Inverse-Free Sparse Variational Gaussian Processes

Stefano Cortinovis University of Oxford
Laurence Aitchison University of Bristol
Stefanos Eleftheriadis No affiliation
Mark van der Wilk University of Oxford

Abstract

Gaussian processes (GPs) offer appealing properties but are costly to train at scale. Sparse variational GP (SVGP) approximations reduce cost yet still rely on Cholesky decompositions of kernel matrices, ill-suited to low-precision, massively parallel hardware. While one can construct valid variational bounds that rely only on matrix multiplications (matmuls) via an auxiliary matrix parameter, optimising them with off-the-shelf first-order methods is challenging. We make the inverse-free approach practical by proposing a better-conditioned bound and deriving a matmul-only natural-gradient update for the auxiliary parameter, markedly improving stability and convergence. We further provide simple heuristics, such as step-size schedules and stopping criteria, that make the overall optimisation routine fit seamlessly into existing workflows. Across regression and classification benchmarks, we demonstrate that our method **1**) serves as a drop-in replacement in SVGP-based models (e.g., deep GPs), **2**) recovers similar performance to traditional methods, and **3**) can be faster than baselines when well tuned.

1 INTRODUCTION

Gaussian processes (GPs) (Williams and Rasmussen, 2006) are flexible Bayesian priors over functions, valued for uncertainty that takes into account infinite basis functions, and automatic model selection. These benefits come with a cubic training cost $\mathcal{O}(N^3)$ in the number of data points N, which limits their use at scale. Sparse variational approximations (Titsias, 2009) replace full GP inference with a variational posterior over M pseudo-inputs, called *inducing points*, reducing the cost of full-batch training to $\mathcal{O}(NM^2 + M^3)$, so that the only cubic dependence is on M rather than N. Stochastic optimisation (Hensman et al., 2013) further reduces the per-iteration cost to $\mathcal{O}(BM^2 + M^3)$ with mini-batches of size $B \ll N$.

While these advances enable arbitrarily accurate approximations with $M \ll N$ (Burt et al., 2019, 2020), they still require computing the inverse and determinant of an $M \times M$ kernel matrix. In practice, these are obtained via Cholesky decompositions, which rely on inherently sequential linear algebra and high-precision arithmetic, poorly matched to modern accelerators optimised for low-precision, massively parallel matrix multiplications (matmuls). To avoid decompositions, van der Wilk et al. (2020b, 2022) derived an inverse-free variational bound that replaces them with matmuls by introducing an auxiliary parameter $\mathbf{T} \in \mathbb{R}^{M \times M}$ whose optimum recovers the required inverse, but this objective can be difficult to optimise with off-the-shelf first-order methods such as Adam (Kingma, 2014), leading to instability and slow convergence (Fig. 1).

In this work, we make the inverse-free approach practical (see Fig. 1). First, we show that, even with optimally tuned $\mathbf{T}$, a preconditioner on a variational parameter is needed to match the performance of standard SVGP methods. Second, we derive a natural-gradient (NG) update for $\mathbf{T}$ that uses only matmuls, substantially improving stability and convergence. Finally, we discuss a number of strategies, such as simple schedules and stopping rules, that make the $\mathbf{T}$ updates efficient and automatic. The resulting SVGP objective **1**) can be *evaluated* in quadratic time, **2**) serves as a drop-in replacement within complex SVGP-based models, and **3**) trains stably using only matmuls. While *training* remains cubic in M due to the NG update for $\mathbf{T}$, the overall procedure is well-suited to parallelisation and low-precision arithmetic, as it only involves matmuls. We evaluate our method on regression and classification benchmarks, where it achieves comparable performance to baselines that rely on Cholesky

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

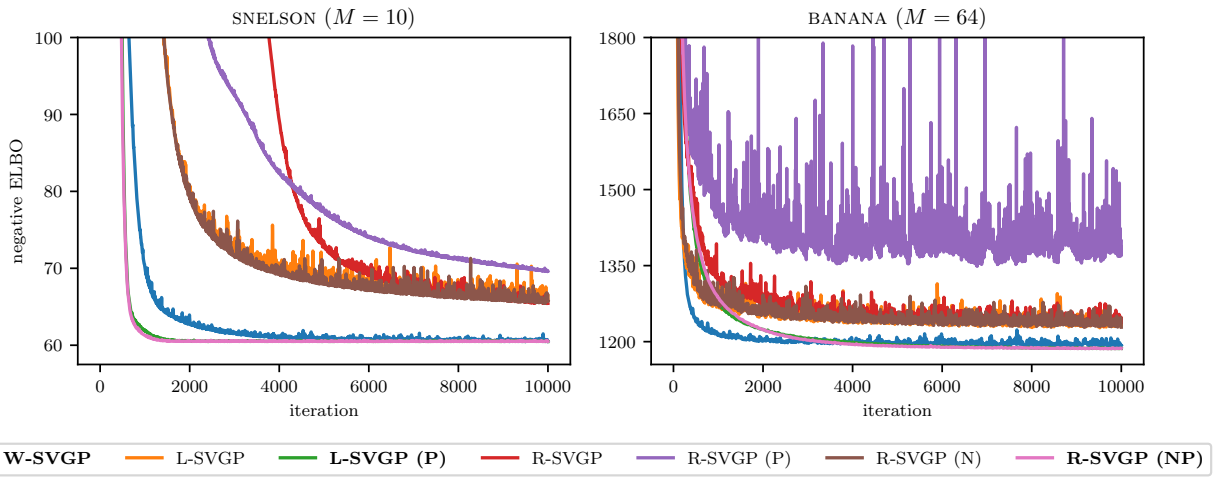

Figure 1: Loss traces on SNELSON and BANANA datasets. The N and P suffixes refer to the use of NG updates (Section 3.1) and inducing mean preconditioning (Section 3.2), respectively. R-SVGP (NP) is the only inverse-free variant that matches the performance of W-SVGP and L-SVGP (P), which use Cholesky decompositions.

decompositions.

The remainder of the paper is organised as follows. Section 2 reviews SVGP parameterisations and the inverse-free bound of van der Wilk et al. (2022). Section 3 presents our **T**-tailored updates, preconditioning, and practical training strategies. Section 4 evaluates the proposed method on a variety of benchmarks. Section 5 briefly surveys related work, and Section 6 discusses limitations and extensions.

2 SVGP PARAMETERISATIONS

We consider[1] the problem of learning a function $f : \mathbb{R}^D \to \mathbb{R}$ through Bayesian inference, with a prior $f(\cdot) \sim \mathcal{GP}(0, k_{\boldsymbol{\psi}}(\cdot,\cdot))$ and an arbitrary factorised likelihood with density $p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} p(y_n|f(\mathbf{x}_n);\boldsymbol{\eta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \boldsymbol{\eta}\}$ are model hyperparameters[2]. To make predictions, we need to approximate the maximum marginal likelihood hyperparameters $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{y}|\boldsymbol{\theta})$, and the posterior $p(f|\mathbf{y})$. To do so while avoiding the cost of full GP inference, one popular approach is to use sparse variational Gaussian processes (SVGPs) (Hensman et al., 2013), which select an approximate posterior distribution $q(f)$ by minimising the KL-divergence to the true posterior $p(f|\mathbf{y})$. The approximate posterior is constructed by conditioning the prior on $M \ll N$ inducing points (Quiñonero-Candela and Rasmussen, 2005), at input locations $\mathbf{Z} \in \mathbb{R}^{M\times D}$ and output values $\mathbf{u} = f(\mathbf{Z})$, resulting in a variational posterior $q(f) = \int p(f_{\neq \mathbf{u}}|\mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u}$ (see Matthews (2017); van der Wilk (2019) for further details). Following variational inference (Blei et al., 2017), $\mathrm{KL}[q(f)||p(f|\mathbf{y})]$ is minimised by maximising the evidence lower bound (ELBO) with respect to $\boldsymbol{\theta}$, $\mathbf{Z}$ and the parameters of $q(\mathbf{u})$. When $q(\mathbf{u})$ is chosen to be Gaussian, the ELBO becomes

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{\mathcal{N}(\mu_n,\sigma_n^2)}\left[\log p(y_n|f(\mathbf{x}_n))\right] - \mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right], \tag{1}$$

where μ_n and σ_n are, respectively, the predictive mean and variance at the input location $\mathbf{x}_n$. When optimised over mini-batches of size B, the SVGP ELBO (1) has time complexity $\mathcal{O}(BM^2 + M^3)$ and can be used to scale GP inference to large datasets under arbitrary likelihoods, mirroring stochastic optimisation in neural networks. However, unlike neural networks, the computation of μ_n, σ_n^2 and $\mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})]$ involves a Cholesky decomposition, which is ill-suited to modern deep-learning hardware. Furthermore, the choice of the parameterisation of the Gaussian distribution $q(\mathbf{u})$ can have a significant impact on the optimisation stability and the tightness of the ELBO. Below, we briefly review three common parameterisations, as well as their relationship with the inverse-free bound of van der Wilk et al. (2022), highlighting in red the terms that involve matrix decompositions.

Marginal Parameterisation (M-SVGP). The most common freeform $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$ results in

$$\mu_n = \mathbf{k}_{n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m}$$
$$\sigma_n^2 = k_{nn} - \mathbf{k}_{n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}n},$$

[1] The SVGP framework is presented for a shallow GP prior with scalar output, but the ideas presented here extend naturally to multi-output (van der Wilk et al., 2020a) and deep GP models (Salimbeni and Deisenroth, 2017).

[2] In the sequel, dependence of $k_{\boldsymbol{\psi}}(\cdot,\cdot)$ and $p(y_n|f(\mathbf{x_n};\boldsymbol{\eta}))$ on $\boldsymbol{\psi}$ and $\boldsymbol{\eta}$ is suppressed for brevity.

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] = \frac{1}{2}\left(\mathrm{tr}(\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}) + \mathbf{m}^\top\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m} - M + \log|\mathbf{K}_{\mathbf{uu}}| - \log|\mathbf{S}|\right),$$

where $\mathbf{K}_{\mathbf{uu}} = k(\mathbf{Z},\mathbf{Z})$, $\mathbf{k}_{\mathbf{u}n} = \mathbf{k}_{n\mathbf{u}}^\top = k(\mathbf{Z},\mathbf{x}_n)$, and $k_{nn} = k(\mathbf{x}_n,\mathbf{x}_n)$.

