$

We now introduce an alternative proposal distribution $P(J=J)$:

$$= H \sum_{J=1}^{H} \frac{\alpha_J}{H P(J=J)} P(J=J)$$

$$= H \mathop{\mathbb{E}}_{J \sim P(J=J)} \frac{\alpha_J}{P(J=J)};$$

Approximating the final expectation using a single Monte Carlo sample results in the SS estimator.

C. Estimating the Marginal Log Likelihood from RR-CG

Recall from Sec. 4.1 that we use the Russian Roulette estimator in conjunction with conjugate gradients to compute an unbiased (stochastic) gradient of the log marginal likelihood. In this section, we briefly describe how to obtain an unbiased estimate of the log marginal likelihood itself.

The $y^\top K_{XX}^{-1} y$ term can be estimated directly from the $K_{XX}^{-1} y$ solve in Eq. (16). The $\log|K_{XX}|$ term is less straightforward, as the estimate from the CG byproducts (Eq. 8) isn't readily expressed as a summation. Instead, we apply the Russian Roulette estimator to the following telescoping series:

$$\log|K_{XX}| = \|z\|^2 e_1^T \log T_z^{(N)} e_1 \qquad \text{(S12)}$$

$$= \|z\|^2 e_1^T \log T_z^{(1)} e_1 + \sum_{j=2}^{N} \Delta_j;$$

where $\Delta_j = \|z\|^2 e_1^T \log T_z^{(j)} - \log T_z^{(j-1)} e_1$. We can therefore apply the Russian Roulette estimator with truncation $P(J)$ to Eq. (S12). This telescoping series can be expensive to compute. Since it is unnecessary for gradient-based optimization, we introduce it only as a tool to analyze the RR-CG log marginal likelihood.

D. Optimal Truncation Distributions

D.1. Proof of Theorem 3

Proof. For Thm. 3 we have to combine the results of Thm. 1 with the optimal distribution for RR that is derived in Beatson & Adams (2019). We will only focus on v_J which that approximates $\log|K_{XX}|$ since the procedure is analogous for u_J which approximates $y^\top K_{XX}^{-1} y$. Beatson & Adams (2019, Theorem 5.4) state that the optimal RR truncation distribution that maximizes the ROE is

$$P(J=j) \propto \sqrt{\frac{\mathbb{E}[\|v_j\|^2]}{c(j) - c(j-1)}}$$

where $c(j)$ is the computational cost of evaluating v_j which is the j-th term being approximated through Russian

Roulette. In this case, the v_j are nonrandom and single-valued and the cost per iteration is constant with respect to j. Hence, we can conclude that

$$P(J \geq j)/v_j = O(C^{2j})$$

which implies that $P(J \geq j)/(1 - C^{2j})$ since the derivative of an exponential function is of the same form we have that $P(J = j)/O(C^{2j})$. □

D.2. Proof of Theorem 4

The strategy for Thm. 4 is to relate the kernel approximation using $J+1$ random Fourier features to the kernel approximation using J features for the two components of the log marginal likelihood: the data-fit term $y^\top (\hat{K}_{XX}^{-1} + \sigma^2 I)^{-1} y$ and the model complexity term $\log|\hat{K}_{XX}^{-1} + \sigma^2 I|$. For $y^\top (\hat{K}_{XX}^{-1} + \sigma^2 I)^{-1} y$ we create this connection through the Sherman-Morisson formula and for the $\log|\hat{K}_{XX} + \sigma^2 I|$ we use the matrix determinant lemma. Throughout these proofs, we will require some results regarding positive definite matrices that we add as remarks below. Before stating those remarks, we will first introduce some auxiliary notation.

$$K_J := \frac{1}{J}\sum_{j=1}^{J} \phi_j \phi_j^\top$$

$$(J+1)K_{J+1} = JK_J + \phi_{J+1}\phi_{J+1}^\top$$

Note the difference between the $\hat{K}_J$ term used throughout the paper which includes the $\sigma^2 I$ against the K_J term which only includes the RFF features. Moreover, to reduce clutter we define the following matrix

$$W := \frac{J}{J+1}K_J + \sigma^2 I$$

whose use will become evident throughout the proof.