Whitened Parameterisation (W-SVGP). Hensman et al. (2015) propose to reparameterise $q(\mathbf{u}) = \mathcal{N}(\mathbf{m},\mathbf{S})$ with $\mathbf{m} = \mathbf{L}_{\mathbf{uu}}\tilde{\mathbf{m}}$ and $\mathbf{S} = \mathbf{L}_{\mathbf{uu}}\tilde{\mathbf{S}}\mathbf{L}_{\mathbf{uu}}^\top$, where $\mathbf{L}_{\mathbf{uu}} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}})$ is the Cholesky factor of $\mathbf{K}_{\mathbf{uu}}$ and $\tilde{\mathbf{S}}$ is PD. This results in

$$\begin{aligned}
\mu_n &= \mathbf{k}_{n\mathbf{u}}\mathbf{L}_{\mathbf{uu}}^{-\top}\tilde{\mathbf{m}}\\
\sigma_n^2 &= k_{nn} - \mathbf{k}_{n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{L}_{\mathbf{uu}}^{-\top}\tilde{\mathbf{S}}\mathbf{L}_{\mathbf{uu}}^{-1}\mathbf{k}_{\mathbf{u}n},\\
\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] &= \mathrm{KL}[\mathcal{N}(\tilde{\mathbf{m}},\tilde{\mathbf{S}})||\mathcal{N}(\mathbf{0},\mathbf{I})].
\end{aligned}$$

This is a form of whitening of the inducing variable $\mathbf{u}$, which often improves optimisation and is used as the default SVGP parameterisation in popular GP packages (e.g., Matthews et al., 2017).

Likelihood Parameterisation (L-SVGP). Panos et al. (2018) suggest to reparameterise $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\tilde{\mathbf{m}}$ and $\mathbf{S} = (\mathbf{K}_{\mathbf{uu}}^{-1} + \tilde{\mathbf{S}}^{-1})^{-1}$, where $\tilde{\mathbf{S}}$ is PD diagonal. This results in

$$\mu_n = \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{m}}, \quad \sigma_n^2 = k_{nn} - \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} =: \sigma_n^{2(\mathrm{L})}, \quad (2)$$

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] = \frac{1}{2}\left(-\mathrm{tr}(\tilde{\mathbf{K}}^{-1}\mathbf{K}_{\mathbf{uu}}) + \tilde{\mathbf{m}}^\top\mathbf{K}_{\mathbf{uu}}\tilde{\mathbf{m}} + \log|\tilde{\mathbf{K}}| - \log|\tilde{\mathbf{S}}|\right),$$

where $\tilde{\mathbf{K}} = \mathbf{K}_{\mathbf{uu}} + \tilde{\mathbf{S}}$. Inverting $\tilde{\mathbf{K}}$ instead of $\mathbf{K}$ is more stable without the need for *jitter* terms, due to having lower-bounded minimum eigenvalue thanks to $\tilde{\mathbf{S}}$. Moreover, further manipulations of the L-SVGP lead to an inverse-free bound, as we discuss next.

Inverse-Free Parameterisation (R-SVGP). Van der Wilk et al. (2022) develop an inverse-free bound for L-SVGP, starting by upper bounding the predictive variance (2) as

$$\sigma_n^{2(\mathrm{L})} \leq k_{nn} - \mathbf{k}_{n\mathbf{u}}(2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T})\mathbf{k}_{\mathbf{u}n} =: U_n, \quad (3)$$

where $\mathbf{T} \in \mathbb{R}^{M\times M}$, and with equality when $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$. Working backwards, they then find that, if the variance $\mathbf{S}$ of M-SVGP is reparameterised as $\mathbf{S} = \mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}(2\mathbf{T}-\mathbf{T}\tilde{\mathbf{K}}\mathbf{T})\mathbf{K}_{\mathbf{uu}}$, the predictive variance matches the upper bound, i.e., $\sigma_n^2 = U_n$. This improves over the previous inverse-free bound for M-SVGP (van der Wilk et al., 2020b) by allowing for a closed-form inverse-free bound for the KL term

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] \leq \frac{1}{2}\left(-\mathrm{tr}((2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T})\mathbf{K}_{\mathbf{uu}}) - M + \tilde{\mathbf{m}}^\top\mathbf{K}_{\mathbf{uu}}\tilde{\mathbf{m}} + \mathrm{tr}(\tilde{\mathbf{K}}\mathbf{T}) - \log|\mathbf{T}| - \log|\tilde{\mathbf{S}}|\right), \quad (4)$$

with equality when $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$. Putting all this together leads to a *relaxed* bound that **1**) is a valid ELBO, **2**) depends on the additional parameter $\mathbf{T}$, **3**) is free of matrix decompositions, **4**) recovers the L-SVGP solution when $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$.

Alternative Inverse-Free Parameterisations. While we do not pursue the idea further in this work, we note that the R-SVGP construction, whereby one exploits an inverse-free upper bound to the predictive variance, is more general and may be applied to parameterisations other than L-SVGP. For instance, starting from M-SVGP, reparameterise $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{m}}$ and $\mathbf{S} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{S}}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}$, where $\tilde{\mathbf{S}}$ is PD and $\mathbf{R}$ is lower triangular with positive diagonal. This results in

$$\mu_n = \mathbf{k}_{n\mathbf{u}}\mathbf{R}\tilde{\mathbf{m}}, \quad \sigma_n^2 = k_{nn} - \mathbf{k}_{n\mathbf{u}}(\mathbf{K}_{\mathbf{uu}}^{-1} - \mathbf{R}\tilde{\mathbf{S}}\mathbf{R}^\top)\mathbf{k}_{\mathbf{u}n}.$$

Similarly to Eq. (3), the predictive variance can be upper bounded with

$$\sigma_n^2 \leq k_{nn} + \mathbf{k}_{n\mathbf{u}}\mathbf{R}(\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R} - 2\mathbf{I} + \tilde{\mathbf{S}})\mathbf{R}^\top\mathbf{k}_{\mathbf{u}n},$$

with equality when $\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$. Working backwards, the upper bound above is equal to the predictive variance induced by the reparameterisation $\mathbf{S} = \mathbf{K}_{\mathbf{uu}} + \mathbf{K}_{\mathbf{uu}}\mathbf{R}(\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R} - 2\mathbf{I} + \tilde{\mathbf{S}})\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}$. The KL term implied by this choice of $\mathbf{m}$ and $\mathbf{S}$ admits the upper bound

$$\begin{aligned}
\mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \leq \frac{1}{2}\Big(&\tilde{\mathbf{m}}^\top\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{m}} - \log|\tilde{\mathbf{S}}|\\
&+\mathrm{tr}\left((\tilde{\mathbf{S}} - 3\mathbf{I} + \mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R})\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\right) + M\Big)\\
&+\mathrm{KL}\left[\mathcal{N}(\mathbf{0},\mathbf{R}\mathbf{R}^\top)||\mathcal{N}(\mathbf{0},\mathbf{K}_{\mathbf{uu}}^{-1})\right],
\end{aligned}$$

with equality when $\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$. The resulting ELBO depends on the additional parameter $\mathbf{R}$ and, at the optimum $\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$, recovers a whitened bound, as in W-SVGP, where the whitening is performed via $\mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$ rather than $\mathbf{L}_{\mathbf{uu}}^{-\top}$. While the bound above still contains an inverse in the inner KL term, we conjecture that the techniques developed in Section 3.1 for the optimisation of $\mathbf{T}$ in R-SVGP may be used to efficiently eliminate it from the bound, thereby making the required update for the rest of the parameters effectively inverse-free. A brief derivation of the bound above is provided in Section S4.

3 IMPROVED INVERSE-FREE BOUND

When $\mathbf{T}$ is optimised to $\tilde{\mathbf{K}}^{-1}$, the R-SVGP bound coincides with the L-SVGP bound. In practice, R-SVGP is competitive when **1**) $\mathbf{T}$ is kept close to its optimum throughout training, and **2**) L-SVGP itself can reach competitive performance. In van der Wilk et al.

(2022), $\mathbf{T}$ is parameterised via its Cholesky factor $\mathbf{L}$ with $\mathbf{T} = \mathbf{L}\mathbf{L}^\top$ and updated jointly with Adam, but this leads to poor optimisation even on simple datasets (Fig. 1). Section 3.1 addresses this with a tailored optimiser for $\mathbf{T}$. Moreover, we find that L-SVGP, as presented in Section 2, exhibits slower convergence than standard W-SVGP, often leading to worse predictive performance. Section 3.2 addresses this by introducing a preconditioner for the L-SVGP inducing mean, together with an inverse-free analogue for R-SVGP. Finally, Sections 3.3 and 3.4 provide practical strategies to make R-SVGP optimisation efficient and automatic.

3.1 Iterative Inversion via Natural Gradients

Our goal is to keep $\mathbf{T}$ close to $\tilde{\mathbf{K}}^{-1}$ without ever performing a matrix decomposition. We parameterise $\mathbf{T}$ via its Cholesky factor $\mathbf{L}$, so that the problem reduces to finding the Cholesky factor of $\tilde{\mathbf{K}}^{-1}$, $\mathbf{L}_{\tilde{\mathbf{K}}^{-1}}$.

Auxiliary Inversion Objective. For any symmetric PD matrix $\mathbf{A}$, the Cholesky factor of its inverse, $\mathbf{L}_{\mathbf{A}^{-1}}$, is the unique minimiser of the KL divergence between two zero-mean Gaussians:

$$\begin{aligned} \mathbf{L}_{\mathbf{A}^{-1}} &= \arg\min_{\mathbf{L}} \ell_{\mathbf{A}}(\mathbf{L}), \\ \ell_{\mathbf{A}}(\mathbf{L}) &:= \mathrm{KL}\big[\mathcal{N}(\mathbf{0}, \mathbf{L}\mathbf{L}^\top) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{A}^{-1})\big]. \end{aligned} \tag{5}$$

Natural Gradient Updates. Natural-gradient (NG) updates are well-suited to KL objectives (Amari, 1998). Let $\mathbf{F}$ be the Fisher information of $\mathcal{N}(\mathbf{0}, \mathbf{L}\mathbf{L}^\top)$ with respect to $\mathbf{L}$. The NG is $\tilde{\nabla}\ell_{\mathbf{A}} = \mathbf{F}^{-1}\nabla\ell_{\mathbf{A}}$ and, for Eq. (5), admits a closed form, derived in Section S1.

Proposition 1. *Let $\ell(\mathbf{L})$ be as in Eq. (5). Then,*

$$\tilde{\nabla}\ell_{\mathbf{A}} = \mathbf{L}\big[\mathrm{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \tfrac{1}{2}\big(\mathbf{I} + \mathrm{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L})\big)\big], \tag{6}$$

where $\mathrm{tril}(\cdot)$ *and* $\mathrm{diag}(\cdot)$ *return the lower triangular part and the diagonal of a matrix.*

Alternating Optimisation. We optimise the R-SVGP bound wrt $\mathbf{L}$ and $\boldsymbol{\xi} = \{\boldsymbol{\theta}, \mathbf{Z}, \tilde{\mathbf{m}}, \tilde{\mathbf{S}}\}$ by alternating:

1) **NG step:** given $\tilde{\mathbf{K}}$, update $\mathbf{L} \leftarrow \mathbf{L} - \gamma\,\tilde{\nabla}_{\mathbf{L}}\ell_{\tilde{\mathbf{K}}}$ for $t^\star$ steps;
2) **Adam step:** given $\mathbf{L}$, update $\boldsymbol{\xi}$ with an Adam step on the R-SVGP ELBO.

The NG expression (6) avoids matrix decompositions, keeping the procedure *inverse-free*. The update represents an $\mathbf{L}$-space version of the classical Newton-Schulz iteration for matrix inversion (Ben-Israel, 1965), which is itself a NG step on the objective (5) in $\mathbf{T}$-space. As a result, by standard Newton-type arguments (e.g., Kelley, 1995, Chapter 5), the auxiliary $\mathbf{L}$-iteration enjoys local quadratic convergence under regularity conditions.

3.2 Inducing Mean Preconditioning

We attribute the suboptimal optimisation of L-SVGP to the inducing-mean parameterisation $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\tilde{\mathbf{m}}$, and introduce a preconditioner $\mathbf{P} \in \mathbb{R}^{M\times M}$ via $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\mathbf{P}\tilde{\mathbf{m}}$. For L-SVGP, we take $\mathbf{P}^{(\mathrm{L})} = \tilde{\mathbf{K}}^{-1} = (\mathbf{K}_{\mathbf{uu}} + \tilde{\mathbf{S}})^{-1}$. When $\mathbf{X} = \mathbf{Z}$, this yields $\mu_n = \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{m}}$, matching the form of the GP posterior mean under a Gaussian likelihood (with optimal $\tilde{\mathbf{m}} = \mathbf{y}$, $\tilde{\mathbf{S}} = \sigma^2_{\mathrm{obs}}\mathbf{I}$ independent of other model parameters).

The same $\tilde{\mathbf{K}}^{-1}$ is incompatible with R-SVGP as it reintroduces inversions. Instead, we use the inverse-free analogue $\mathbf{P}^{(\mathrm{R})} = 2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T}$, which recovers the *same* ELBO gradients as preconditioned L-SVGP when $\mathbf{T}$ is held at $\tilde{\mathbf{K}}^{-1}$ (proof in Section S2):

Proposition 2. *Let $\tilde{\mathcal{L}}_{lsvgp}$ and $\tilde{\mathcal{L}}_{rsvgp}$ denote the L-SVGP and R-SVGP bounds with $\mathbf{P}^{(\mathrm{L})} = \tilde{\mathbf{K}}^{-1}$ and $\mathbf{P}^{(\mathrm{R})} = 2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T}$, respectively, and parameters $\boldsymbol{\xi} = \{\boldsymbol{\theta}, \mathbf{Z}, \tilde{\mathbf{m}}, \tilde{\mathbf{S}}\}$. Then*

$$\nabla_{\boldsymbol{\xi}}\tilde{\mathcal{L}}_{lsvgp} = \nabla_{\boldsymbol{\xi}}\tilde{\mathcal{L}}_{rsvgp}$$

when $\mathbf{T} = \mathrm{stopgrad}(\tilde{\mathbf{K}}^{-1})$. The equivalence does not hold for the naive choice $\mathbf{P} = \mathbf{T}$.

Summarising, substituting $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\mathbf{P}^{(\mathrm{R})}\tilde{\mathbf{m}}$ and $\mathbf{S} = \mathbf{K}_{\mathbf{uu}} - \mathbf{K}_{\mathbf{uu}}\mathbf{P}^{(\mathrm{R})}\mathbf{K}_{\mathbf{uu}}$ into the M-SVGP form and using the KL bound (4) yields

$$\begin{aligned} &\mu_n = \mathbf{k}_{n\mathbf{u}}\mathbf{P}^{(\mathrm{R})}\tilde{\mathbf{m}}, \quad \sigma_n^2 = k_{nn} - \mathbf{k}_{n\mathbf{u}}\mathbf{P}^{(\mathrm{R})}\mathbf{k}_{\mathbf{u}n}, \\ &\mathrm{KL}[q(\mathbf{u}) \,\|\, p(\mathbf{u})] \le \tfrac{1}{2}\Big(-\mathrm{tr}(\mathbf{P}^{(\mathrm{R})}\mathbf{K}_{\mathbf{uu}}) + \mathrm{tr}(\tilde{\mathbf{K}}\mathbf{T}) - M \\ &\qquad + \tilde{\mathbf{m}}^\top\mathbf{P}^{(\mathrm{R})}\mathbf{K}_{\mathbf{uu}}\mathbf{P}^{(\mathrm{R})}\tilde{\mathbf{m}} - \log|\mathbf{T}| - \log|\tilde{\mathbf{S}}|\Big). \end{aligned} \tag{7}$$

As will be shown in Section 4, the proposed preconditioning allows R-SVGP (and L-SVGP) to reach performance comparable to W-SVGP (Fig. 1).

Remark (connection to Newton-Schulz). The preconditioner $\mathbf{P}^{(\mathrm{R})} = 2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T}$, which appears several times in both the standard (4) and preconditioned (7) R-SVGP bound, is exactly one Newton-Schulz iteration for $\tilde{\mathbf{K}}^{-1}$ starting from $\mathbf{T}$, which is also a NG step on the objective (5) in $\mathbf{T}$-space. Thus, R-SVGP optimisation effectively alternates NG updates in $\mathbf{L}$-space with a single implicit NG step in $\mathbf{T}$-space as part of the bound computation, unveiling an insightful connection between inverse-free variational bounds and iterative matrix inversion.

3.3 Stopping Criteria

As part of the alternating optimisation routine proposed in Section 3.1, we wish to choose the minimum number t^* of NG updates per Adam step that brings R-SVGP close enough to L-SVGP. Below we discuss two possible stopping criteria to choose t^* adaptively.

A first option is to monitor when the normalised residual

$$||\mathbf{L}_t^\top \tilde{\mathbf{K}} \mathbf{L}_t - \mathbf{I}||_{\mathrm{F}} / \sqrt{M} \tag{8}$$

falls below a threshold ϵ. This quantity is virtually free, as $\mathbf{L}_t^\top \tilde{\mathbf{K}} \mathbf{L}_t$ is reused in the NG computation (6), and it is meaningful, as it is zero iff $\mathbf{L}_t = \mathrm{chol}(\tilde{\mathbf{K}}^{-1})$. Moreover, this stopping criterion is general and can be applied to arbitrary likelihoods for both shallow and deep GPs. In practice, we find that a tolerance of $\epsilon = 10^{-3}$ to 5×10^{-3} works well for a wide range of shallow GP prediction tasks.

Alternatively, for Gaussian likelihoods, one can directly monitor the slack introduced in the R-SVGP bound wrt L-SVGP by the upper bound (3) on the predictive variance $\sigma_n^{2(\mathrm{L})}$. This approach is similar in spirit to stopping criteria used for conjugate gradient approximations of GPs (MacKay and Gibbs, 1997; Artemev et al., 2021). We start with the following lower bound on $\sigma_n^{2(\mathrm{L})}$, proved in Section S3.

Proposition 3. *Let $\sigma_n^{2(L)}$ be as in Eq. (2). Since $\tilde{\mathbf{S}}$ is diagonal and PD, wlog, we can rewrite it as $\tilde{\mathbf{S}} = \tilde{\mathbf{S}}' + \sigma^2\mathbf{I}$, where $\tilde{\mathbf{S}}'$ is diagonal and PD. Then,*

$$\sigma_n^{2(L)} \geq k_{nn} - \frac{1}{\sigma^2}\Big(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_{-}\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_{-}\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\Big) =: L_n, \tag{9}$$

where $\tilde{\mathbf{K}}_{-} = \mathbf{K}_{\mathbf{uu}} + \tilde{\mathbf{S}}'$, so that $\tilde{\mathbf{K}} = \tilde{\mathbf{K}}_{-} + \sigma^2\mathbf{I}$.

By subtracting the upper and lower bounds on $\sigma_n^{2(\mathrm{L})}$, we find the quantity

$$G_n := U_n - L_n = ||(\mathbf{I} - \tilde{\mathbf{K}}\mathbf{T})\mathbf{k}_{\mathbf{u}n}||^2/\sigma^2.$$

Since σ_n^2 enters the expected log-likelihood term of the ELBO as $-\sum_n \sigma_n^2/(2\sigma_{\mathrm{obs}}^2)$, stopping when $G := \sum_n G_n \leq 2\sigma_{\mathrm{obs}}^2\epsilon$ ensures the R-SVGP to L-SVGP gap from variance approximation is at most ϵ. While less general, this criterion measures distance in the ELBO's natural units and, like the first, is minimised at $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$. The per-datum computation of G is quadratic in M, and suitable values of ϵ depend on the prediction task at hand.

3.4 Other Practical Considerations

Bound Evaluation. While inverse-free, a direct evaluation of the R-SVGP bound (7) still entails $\mathcal{O}(BM^2 + M^3)$ due to the cubic trace terms in the KL bound, each involving products of three or more $M \times M$ matrices. Luckily, cheap unbiased estimators of such traces can be obtained via Hutchinson's method (Hutchinson, 1989), which only require matrix vector products, reducing the cost to $\mathcal{O}((B+K)M^2)$ with K random probe vectors used. When K is too small, however, the resulting ELBO estimates become noisy, which may hinder optimisation. Section S6.1 studies this runtime–accuracy trade-off on two regression datasets and shows that moderate values of K can yield significant speedups without materially affecting performance.