Remark 1. Given a positive definite matrix A we have that for any $a \in (0,1)$ and any $b > 0$

$$\|aA + bI\|_2^2 \leq \|A + bI\|_2^2$$

If we express $A = VDV^\top$, where V is an orthogonal matrix and D is a diagonal matrix containing the positive eigenvalues of A, then we have that $aA + bI = V(aD + bI)V^\top$. This implies that the eigenvalues of $aA + bI$ are larger than those of $aA + bI$. An immediate consequence of this remark is that $\|W\|_2^2 \leq \|K_J + \sigma^2 I\|_2^2$ or that $\|W^{-1}\|_2^2 \leq (\lambda_1 + \sigma^2)^{-2} \leq \sigma^{-4}$ where λ_1 contains the positive eigenvalues of K_J.

Remark 2. Given a positive definite matrix A we have that for any $a \in (0,1)$, any $b > 0$ and any vector x

$$x^\top (aA + bI)^{-1} x \geq x^\top (A + bI)^{-1} x \geq b^{-1} x^\top x$$

This remark follows by using the same diagonal decomposition as the one used in the previous remark. Hence, by noting that the eigenvalues of $(bI)^{-1}$ are larger than those of $(aA + bI)^{-1}$ and this last matrix has larger eigenvalues than $(A + bI)^{-1}$ then the result follows.

Proof of Theorem 4. We start by showing the rate of decay for the ϵ_j involving the data-fit terms. Note that we can express

$$\begin{aligned} & y^\top (K_{J+1} + \sigma^2 I)^{-1} y \\ &= y^\top \left(\frac{J}{J+1}K_J + \sigma^2 I + \frac{\phi_{J+1}\phi_{J+1}^\top}{J+1}\right)^{-1} y \\ &= y^\top \left(W + \frac{\phi_{J+1}\phi_{J+1}^\top}{J+1}\right)^{-1} y \end{aligned} \tag{S13}$$

Applying Sherman-Morisson formula to the inverse of the R.H.S results in

$$W^{-1} - \frac{\phi_{J+1}^\top W^{-1} W^{-1} \phi_{J+1}}{(J+1) + \phi_{J+1}^\top W^{-1} \phi_{J+1}}$$

where the symmetry of W allows us to express the numerator as above. Substituting the previous result into Eq. (S13) and rearranging terms allows us to conclude that

$$\begin{aligned} & y^\top (\sigma^2 I + K_J)^{-1} y - y^\top (\sigma^2 I + K_{J+1})^{-1} y \\ & y^\top \left(\sigma^2 I + \frac{J}{J+1}K_J\right)^{-1} y - y^\top (\sigma^2 I + K_{J+1})^{-1} y \\ &= \frac{\phi_{J+1}^\top W^{-1} y^2}{(J+1) + \phi_{J+1}^\top W^{-1} \phi_{J+1}} \\ &\leq \frac{\phi_{J+1}^\top W^{-1} y^2}{J+1} \\ &\leq \frac{\|\phi_{J+1}\|_2^2 \|W^{-1} y\|_2^2}{J+1} \\ &\leq \frac{\|\phi_{J+1}\|_2^2 \sigma^{-4} \|y\|_2^2}{J+1} \end{aligned} \tag{S14}$$

where the first inequality follows from Remark 2, the second inequality occurs since $\phi_{J+1}^\top W^{-1} \phi_{J+1} > 0$, the third inequality results from applying Cauchy-Schwarz and the fourth stems from Remark 1. Finally, taking expectations in Eq. (S14) we can conclude that

$$\begin{aligned} & \mathbb{E}_{P(\omega)}\left[y^\top (\sigma^2 I + K_J)^{-1} y\right] \\ & \mathbb{E}_{P(\omega)}\left[y^\top (\sigma^2 I + K_{J+1})^{-1} y\right] \\ &= O(1/J). \end{aligned}$$

We now move into the rate of decay of the Δ_j terms involving the model complexity terms. Note that we can express

$$\mathsf{K}_{J+1} + \sigma^2 \mathsf{I} \;=\; \frac{J}{J+1}\mathsf{K}_J + \sigma^2 \mathsf{I} + \frac{\overset{>}{J+1}\;\overset{>}{J+1}}{J+1}$$

then by using the matrix determinant lemma we have that

$$\begin{aligned}
\mathsf{K}_{J+1} + \sigma^2 \mathsf{I} \;&=\; 1 + \frac{1}{J+1}\;\overset{>}{J+1}\mathsf{W}^{\;-1}\;J+1 \quad |\mathsf{W}| \\
&\leq\; 1 + \frac{1}{J+1}\;\overset{>}{J+1}\mathsf{W}^{\;-1}\;J+1 \quad \mathsf{K}_J + \sigma^2 \mathsf{I} \\
&\leq\; 1 + \frac{\sigma^{\;2}}{J+1}\;\overset{>}{J+1}\;J+1 \quad \mathsf{K}_J + \sigma^2 \mathsf{I}
\end{aligned} \tag{S15}$$