Bound Optimisation. Despite the savings on evaluation, optimisation of R-SVGP remains cubic in M due to the full matmuls in the NG (6). On modern accelerators, matmuls are highly optimised operations that are more suited to low-precision arithmetic than matrix decompositions. Therefore, with the right hardware, the proposed inverse-free approach may be faster than other SVGP variants when the number of NG updates t^* is moderate. Furthermore, we conjecture that randomized matmul techniques (Drineas and Mahoney, 2016) could further reduce NG cost.

Step Size Scheduling. Given a stopping criterion, we seek a step-size schedule $\{\gamma_t\}$ that minimises the number of NG updates t^* necessary to satisfy it. We find that a suitable schedule crucially depends on whether the inducing location $\mathbf{Z}$ is optimised or not. When $\mathbf{Z}$ is trained, a log-linear schedule (Salimbeni et al., 2018) from a small value, e.g., $\gamma_0 = 10^{-5}$, to $\gamma_T = 1$, where it stays constant, over 10 steps is a robust, if a bit conservative, choice. On the other hand, when $\mathbf{Z}$ is fixed, a single NG step with $\gamma = 1$ typically suffices ($t^* = 1$), thus rendering the stopping criterion unnecessary. This suggests rapid movement in $\mathbf{Z}$ is the dominant source of $\tilde{\mathbf{K}}^{-1}$ drift that prevents cheap NG convergence. When runtime is a priority, we recommend mild regularisation of the learning of $\mathbf{Z}$ by **1**) freezing $\mathbf{Z}$ for the first few training iterations, **2**) using a smaller learning rate for $\mathbf{Z}$, and **3**) raising Adam's β_1 parameter from the default 0.9 to 0.99 for $\mathbf{Z}$ only. When coupled with k-means++ initialisation of $\mathbf{Z}$ (Arthur and Vassilvitskii, 2007) on shallow GP models, these heuristics usually make $\mathbf{T}$ optimisation much cheaper, often requiring only a small number $t^* \leq 3$ of NG updates with fixed step size $\gamma_t = 1$. Furthermore, in some cases, they also improve the model's overall training performance, possibly helping tame instability of the gradients wrt $\mathbf{Z}$.

Computational Summary. W-SVGP and L-SVGP both have per-iteration cost $\mathcal{O}(BM^2 + M^3)$. For

R-SVGP, Hutchinson trace estimation reduces the cost of bound *evaluation* from $\mathcal{O}(BM^2 + M^3)$ to $\mathcal{O}((B + K)M^2)$, where K is the number of probe vectors. *Optimising* R-SVGP, however, still requires t^* NG updates of $\mathbf{T}$, which contribute a cubic $\mathcal{O}(t^*M^3)$ cost, so that one outer training iteration costs $\mathcal{O}((B+K)M^2 + t^*M^3)$. Hence, potential speedups over W-SVGP and L-SVGP depend on: the gains from cheaper bound evaluation, the number t^* of NG updates required, and the practical speed of matrix multiplications relative to matrix decompositions on the hardware used, despite their identical cubic asymptotic scaling. In Section 4.2, we illustrate a setting where, using the heuristics discussed above, these factors favour R-SVGP, leading to faster training than the other variants.

4 RESULTS

This section investigates three main questions:

1. **Efficacy**: do the tools introduced in Section 3 solve the optimisation issues of R-SVGP?
2. **Efficiency**: can R-SVGP be made faster than standard baselines?
3. **Generality**: can R-SVGP be used as a drop-in replacement for complex SVGP-based models?

To answer these questions, we perform experiments on toy datasets, UCI regression benchmarks, and complex models such as deep GPs (Salimbeni and Deisenroth, 2017) and convolutional GPs (van der Wilk et al., 2017). We compare R-SVGP against W-SVGP, as it is the default parameterisation in popular GP libraries (e.g., Matthews et al., 2017), and L-SVGP, as it is the parameterisation to which R-SVGP reverts when $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$. Unless otherwise specified, we write R-SVGP and L-SVGP to refer to the versions with inducing mean preconditioning (Section 3.2) and, for R-SVGP, NG updates for $\mathbf{T}$ (Section 3.1), and we employ the standard ARD RBF kernel (Williams and Rasmussen, 2006) for all models. All results are averaged over 5 random seeds, reported as mean ± standard error (unless noted). The setup of all experiments below is detailed in Section S5.

4.1 Efficacy: Toy Datasets

To assess the effectiveness of the NG updates for $\mathbf{T}$ (Section 3.1) and the inducing mean preconditioning (Section 3.2), we compare versions of the R-SVGP bound that take advantage of either or both of these tools. In the following, we use suffixes N and P to indicate the presence of NG updates for $\mathbf{T}$ and inducing mean preconditioning, respectively. In the context of efficacy, the most interesting baseline is the R-SVGP bound without preconditioning and where $\mathbf{T}$ is trained with Adam, as in van der Wilk et al. (2022).

The benefits of the tools proposed in Section 3 are apparent even with toy datasets. Fig. 1 shows single-run training loss traces for each bound optimised on SNELSON, a one-dimensional regression task, and BANANA, a two-dimensional binary classification task. In both cases, the R-SVGP bounds with NG updates (i.e., R-SVGP (N) and R-SVGP (NP)) match the performance of the corresponding L-SVGP bounds (i.e., L-SVGP and L-SVGP (P), respectively). Conversely, the bounds trained only with Adam suffer from either much slower convergence (R-SVGP) or worse results (R-SVGP (P)) than their NG counterparts. Furthermore, preconditioning is crucial for the L-SVGP and R-SVGP bounds to match the performance of W-SVGP. However, for R-SVGP it helps *only* when coupled with NG: the inverse-free preconditioner $\mathbf{P}^{(\mathrm{R})}$ is an effective surrogate for $\tilde{\mathbf{K}}^{-1}$ only when $\mathbf{T}$ tracks $\tilde{\mathbf{K}}^{-1}$ closely.

Overall, R-SVGP (NP) is the only inverse-free variational bound that matches the performance of the Cholesky-based variants on both datasets.

4.2 Efficiency: UCI Benchmarks

Next, we investigate whether a well-tuned version of R-SVGP, which takes advantage of the heuristics in Sections 3.3 and 3.4, can be made faster than standard baselines on realistic datasets. To this end, we consider the ELEVATORS ($N = 16599$, $D = 18$) and KIN40K ($N = 40000$, $D = 8$) UCI scalar regression datasets. These datasets have different characteristics: SVGPs are known to perform well on ELEVATORS, with performance saturating at $M = 1000$ to 2000 inducing points, while KIN40K is a more challenging dataset where performance keeps improving as M increases.

For all methods, given a number M of inducing points, we initialise the inducing location $\mathbf{Z}$ with k-means++ and train it alongside the other model parameters using Adam with a learning rate of 5×10^{-3}. For R-SVGP, we apply the heuristics discussed in Section 3.4 to regularise the optimisation of $\mathbf{Z}$, with details given in Section S5. In turn, this allows us to choose a fixed NG step size $\gamma_t = 1$ for $\mathbf{T}$ optimisation, which we stop adaptively using the general criterion (8) with tolerance $\epsilon = 5 \times 10^{-3}$. All methods are trained for 20000 iterations with batch size $B = 100$. Fig. 2 plots the test negative log-predictive density (NLPD) against training time for each method on both datasets, with M ranging from 1000 to 4000. As expected, the performance of all methods is stable for $M \geq 2000$ on ELEVATORS, while it improves as M increases on KIN40K.

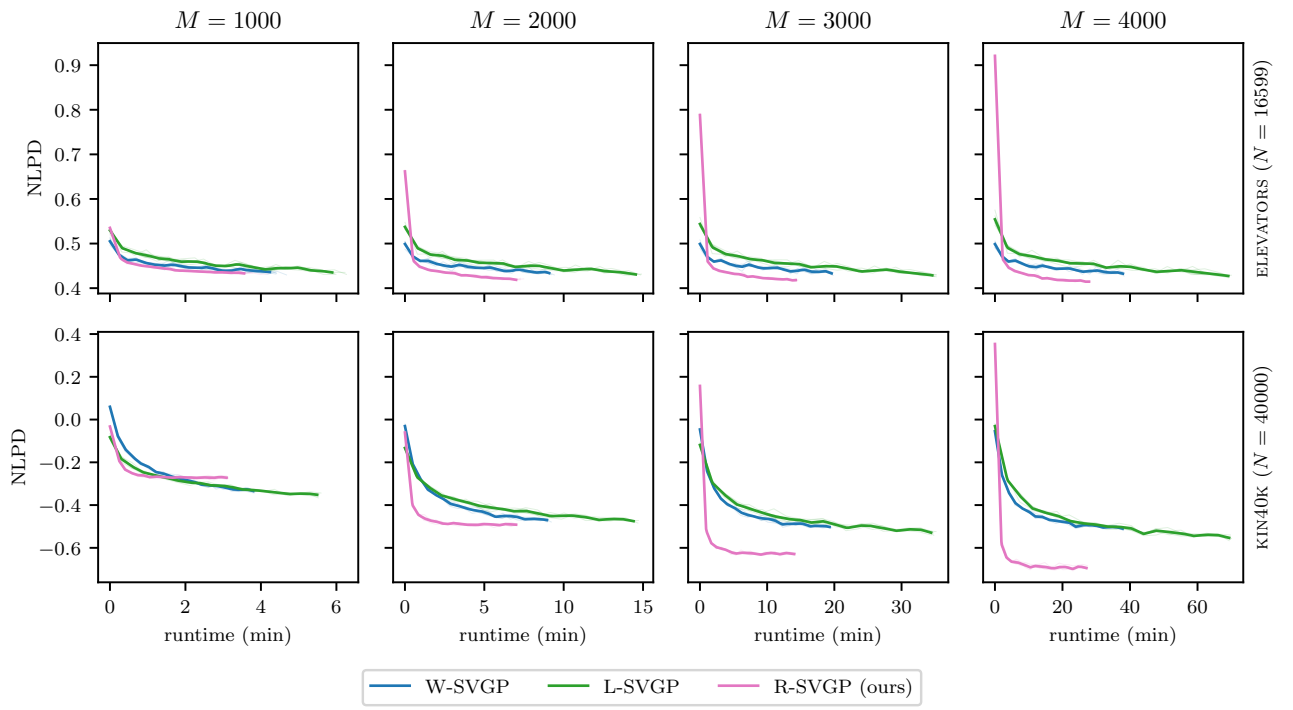

Figure 2: NLPD/runtime on ELEVATORS and KIN40K datasets for different choices of M and batch size $B = 100$. Lines show the mean over 5 seeds; shaded regions indicate ± 1 standard error (often smaller than line width).

Overall, R-SVGP shows faster convergence than both W-SVGP and L-SVGP, with a total training time that is up to 2.5× faster than L-SVGP and up to 1.4× faster than W-SVGP in our setup. On ELEVATORS, R-SVGP achieves comparable NLPD to W-SVGP and L-SVGP, while, on KIN40K, R-SVGP attains significantly better performance than other baselines for all M except for $M = 1000$. This shows a potential performance compromise when applying the heuristics from Section 3.4 to the optimisation of $\mathbf{Z}$: while they can improve the overall training dynamics when M is sufficiently large (e.g., $M \geq 2000$ on KIN40K), regularising $\mathbf{Z}$ updates too much may lead to suboptimal performance when M is much smaller than the optimal number for a given dataset (e.g., $M = 1000$ on KIN40K).

4.3 Generality: Beyond Scalar Shallow SVGP

So far our experiments used shallow, scalar-output SVGPs with RBF kernels. Here we test whether the R-SVGP bound can be used for more complex SVGP-based models, including multi-output GPs (van der Wilk et al., 2020a), deep GPs (Salimbeni and Deisenroth, 2017), and non-standard kernels such as convolutional kernels (van der Wilk et al., 2017). Our goal in this subsection is *efficacy*, not efficiency: we ask whether R-SVGP can be successfully optimised and compete with standard baselines in terms of predictive performance, without optimising for the fastest training setup. Accordingly, we *do not* apply the heuristics of Section 3.4 to $\mathbf{Z}$. For $\mathbf{T}$, we run natural gradients (NG) with a conservative increasing log-linear step-size schedule from $\gamma_0 = 10^{-5}$ to $\gamma_T = 1$ over 10 steps, and select t^* adaptively using the general criterion (8) with a strict tolerance ϵ between 10^{-9} and 10^{-6} depending on the specific experiment. This intentionally conservative routine is *not intended* to yield a speed-up over baselines; our aim is to verify that R-SVGP achieves comparable predictive performance in multi-output, deep, and convolutional-kernel settings. While we expect the heuristics of Section 3.4 to transfer to these settings, doing so is non-trivial. In particular, regularising the updates of $\mathbf{Z}$ presumes a good initialisation of the latter: for shallow GPs, k-means++ is effective, but in more complex models finding a suitable initialisation is more challenging, and we leave a systematic study to future work.

Deep GPs. In the formulation of Salimbeni and Deisenroth (2017), a deep GP (DGP) is built by stacking L SVGP layers: each layer $\ell \in \{1, \dots, L\}$ has its own inducing location $\mathbf{Z}^{(\ell)}$ and inducing variable $\mathbf{u}^{(\ell)}$, the variational posterior factorises as $\prod_{\ell=1}^{L} q(\mathbf{u}^{(\ell)})$, and the ELBO equals a Monte Carlo estimate of the expected log-likelihood through the latent layers mi-

Table 1: NLPD on KIN40K for different choices of L, $M = 500$ and batch size $B = 1000$, after 200k iterations. Entries are mean (standard error) over 5 seeds.

L	W-SVGP	L-SVGP	R-SVGP
1	-0.38 (0.001)	-0.39 (0.002)	-0.39 (0.002)
2	-1.86 (0.03)	-1.88 (0.02)	-1.88 (0.02)
3	-2.39 (0.03)	-2.64 (0.01)	-2.64 (0.01)

nus the sum of layer-wise KL terms. Consequently, R-SVGP applies to the deep setting layer-wise, with each layer endowed with its own $\mathbf{T}^{(\ell)}$ optimised independently using the tools of Section 3. When modelling a scalar function $f\colon \mathbb{R}^D \to \mathbb{R}$, standard DGP implementations (Dutordoir et al., 2021) use multi-output hidden layers with width Q_ℓ (often $Q_\ell = D$): each hidden layer is modelled as Q_ℓ independent SVGPs sharing $\mathbf{Z}^{(\ell)}$ and kernel hyperparameters, but with per-output variational parameters $\{\tilde{\mathbf{m}}^{(\ell,q)}, \tilde{\mathbf{S}}^{(\ell,q)}\}_{q=1}^{Q_\ell}$. For W-SVGP layers, this sharing implies a single Cholesky decomposition of $\mathbf{K}_{\mathbf{u}^{(\ell)}\mathbf{u}^{(\ell)}}$ is required per hidden layer, regardless of Q_ℓ. For L-SVGP or R-SVGP layers, the matrix of interest becomes $\tilde{\mathbf{K}}^{(\ell,q)} = \mathbf{K}_{\mathbf{u}^{(\ell)}\mathbf{u}^{(\ell)}} + \tilde{\mathbf{S}}^{(\ell,q)}$, which depends on the output via $\tilde{\mathbf{S}}^{(\ell,q)}$. Although each $\tilde{\mathbf{K}}^{(\ell,q)}$ remains $M \times M$, one now needs Q_ℓ distinct factorisations (for L-SVGP) or Q_ℓ NG inner loops (for R-SVGP) per training step. To avoid this mismatch, we share a single $\tilde{\mathbf{S}}^{(\ell)}$ across outputs within a layer, which restores one decomposition/NG inner loop at the cost of reduced variational flexibility.

We evaluate the three DGP parameterisations on KIN40K, ranging L from 1 to 3 layers and using $M = 500$ inducing points per layer. Table 1 reports the test NLPD for each method. All methods improve as L increases, showcasing the benefits of depth on this dataset. R-SVGP matches the performance of L-SVGP for all L, proving that the inverse-free bound can be successfully optimised in deep models using the tools of Section 3. W-SVGP performs slightly worse for $L = 3$, perhaps because of the increased difficulty of optimising the more flexible variational posterior in deeper models.

Convolutional GPs. Convolutional GPs (ConvGPs) (van der Wilk et al., 2017) encode translation–equivariant structure in SVGPs via specialised kernels and interdomain inducing variables that live in patch space rather than input space. Consequently, R-SVGP and L-SVGP apply unchanged: one simply substitutes the interdomain covariances $(\mathbf{K}_{\mathbf{uu}}, \mathbf{k}_{\mathbf{u}n})$ in the standard formulas. For multi-class classification with C classes, ConvGPs are typically implemented as C independent per-class latent GPs that share inducing locations $\mathbf{Z}$ and kernel hyperparameters. Hence, the same cost considerations as in multi-output SVGPs apply: to avoid C per-step factorisations (L-SVGP) or NG inner loops (R-SVGP), we tie a single-channel $\tilde{\mathbf{S}}$ across classes, trading some variational flexibility relative to W-SVGP for matched computational cost.

Table 2: Test error (%) on MNIST with RBF and weighted convolutional kernels, $M = 750$, $B = 200$. Entries are mean (standard error) over 5 seeds.

Kernel	W-SVGP	L-SVGP	R-SVGP
RBF	1.96 (0.02)	1.95 (0.01)	1.94 (0.02)
Conv	1.31 (0.02)	1.53 (0.02)	1.53 (0.03)

We follow the same experimental setup as in van der Wilk et al. (2017) to evaluate the three parameterisations on the MNIST dataset. Table 2 compares the test error of each method using either a standard RBF kernel or a weighted convolutional kernel (van der Wilk et al., 2017) at the end of training. All methods benefit from the inductive bias of the convolutional kernel, with W-SVGP slightly outperforming the other methods because of its more flexible variational posterior. R-SVGP matches the performance of L-SVGP, confirming that the effectiveness of the tools in Section 3 extends to non-standard kernels.

5 RELATED WORK

Every Gaussian process method relies on a numerical (approximation) method for calculating inverse-vector products and determinants, with the Cholesky decomposition being commonly used because of its accuracy. However, due to its $O(N^2)$ memory and $O(N^3)$ time costs, alternatives have long been investigated.

In particular, iterative Krylov-subspace solvers (Liesen and Strakoš, 2013) and Conjugate Gradients (CG) (Hestenes et al., 1952) were proposed as alternatives (MacKay and Gibbs, 1997; Davies, 2015). When terminated early, these methods can provide good approximations at lower computational and memory ($O(N)$) cost, and their reliance on matmuls make them well-suited to modern GPU hardware (Gardner et al., 2018; Wang et al., 2019). However, selecting when to terminate drastically affects the stability and quality of the results, although partial solutions for automatically finding a good trade-off exist (Artemev et al., 2021).

The methods above are limited to full-batch regression with Gaussian likelihoods, where the marginal likelihood is a log Gaussian density evaluation. By contrast, variational approximations (Titsias, 2009) support non-Gaussian likelihoods (Hensman et al., 2015),

mini-batching Hensman et al. (2013), and deep structures (Salimbeni and Deisenroth, 2017). In these settings, CG is not applicable in the same way, and existing methods therefore rely on Cholesky factorisations, which are poorly matched to low-precision, parallel hardware.

We attempt to bridge these lines of work by proposing an objective for GP inference that is hardware-friendly yet retains the benefits of the SVGP framework.

6 DISCUSSION

This work introduces, to our knowledge, the first practical inverse-free sparse variational GP method that trains stably on realistic datasets without matrix decompositions. Using only matrix multiplications (matmuls), R-SVGP attains comparable predictive performance to Cholesky-based baselines and serves as a drop-in replacement in deeper and interdomain SVGP models, while offering potential speed-ups when well tuned. Several avenues for future work remain.

While inverse-free, R-SVGP remains cubic in M due to the matmuls required by the NG optimisation of $\mathbf{T}$. Consequently, potential speed-ups over Cholesky-based methods depend on keeping the NG inner loop short. We provide heuristics that help in shallow GPs, but their benefit in deep, multi-output, or interdomain settings merits further study and tuning for speed.

Moreover, R-SVGP is built on L-SVGP, which, in multi-output regimes, trades some variational flexibility to match the computational cost of W-SVGP. This suggests that alternative inverse-free parameterisations targeting W-SVGP, such as the construction sketched in Section 2, represent a promising direction.

Finally, lower-precision regimes (e.g., FP16/FP32), where matmuls fully exploit modern accelerators, may amplify the potential speed and memory advantages of R-SVGP over Cholesky-based alternatives, and we are excited to investigate this systematically.