where the first inequality follows from Remark 1 and from noting that the determinant is equivalent to the product of the eigenvalues of the matrix. The second inequality follows from Remark 2. Now, we will take the logarithms and the expectations of Eq. (S15). We will first focus on the first term in the RHS. By using a Taylor expansion of the logarithm up to a second term we have that

$$\begin{aligned}
&\underset{\mathsf{P}(!)}{\mathsf{E}}\;\log\;\left(1 + \frac{\sigma^{\;2}}{J+1}\;\overset{>}{J+1}\;J+1\right) \\
&= \frac{\sigma^{\;2}\,\mathsf{E}_{\mathsf{P}(!)}\;\overset{>}{J+1}\;J+1}{J+1} + \mathcal{O}\;\left(1/J^2\right)
\end{aligned}$$

therefore, substituting the previous result into Eq. (S15)

$$\begin{aligned}
&\underset{\mathsf{P}(!)}{\mathsf{E}}\;\log\;\mathsf{K}_{J+1} + \sigma^2 \mathsf{I} \;-\; \underset{\mathsf{P}(!)}{\mathsf{E}}\;\log\;\mathsf{K}_J + \sigma^2 \mathsf{I} \\
&= \frac{\sigma^{\;2}\,\mathsf{E}_{\mathsf{P}(!)}\;\overset{>}{J+1}\;J+1}{J+1} + \mathcal{O}\;\left(1/J^2\right) \\
&= \mathcal{O}\left(1/J\right)
\end{aligned}$$

which concludes the analysis of the rate of decay of the log determinant terms. Thus, since each rate of decay if $\mathcal{O}\left(1/J\right)$, then following Eq. (S11) we have that the truncation distribution that minimizes the variance is

$$\mathsf{P}\;\;(J) \propto \Delta_J = \mathcal{O}\left(1/J\right).$$

□

E. Experiment Details.

We provide the experiment details for the predictive performance experiments in Sec. 5.

Experiment setup. To optimize the hyperparameters of the CG, RR-CG and Cholesky models, we use an Adam optimizer with learning rate = 0.01 and a MultiStepLR scheduler dropping the learning rate by a factor of 10 at the 50%, 70% and 90% of the optimization iterations. We run the optimization for 1500, 800 and 300 iterations on small (PoleTele, Elevators and Bike), medium (Kin40K, Protein, KEGG and KEGGU) and large (3DRoad) datasets, respectively. The number of iterations for CG and the expected number of iterations for RR-CG are both set to 100. The latter is achieved by using the truncation distribution from Eq. (17) with $\lambda = 0.05$ and $J_{\min} = 80$.

For CG and RR-CG, we use the rank-5 pivoted Cholesky preconditioner of Gardner et al. (2018). To reduce the number of optimization steps needed for the 3DRoad dataset, we initialize the hyperparameters to those found with the (biased) sgGP method. During evaluation, we compute the predictive mean using $1{,}000$ iterations of CG. Predictive variances are estimated using the rank-100 Lanczos approximation of Pleiss et al. (2018).

For RFF we use 1,000 random Fourier features across all experiments. In terms of optimization, we use an Adam optimizer with learning rate = 0.005 for KEGG, 0.001 for KEGGU and 0.01 for the remaining datasets. We also use a MultiStep scheduler that activates at 85%, 90%, 95% of the optimization iterations with a decay rate of 0.5 and also take a total of 500 optimization iterations on all the datasets.

In all the SVGP models we use $1{,}024$ inducing points, with a full-rank multivariate Gaussian variational distribution. Hyperparameters and variational parameters are jointly optimized for 300 epochs using minibatches of size $1{,}024$. As with the other baselines, we use the an Adam optimizer with a learning rate of 0.01, dropping the learning rate by a factor of 10 after 50%, 70% and 90% of the optimization iterations.