Acknowledgements

The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility (Richards, 2015) in carrying out this work. Stefano Cortinovis is supported by the EPSRC Centre for Doctoral Training in Modern Statistics and Statistical Machine Learning (EP/S023151/1).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Artemev, A., Burt, D. R., and van der Wilk, M. (2021). Tighter bounds on the log marginal likelihood of Gaussian process regression using conjugate gradients. In *International Conference on Machine Learning*, pages 362–372. PMLR.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Ben-Israel, A. (1965). An iterative method for computing the generalized inverse of an arbitrary matrix. *Mathematics of Computation*, pages 452–455.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Burt, D., Rasmussen, C. E., and van der Wilk, M. (2019). Rates of convergence for sparse variational Gaussian process regression. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871. PMLR.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. (2020). Convergence of sparse variational inference in Gaussian processes regression. *Journal of Machine Learning Research*, 21(131):1–63.

Davies, A. (2015). *Effective implementation of Gaussian process regression for machine learning*. Phd thesis, University of Cambridge.

Drineas, P. and Mahoney, M. W. (2016). Randnla: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90.

Dutordoir, V., Salimbeni, H., Hambro, E., McLeod, J., Leibfried, F., Artemev, A., van der Wilk, M., Hensman, J., Deisenroth, M. P., and John, S. (2021). GPflux: A library for deep Gaussian processes.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 31.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*.

Hensman, J., Matthews, A. G. d. G., Filippone, M., and Ghahramani, Z. (2015). MCMC for variationally sparse Gaussian processes. *Advances in neural information processing systems*, 28.

Hestenes, M. R., Stiefel, E., et al. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436.

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

Kelley, C. T. (1995). *Iterative methods for linear and nonlinear equations*. SIAM.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liesen, J. and Strakoš, Z. (2013). *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, Oxford.

MacKay, D. J. C. and Gibbs, M. (1997). Efficient implementation of Gaussian processes. *Neural Computation*.

Matthews, A. G. d. G. (2017). *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., Le, P., Ghahramani, Z., Hensman, J., et al. (2017). GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6.

Panos, A., Dellaportas, P., and Titsias, M. K. (2018). Fully scalable Gaussian processes using subspace inducing inputs. arXiv:1807.02537 [cs, stat].

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*.

Richards, A. (2015). University of Oxford Advanced Research Computing.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. *Advances in neural information processing systems*, 30.

Salimbeni, H., Eleftheriadis, S., and Hensman, J. (2018). Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 689–697. PMLR.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574. PMLR.

van der Wilk, M. (2019). *Sparse Gaussian process approximations and applications*. PhD thesis, University of Cambridge.

van der Wilk, M., Artemev, A., and Hensman, J. (2022). Improved inverse-free variational bounds for sparse Gaussian processes. In *Fourth Symposium on Advances in Approximate Bayesian Inference*.

van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. (2020a). A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*.

van der Wilk, M., John, S., Artemev, A., and Hensman, J. (2020b). Variational Gaussian process models without matrix inverses. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–9. PMLR.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional Gaussian processes. *Advances in neural information processing systems*, 30.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact Gaussian processes on a million data points. *Advances in neural information processing systems*, 32.

Williams, C. K. I. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Sections 2 and 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] Sections 2 and 3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Public code is provided via GitHub URL.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] Section 3 and supplementary material.
   (b) Complete proofs of all theoretical results. [Yes] Provided in the supplementary material.
   (c) Clear explanations of any assumptions. [Yes] Section 3 and supplementary material.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] Provided as part of the supplementary material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Section 4 and supplementary material.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] Where applicable, mean and standard error over 5 seeds are reported.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Provided in the supplementary.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] All datasets and codebases used are cited in the supplementary material.
   (b) The license information of the assets, if applicable. [Not Applicable] All datasets used are publicly available.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Inverse-Free Sparse Variational Gaussian Processes: Supplementary Material

The supplementary material contains the proofs of Propositions 1 to 3 in Section 3, a brief derivation of the alternative inverse-free bound mentioned in Section 2, as well as additional details on the experiments in Section 4 and further experimental results. All sections and equations in the supplementary material are prefixed with an 'S' for clarity.

S1 PROOF OF PROPOSITION 1

Proof. Given the decomposition for our variational parameter $\mathbf{T} = \mathbf{L}\mathbf{L}^\top$, our goal is to find the optimal $\mathbf{L}$ so that $\mathbf{T} = (\mathbf{K}_{\mathbf{uu}} + \tilde{\mathbf{S}})^{-1}$, without explicitly computing the inverse. We consider the problem of finding the Cholesky factor of the inverse of a matrix $\mathbf{A}$ via:

$$\mathbf{L}_{\mathbf{A}^{-1}} = \arg\min_{\mathbf{L}} \ell_{\mathbf{A}}(\mathbf{L}) \qquad \ell_{\mathbf{A}}(\mathbf{L}) := \mathrm{KL}\left[\mathcal{N}(\mathbf{0}, \mathbf{L}\mathbf{L}^\top) || \mathcal{N}(\mathbf{0}, \mathbf{A}^{-1})\right]. \tag{S1}$$

The above optimisation problem describes the update we take in the direction of the natural gradient:

$$\tilde{\mathbf{g}} = \mathbf{F}^{-1}\mathbf{g}, \tag{S2}$$

where $\mathbf{g}$ and $\tilde{\mathbf{g}}$ are column vectors denoting the gradient and the natural gradient of $\ell_{\mathbf{A}}$ wrt $\mathbf{L}$, while we denote with $\mathbf{F}$ the Fisher information matrix of the distribution $p(\mathbf{x}; \mathbf{L}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{L}\mathbf{L}^\top)$ parameterised by $\mathbf{L}$:

$$\mathbf{F} = -\mathbb{E}_p\left[\frac{\partial^2}{\partial \operatorname{vec}(\mathbf{L})\, \partial \operatorname{vec}(\mathbf{L})^\top} \log p(\mathbf{x}; \mathbf{L})\right], \tag{S3}$$

with $\operatorname{vec}(\cdot)$ the column-wise vectorisation operator.

We start by computing the gradient of the loss $\nabla \ell_{\mathbf{A}}$ wrt $\mathbf{L}$. As in Eq. (5),

$$\ell_{\mathbf{A}}(\mathbf{L}) = \mathrm{KL}\left[p(\mathbf{x}; \mathbf{L}) || \mathcal{N}(\mathbf{0}, \mathbf{A}^{-1})\right] = \frac{1}{2}\left(\operatorname{tr}(\mathbf{A}\mathbf{L}\mathbf{L}^\top) - \log|\mathbf{L}\mathbf{L}^\top|\right) + c_1, \tag{S4}$$

where c_1 is a constant that does not depend on $\mathbf{L}$. The first differential of $\ell_{\mathbf{A}}$ with respect to $\mathbf{L}$ is given by

$$\mathrm{d}\,\ell_{\mathbf{A}} = \frac{1}{2}\underbrace{\mathrm{d}\operatorname{tr}(\mathbf{A}\mathbf{L}\mathbf{L}^\top)}_{(a)} - \frac{1}{2}\underbrace{\mathrm{d}\log|\mathbf{L}\mathbf{L}^\top|}_{(b)}. \tag{S5}$$

Term (a) can be computed as

$$\begin{aligned}
\mathrm{d}\operatorname{tr}(\mathbf{A}\mathbf{L}\mathbf{L}^\top) &= \operatorname{tr}(\mathbf{A}\,\mathrm{d}(\mathbf{L}\mathbf{L}^\top)) \\
&= \operatorname{tr}(\mathbf{A}\,\mathrm{d}\mathbf{L}\mathbf{L}^\top + \mathbf{A}\mathbf{L}\,\mathrm{d}\mathbf{L}^\top) \\
&= \operatorname{tr}(\mathbf{L}^\top\mathbf{A}\,\mathrm{d}\mathbf{L}) + \operatorname{tr}(\mathbf{A}\mathbf{L}\,\mathrm{d}\mathbf{L}^\top) \\
&= 2\operatorname{vec}(\mathbf{A}\mathbf{L})^\top \operatorname{vec}(\mathrm{d}\mathbf{L}),
\end{aligned} \tag{S6}$$

where the last equality holds because $\mathbf{A}$ is symmetric. Term (b) is given by

$$\begin{aligned}
\mathrm{d}\log|\mathbf{L}\mathbf{L}^\top| &= \frac{1}{|\mathbf{L}\mathbf{L}^\top|}\,\mathrm{d}\,|\mathbf{L}\mathbf{L}^\top| \\
&= \frac{1}{|\mathbf{L}\mathbf{L}^\top|}|\mathbf{L}\mathbf{L}^\top|\operatorname{tr}(\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}(\mathbf{L}\mathbf{L}^\top)) \\
&= \operatorname{tr}(\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\mathbf{L}\mathbf{L}^\top + \mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{L}\,\mathrm{d}\mathbf{L}^\top) \\
&= \operatorname{tr}(\mathbf{L}^{-1}\,\mathrm{d}\mathbf{L}) + \operatorname{tr}(\mathbf{L}^{-\top}\,\mathrm{d}\mathbf{L}^\top) \\
&= 2\operatorname{vec}(\mathbf{L}^{-\top})^\top \operatorname{vec}(\mathrm{d}\mathbf{L}).
\end{aligned} \tag{S7}$$

Putting it all together, the differential of $\ell_{\mathbf{A}}$ takes the form

$$\mathrm{d}\,\ell_{\mathbf{A}} = (\mathrm{vec}(\mathbf{AL}) - \mathrm{vec}(\mathbf{L}^{-\top}))^\top \mathrm{vec}(\mathrm{d}\,\mathbf{L}), \tag{S8}$$

which implies that

$$\mathbf{g} = \mathrm{vec}(\mathbf{AL}) - \mathrm{vec}(\mathbf{L}^{-\top}). \tag{S9}$$

We now move to computing the Fisher information matrix $\mathbf{F}$ of $p(\mathbf{x};\mathbf{L})$ wrt $\mathbf{L}$. We have that

$$\log p(\mathbf{x};\mathbf{L}) = -\frac{1}{2}\log|\mathbf{L}\mathbf{L}^\top| - \frac{1}{2}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x} + c_2, \tag{S10}$$

where c_2 is a constant that does not depend on $\mathbf{L}$. The quantity of interest is the expected value of the second differential of $\log p(\mathbf{x};\mathbf{L})$ wrt p, which takes the form

$$\mathbb{E}_p\left[\mathrm{d}^2 \log p(\mathbf{x};\mathbf{L})\right] = -\frac{1}{2}\underbrace{\mathrm{d}^2\log|\mathbf{L}\mathbf{L}^\top|}_{(c)} - \frac{1}{2}\mathbb{E}_p\left[\underbrace{\mathrm{d}^2(\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x})}_{(d)}\right] \tag{S11}$$

because $\log|\mathbf{L}\mathbf{L}^\top|$ does not depend on $\mathbf{x}$. By taking advantage of Eq. (S7), term (c) is given by

$$\begin{aligned}
\mathrm{d}^2\log|\mathbf{L}\mathbf{L}^\top| &= 2\,\mathrm{d}\,\mathrm{tr}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&= -2\,\mathrm{tr}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&= -2\,\mathrm{vec}((\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})^\top)^\top\,\mathrm{vec}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \qquad \text{(S12)} \\
&= -2\left[\mathbf{C}\,\mathrm{vec}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}))\right]^\top\mathrm{vec}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&= -2\left[\mathbf{C}(\mathbf{I}\otimes\mathbf{L}^{-1})\,\mathrm{vec}(\mathbf{dL})\right]^\top(\mathbf{I}\otimes\mathbf{L}^{-1})\,\mathrm{vec}(\mathrm{d}\,\mathbf{L}) \\
&= -2\,\mathrm{vec}(\mathbf{dL})^\top(\mathbf{I}\otimes\mathbf{L}^{-\top})\mathbf{C}(\mathbf{I}\otimes\mathbf{L}^{-1})\,\mathrm{vec}(\mathrm{d}\,\mathbf{L}) \\
&= -2\,\mathrm{vec}(\mathrm{d}\,\mathbf{L})^\top\mathbf{C}(\mathbf{L}^{-\top}\otimes\mathbf{L}^{-1})\,\mathrm{vec}(\mathrm{d}\,\mathbf{L}), \qquad \text{(S13)}
\end{aligned}$$

where Eq. (S12) follows from $\mathrm{tr}(\mathbf{X}^\top\mathbf{Y}) = \mathrm{vec}(\mathbf{X})^\top\mathrm{vec}(\mathbf{Y})$, and $\mathbf{C} = \mathbf{C}^\top$ is the commutator matrix such that $\mathbf{C}\,\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{X}^\top)$ and $\mathbf{C}(\mathbf{X}\otimes\mathbf{Y}) = (\mathbf{Y}\otimes\mathbf{X})\mathbf{C}$.

Similarly, term (d) can be computed starting from the first differential

$$\begin{aligned}
\mathrm{d}(\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x}) &= \mathbf{x}^\top\,\mathrm{d}(\mathbf{L}^{-\top}\mathbf{L}^{-1})\mathbf{x} \\
&= -\mathbf{x}^\top(\mathbf{L}^{-\top}\,\mathrm{d}\,\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1} + \mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1})\mathbf{x} \\
&= -\,\mathrm{tr}(\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\,\mathrm{d}\,\mathbf{L}^\top) - \mathrm{tr}(\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&= -2\,\mathrm{tr}(\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}). \qquad \text{(S14)}
\end{aligned}$$

Then, the second differential takes the form

$$\begin{aligned}
\mathrm{d}^2(\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x}) &= -2\,\mathrm{tr}(\mathrm{d}(\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})) \\
&= 2\left[\mathrm{tr}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right. \\
&\qquad + \mathrm{tr}(\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\,\mathrm{d}\,\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&\qquad \left. + \mathrm{tr}(\mathbf{L}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right]. \qquad \text{(S15)}
\end{aligned}$$

By taking the expectation wrt p, we have that

$$\begin{aligned}
\mathbb{E}_p\left[\mathrm{d}^2(\mathbf{x}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{x})\right] &= 2\left[\mathrm{tr}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\mathbf{L}\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right. \\
&\qquad + \mathrm{tr}(\mathbf{L}^{-1}\mathbf{L}\mathbf{L}^\top\mathbf{L}^{-\top}\,\mathrm{d}\,\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) \\
&\qquad \left. + \mathrm{tr}(\mathbf{L}^{-1}\mathbf{L}\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right] \\
&= 2\left[2\,\mathrm{tr}(\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L}) + \mathrm{tr}(\mathrm{d}\,\mathbf{L}^\top\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right] \\
&= 2\left[2\,\mathrm{vec}(\mathrm{d}\,\mathbf{L})^\top\mathbf{C}(\mathbf{L}^{-\top}\otimes\mathbf{L}^{-1})\,\mathrm{vec}(\mathrm{d}\,\mathbf{L})\right. \qquad \text{(S16)} \\
&\qquad \left. + \mathrm{vec}(\mathrm{d}\,\mathbf{L})^\top\,\mathrm{vec}(\mathbf{L}^{-\top}\mathbf{L}^{-1}\,\mathrm{d}\,\mathbf{L})\right] \\
&= 2\,\mathrm{vec}(\mathrm{d}\,\mathbf{L})^\top\left[2\mathbf{C}(\mathbf{L}^{-\top}\otimes\mathbf{L}^{-1}) + (\mathbf{I}\otimes\mathbf{L}^{-\top}\mathbf{L}^{-1})\right]\mathrm{vec}(\mathrm{d}\,\mathbf{L}), \qquad \text{(S17)}
\end{aligned}$$

where the first component of Eq. (S16) follows from the expression already derived in Eq. (S13). Overall, the expectation in Eq. (S11) becomes

$$\mathbb{E}_p\left[\mathrm{d}^2 \log p(\mathbf{x};\mathbf{L})\right] = -\operatorname{vec}(\mathrm{d}\,\mathbf{L})^\top \left[\mathbf{C}(\mathbf{L}^{-\top}\otimes\mathbf{L}^{-1}) + (\mathbf{I}\otimes\mathbf{L}^{-\top}\mathbf{L}^{-1})\right]\operatorname{vec}(\mathrm{d}\,\mathbf{L}), \tag{S18}$$

which implies that the Fisher information matrix $\mathbf{F}$ takes the form

$$\begin{aligned}\mathbf{F} &= -\mathbb{E}_p\left[\frac{\partial^2}{\partial \operatorname{vec}\mathbf{L}\,\partial \operatorname{vec}\mathbf{L}^\top}\log p(\mathbf{x};\mathbf{L})\right] \\ &= \mathbf{C}(\mathbf{L}^{-\top}\otimes\mathbf{L}^{-1}) + (\mathbf{I}\otimes\mathbf{L}^{-\top}\mathbf{L}^{-1}) \\ &= (\mathbf{I}\otimes\mathbf{L}^{-\top})(\mathbf{I}+\mathbf{C})(\mathbf{I}\otimes\mathbf{L}^{-1}).\end{aligned} \tag{S19}$$

By rearranging the terms of *Eq.* (S2), we have that the natural gradient may be found without explicitly computing the inverse $\mathbf{F}$ by solving the equation

$$\mathbf{F}\tilde{\mathbf{g}} = \mathbf{g} \tag{S20}$$

for $\tilde{\mathbf{g}}$, under the constraint that the latter is the vectorisation of a lower triangular. Before we proceed, let us introduce the matrix form of the natural gradient vector as $\tilde{\mathbf{G}} = \operatorname{unvec}(\tilde{\mathbf{g}})$. Then we compute

$$\begin{aligned}\mathbf{F}\tilde{\mathbf{g}} &= (\mathbf{I}\otimes\mathbf{L}^{-\top})(\mathbf{I}+\mathbf{C})(\mathbf{I}\otimes\mathbf{L}^{-1})\tilde{\mathbf{g}} \\ &= (\mathbf{I}\otimes\mathbf{L}^{-\top})(\mathbf{I}+\mathbf{C})\operatorname{vec}(\mathbf{L}^{-1}\tilde{\mathbf{G}}) \\ &= (\mathbf{I}\otimes\mathbf{L}^{-\top})\operatorname{vec}(\mathbf{L}^{-1}\tilde{\mathbf{G}} + (\mathbf{L}^{-1}\tilde{\mathbf{G}})^\top) \\ &= \operatorname{vec}(\mathbf{L}^{-\top}\mathbf{L}^{-1}\tilde{\mathbf{G}} + \mathbf{L}^{-\top}(\mathbf{L}^{-1}\tilde{\mathbf{G}})^\top).\end{aligned} \tag{S21}$$

Then, Eq. (S20) is equivalent to

$$\mathbf{L}^{-\top}\mathbf{L}^{-1}\tilde{\mathbf{G}} + \mathbf{L}^{-\top}(\mathbf{L}^{-1}\tilde{\mathbf{G}})^\top = \mathbf{A}\mathbf{L} - \mathbf{L}^{-\top}. \tag{S22}$$

By left-multiplying both sides by $\mathbf{L}^\top$, we obtain

$$\begin{aligned}\mathbf{L}^{-1}\tilde{\mathbf{G}} + (\mathbf{L}^{-1}\tilde{\mathbf{G}})^\top &= \mathbf{L}^\top\mathbf{A}\mathbf{L} - \mathbf{I} \\ &= \operatorname{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) + \operatorname{triu}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \operatorname{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \mathbf{I} \\ &= \operatorname{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \frac{1}{2}(\mathbf{I} + \operatorname{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L})) \\ &\qquad + \left[\operatorname{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \frac{1}{2}(\mathbf{I} + \operatorname{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L}))\right]^\top,\end{aligned} \tag{S23}$$

which implies that

$$\mathbf{L}^{-1}\tilde{\mathbf{G}} = \operatorname{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \frac{1}{2}(\mathbf{I} + \operatorname{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L})). \tag{S24}$$

Finally, by left-multiplying both sides by $\mathbf{L}$, we find that

$$\tilde{\mathbf{G}} = \mathbf{L}\left[\operatorname{tril}(\mathbf{L}^\top\mathbf{A}\mathbf{L}) - \frac{1}{2}(\mathbf{I} + \operatorname{diag}(\mathbf{L}^\top\mathbf{A}\mathbf{L}))\right]. \tag{S25}$$

□

Notice that a similar procedure as the one above can be used to derive the natural gradient of the loss $\ell_{\mathbf{A}}$ with respect to different parameterisations of the covariance matrix of $p(\mathbf{x})$. This includes the marginal parameterisation $p(\mathbf{x};\mathbf{T}) = \mathcal{N}(\mathbf{0},\mathbf{T})$, as well as a general square root parameterisation $p(\mathbf{x};\mathbf{B}) = \mathcal{N}(\mathbf{0},\mathbf{B}\mathbf{B})$. In particular, it can be shown that, when using the marginal parameterisation, natural gradient descent with unit step size recovers the well-known Newtonian iteration for computing the inverse of a matrix (Ben-Israel, 1965).