For sgGP, we train using minibatches of 16 data points. As suggested by Chen et al. (2020), the minibatches are constructed by sampling one training data point and selecting its 15 nearest neighbors. To accelerate optimization, we accumulate the gradients of $1{,}024$ minibatches before performing an optimization step (these $1{,}024$ minibatch updates can be performed in parallel, enabling GPU acceleration). We optimize the models for 300 epochs, using the same learning rate and scheduler as with SVGP. During evaluation, we use the same procedure as for CC/RR-CG ($1{,}000$ iterations of CG, rank-100 Lanczos variance estimates).

For POE, we divide the training dataset into disjoint subsets, each with $M = 1024$ data points. If N is not divisible by 1024, we pad the final subset with elements from the first subset. We train independent GP with independent hyperparameters on each of the subsets using the same optimization

Dataset	n	Cholesky	POE	RFF	RMSE SVGP	sgGP	CG	RR-CG
PolTele	9.6K	$.112 \pm .002$	$.174 \pm .006$	$\mathbf{.106 \pm .000}$	$.150 \pm .000$	$.128 \pm .001$	$.119 \pm .002$	$.112 \pm .002$
Elevators	10.6K	$\mathbf{.360 \pm .006}$	$.379 \pm .005$	$\mathbf{.365 \pm .001}$	$.376 \pm .006$	$\mathbf{.362 \pm .006}$	$\mathbf{.360 \pm .006}$	$\mathbf{.360 \pm .006}$
Bike	11.1K	$\mathbf{.035 \pm .003}$	$.071 \pm .002$	$.063 \pm .001$	$.045 \pm .002$	$1.044 \pm .334$	$\mathbf{.040 \pm .005}$	$\mathbf{.035 \pm .002}$
Kin40K	25.6K	—	$.248 \pm .001$	$\mathbf{.074 \pm .000}$	$.152 \pm .001$	$.081 \pm .000$	$.093 \pm .000$	$.091 \pm .001$
Protein	25.6K	—	$.715 \pm .007$	$\mathbf{.547 \pm .001}$	$.664 \pm .008$	$.562 \pm .005$	$\mathbf{.541 \pm .008}$	$\mathbf{.541 \pm .008}$
KEGG	31.2K	—	$.097 \pm .004$	$.101 \pm .001$	$\mathbf{.088 \pm .002}$	$\mathbf{.089 \pm .002}$	$.195 \pm .064$	$\mathbf{.087 \pm .003}$
KEGGU	40.7K	—	$.125 \pm .001$	$.129 \pm .001$	$.122 \pm .001$	$.123 \pm .001$	$\mathbf{.120 \pm .000}$	$\mathbf{.120 \pm .000}$
3DRoad	278K	—	$.675 \pm .001$	$.348 \pm .001$	$.439 \pm .002$	$.285 \pm .003$	$.202 \pm .003$	$\mathbf{.114 \pm .013}$

Dataset	n	Cholesky	POE	RFF	NLL SVGP	sgGP	CG	RR-CG
PolTele	9.6K	$-.464 \pm .006$	$-.252 \pm .010$	$-.159 \pm .005$	$-.442 \pm .004$	$\mathbf{-.480 \pm .004}$	$-.354 \pm .003$	$-.458 \pm .006$
Elevators	10.6K	$\mathbf{.425 \pm .013}$	$.469 \pm .016$	$.780 \pm .006$	$\mathbf{.442 \pm .015}$	$\mathbf{.426 \pm .015}$	$\mathbf{.429 \pm .012}$	$\mathbf{.425 \pm .013}$
Bike	11.1K	$-.984 \pm .018$	$-.799 \pm .010$	$.099 \pm .030$	$\mathbf{-1.514 \pm .024}$	85.715 ± 49.088	$-.971 \pm .008$	$-.982 \pm .018$
Kin40K	25.6K	—	$.464 \pm .002$	$1.407 \pm .004$	$\mathbf{-.410 \pm .003}$	$.427 \pm .001$	$.468 \pm .003$	$.449 \pm .013$
Protein	25.6K	—	$1.105 \pm .008$	$1.163 \pm .003$	$1.014 \pm .012$	$.951 \pm .004$	$\mathbf{.934 \pm .006}$	$\mathbf{.934 \pm .006}$
KEGG	31.2K	—	$-.874 \pm .011$	6.311 ± 2.323	$\mathbf{-1.022 \pm .023}$	$\mathbf{-.981 \pm .033}$	$.415 \pm .653$	$-.884 \pm .009$
KEGGU	40.7K	—	$-.636 \pm .002$	$4.000 \pm .303$	$\mathbf{-.685 \pm .005}$	$-.668 \pm .005$	$-.637 \pm .006$	$-.650 \pm .004$
3DRoad	278K	—	$1.031 \pm .002$	$1.317 \pm .004$	$\mathbf{.601 \pm .004}$	$.831 \pm .000$	$\mathbf{.613 \pm .010}$	$.776 \pm .030$

Table S1. Root-mean-square-error (RMSE) and negative log-likelihood (NLL) of exact GPs using CG, RRCG and other baselines on UCI regression datasets using a constant prior mean and a RBF kernel with independent lengthscale for each dimension. All trials were averaged over 3 trials with different splits. N and d are the size and dimensionality of the training dataset, respectively.

Dataset	n	Cholesky	POE	RFF	Training time (m) SVGP	sgGP	CG	RR-CG
PolTele	9.6K	$22.417 \pm .035$	$1.167 \pm .006$	$.464 \pm .002$	$3.862 \pm .018$	$.629 \pm .006$	$12.793 \pm .111$	$14.968 \pm .258$
Elevators	10.6K	$30.617 \pm .016$	$1.051 \pm .008$	$.503 \pm .002$	$4.236 \pm .023$	$.696 \pm .001$	$14.443 \pm .080$	$16.519 \pm .152$
Bike	11.1K	$34.991 \pm .016$	$1.364 \pm .012$	$.476 \pm .014$	$4.261 \pm .023$	$.691 \pm .002$	$11.634 \pm .131$	$13.669 \pm .080$
Kin40K	25.6K	—	$.930 \pm .006$	$.772 \pm .008$	$9.597 \pm .043$	$1.559 \pm .010$	$11.345 \pm .073$	$12.823 \pm .114$
Protein	25.6K	—	$.851 \pm .003$	$.716 \pm .008$	$11.115 \pm .033$	$1.867 \pm .007$	$10.507 \pm .044$	$12.142 \pm .052$
KEGG	31.2K	—	$1.135 \pm .003$	$.682 \pm .002$	$11.881 \pm .058$	$1.786 \pm .009$	$22.780 \pm .021$	$25.390 \pm .089$
KEGGU	40.7K	—	$1.140 \pm .005$	$.835 \pm .001$	$15.281 \pm .070$	$2.572 \pm .029$	$34.396 \pm .026$	$37.825 \pm .064$
3DRoad	278K	—	$2.176 \pm .022$	$6.089 \pm .031$	$104.164 \pm .294$	$22.615 \pm .147$	145.396 ± 1.373	$158.657 \pm .554$

Table S2. Total training time (in minutes) of exact GPs using CG, RRCG and other baselines on UCI regression datasets (see the number of optimization iterations for each method in experiment setup). All trials were averaged over 3 trials with different splits.

procedure as Cholesky models. Following Deisenroth & Ng (2015, Eqs. 11 and 12), the posterior distribution for a test input $\mathbf{x}$ is Gaussian with mean $\mu_{\text{POE}}(\mathbf{x})$ and variance $\sigma^2_{\text{POE}}(\mathbf{x})$:

$$\mu_{\text{POE}}(\mathbf{x}) = \frac{\sigma^2_{\text{POE}}(\mathbf{x})}{K} \sum_{k=1}^{K} \sigma_k^{-2}(\mathbf{x})\mu_k(\mathbf{x}),$$

$$\sigma^{-2}_{\text{POE}}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} \sigma_k^{-2}(\mathbf{x}),$$

where K is the number of independent GP experts and $\mu_k(\cdot)$ and $\sigma_k^2(\cdot)$ are the posterior mean and variance of each expert model.

Full tables. In Table S1, we report the RMSE and NLL numbers which are used to plot Fig. 6 in the paper, and in Table S2 we report the corresponding training time.

F. Additional Experiments

F.1. Predictive Performance with Different CG iterations

Here we include an additional experiment to show that early truncated CG can be detrimental to GP learning while RR-CG remains robust to the expected truncation number.

We conduct GP learning with the CG, RR-CG and Cholesky methods in the Elevators dataset. We vary the number of (expected) iterations for CG and RR-CG from 20 to 100 and plot the corresponding RMSE / NLL in Fig. S1. From the figure, we conclude that: 1) early truncation of the CG algorithm impedes GP optimization and leads to poor predictions. This problem lessens as we increase the number of CG iterations from 20 to 100. 2) The RR-CG model is robust to the expected truncation number, as it keeps comparable RMSE and NLL values to the Cholesky model under different expected truncation numbers. This experiment can