S2 PROOF OF PROPOSITION 2

All gradients and differentials below are taken with respect to $\boldsymbol{\xi} = \{\boldsymbol{\theta}, \mathbf{Z}, \tilde{\mathbf{m}}, \tilde{\mathbf{S}}\}$. We show that the first differentials of $\tilde{\mathcal{L}}_{\text{lsvgp}}$ and $\tilde{\mathcal{L}}_{\text{rsvgp}}$ are equal when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$, from which the gradient equivalence follows. For ease of reference, we start by restating the SVGP ELBO (1), the preconditioned L-SVGP bound (Section 2), and the preconditioned R-SVGP bound (Section 3.2).

The general SVGP ELBO is

$$\mathcal{L} = \sum_{n=1}^{N} \underbrace{\mathbb{E}_{\mathcal{N}(\mu_n, \sigma_n^2)}\left[\log p(y_n | f(\mathbf{x}_n))\right]}_{(a)} - \underbrace{\text{KL}\left[q(\mathbf{u}) || p(\mathbf{u})\right]}_{(b)}. \tag{S26}$$

For L-SVGP, substituting $\mathbf{P}^{(\text{L})} = \tilde{\mathbf{K}^{-1}}$, $\tilde{\mathcal{L}}_{\text{lsvgp}}$ is defined by plugging the following expressions into the ELBO:

$$\begin{aligned}
\mu_n &= \mathbf{k}_{n\mathbf{u}} \mathbf{P}^{(\text{L})} \tilde{\mathbf{m}}, \\
\sigma_n^2 &= k_{nn} - \mathbf{k}_{n\mathbf{u}} \mathbf{P}^{(\text{L})} \mathbf{k}_{\mathbf{u}n}, \\
\text{KL}\left[q(\mathbf{u}) || p(\mathbf{u})\right] &= \frac{1}{2}\left(-\text{tr}(\mathbf{P}^{(\text{L})}\mathbf{K}_{\mathbf{uu}}) + \tilde{\mathbf{m}}^\top \mathbf{P}^{(\text{L})} \mathbf{K}_{\mathbf{uu}} \mathbf{P}^{(\text{L})} \tilde{\mathbf{m}} + \log|\tilde{\mathbf{K}}| - \log|\tilde{\mathbf{S}}|\right) =: \text{KL}_{\text{lsvgp}}.
\end{aligned} \tag{S27}$$

For R-SVGP, $\tilde{\mathcal{L}}_{\text{rsvgp}}$ is defined by plugging the following expressions into the ELBO:

$$\begin{aligned}
\mu_n &= \mathbf{k}_{n\mathbf{u}} \mathbf{P}^{(\text{R})} \tilde{\mathbf{m}}, \\
\sigma_n^2 &= k_{nn} - \mathbf{k}_{n\mathbf{u}} \mathbf{P}^{(\text{R})} \mathbf{k}_{\mathbf{u}n}, \\
\text{KL}\left[q(\mathbf{u}) \,\|\, p(\mathbf{u})\right] &\leq \frac{1}{2}\left(-\text{tr}(\mathbf{P}^{(\text{R})}\mathbf{K}_{\mathbf{uu}}) + \text{tr}(\tilde{\mathbf{K}}\mathbf{T}) - M + \tilde{\mathbf{m}}^\top \mathbf{P}^{(\text{R})} \mathbf{K}_{\mathbf{uu}} \mathbf{P}^{(\text{R})} \tilde{\mathbf{m}} - \log|\mathbf{T}| - \log|\tilde{\mathbf{S}}|\right) =: \text{KL}_{\text{rsvgp}}.
\end{aligned} \tag{S28}$$

Consider the preconditioners $\mathbf{P}^{(\text{L})} = \tilde{\mathbf{K}}^{-1}$ and $\mathbf{P}^{(\text{R})} = 2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T}$. Note that $\mathbf{T}$ does not depend on $\boldsymbol{\xi}$, i.e., $\text{d}\mathbf{T} = 0$. Their differentials are

$$\begin{aligned}
\text{d}\mathbf{P}^{(\text{L})} &= \text{d}(\tilde{\mathbf{K}}^{-1}) = -\tilde{\mathbf{K}}^{-1}(\text{d}\tilde{\mathbf{K}})\tilde{\mathbf{K}}^{-1}, \\
\text{d}\mathbf{P}^{(\text{R})} &= \text{d}(2\mathbf{T} - \mathbf{T}\tilde{\mathbf{K}}\mathbf{T}) = -\mathbf{T}(\text{d}\tilde{\mathbf{K}})\mathbf{T}.
\end{aligned} \tag{S29}$$

Hence, we have

$$\mathbf{P}^{(\text{L})} = \mathbf{P}^{(\text{R})} \text{ and } \text{d}\mathbf{P}^{(\text{L})} = \text{d}\mathbf{P}^{(\text{R})} \quad \text{when} \quad \mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1}), \tag{S30}$$

which is the key observation for the rest of the proof.

Consider the variational expectation terms (a) in the ELBO. These depend on $\boldsymbol{\xi}$ only through μ_n and σ_n^2. In turn, μ_n and σ_n^2 differ between L-SVGP and R-SVGP only through $\mathbf{P}^{(\text{L})}$ and $\mathbf{P}^{(\text{R})}$, respectively. Hence, by (S30), the differentials of the variational expectation terms match when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$.

Next, consider the KL term (b) in the ELBO, denoted by KL_{lsvgp} (S27) and KL_{rsvgp} (S28) for L-SVGP and R-SVGP, respectively. The terms coloured in blue in their expressions also differ between L-SVGP and R-SVGP only through $\mathbf{P}^{(\text{L})}$ and $\mathbf{P}^{(\text{R})}$. As a result, by (S30), the differentials of these terms match when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$. Excluding multiplicative constants and the terms in common between KL_{lsvgp} and KL_{rsvgp} (i.e., $\log|\tilde{\mathbf{S}}|$), the differentials of the remaining terms are given by

$$\text{d}\log|\tilde{\mathbf{K}}| = \text{tr}(\tilde{\mathbf{K}}^{-1}\text{d}(\tilde{\mathbf{K}})) \tag{S31}$$

for L-SVGP, and

$$\text{d}\left(\text{tr}(\tilde{\mathbf{K}}\mathbf{T}) - M - \log|\mathbf{T}|\right) = \text{tr}\left(\mathbf{T}\text{d}\tilde{\mathbf{K}}\right) \tag{S32}$$

for R-SVGP, which match when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$. Therefore, the differentials of the KL terms, and in turn of $\tilde{\mathcal{L}}_{\text{lsvgp}}$ and $\tilde{\mathcal{L}}_{\text{rsvgp}}$, match when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$, as desired.

Lastly, the same is not true for the naive preconditioner $\mathbf{P} = \mathbf{T}$. In particular, the second part of Eq. (S30) ceases to hold when $\mathbf{T} = \text{stopgrad}(\tilde{\mathbf{K}}^{-1})$, as $\text{d}\mathbf{P} = \text{d}\mathbf{T} = 0$.

S3 PROOF OF PROPOSITION 3

Proof. As in the statement of Proposition 3, we use the shorthand $\tilde{\mathbf{K}}_- = \mathbf{K_{uu}} + \tilde{\mathbf{S}}'$, so that $\tilde{\mathbf{K}} = \tilde{\mathbf{K}}_- + \sigma^2\mathbf{I}$. Then, proving the inequality

$$\begin{aligned}\sigma_n^{2(\mathrm{L})} &= k_{nn} - \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} \\ &\geq k_{nn} - \frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\right) \\ &= L_n \end{aligned} \tag{S33}$$

is equivalent to showing that

$$\frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\right) \geq \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n}. \tag{S34}$$

We write $\mathbf{T}\mathbf{k}_{\mathbf{u}n} = \tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the discrepancy due to the error with which $\mathbf{T}$ approximates $\tilde{\mathbf{K}}^{-1}$. Notice that both $\tilde{\mathbf{K}}_-$ and $\tilde{\mathbf{K}}$ are symmetric PD. Moreover, since $\mathbf{T}$ is parameterised via its Cholesky factor $\mathbf{L}$ with $\mathbf{T} = \mathbf{L}\mathbf{L}^\top$, $\mathbf{T}$ is also symmetric PD. Then, we have

$$\begin{aligned}&\frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\right) \\ &\quad= \frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2(\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} + \boldsymbol{\delta})^\top\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} + (\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} + \boldsymbol{\delta})^\top\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}(\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} + \boldsymbol{\delta})\right) \\ &\quad= \frac{1}{\sigma^2}\Big(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - \cancel{2}\mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} \cancel{-2\boldsymbol{\delta}^\top\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n}} \\ &\qquad\qquad \cancel{+\mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n}} \cancel{+2\boldsymbol{\delta}^\top\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n}} + \boldsymbol{\delta}^\top\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\boldsymbol{\delta}\Big) \\ &\quad= \frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}(\tilde{\mathbf{K}} - \sigma^2\mathbf{I})\mathbf{k}_{\mathbf{u}n} + \boldsymbol{\delta}^\top\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\boldsymbol{\delta}\right) \\ &\quad= \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n} + \frac{1}{\sigma^2}\boldsymbol{\delta}^\top\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\boldsymbol{\delta} \\ &\quad\geq \mathbf{k}_{n\mathbf{u}}\tilde{\mathbf{K}}^{-1}\mathbf{k}_{\mathbf{u}n}, \end{aligned} \tag{S35}$$

with equality when $\boldsymbol{\delta} = \mathbf{0}$, which is implied by $\mathbf{T} = \tilde{\mathbf{K}}^{-1}$. The last inequality follows from the fact that the product of $\tilde{\mathbf{K}}_-$ and $\tilde{\mathbf{K}}$ is PSD, as it is the sum of two PSD matrices, i.e.,

$$\tilde{\mathbf{K}}_-\tilde{\mathbf{K}} = \tilde{\mathbf{K}}_-(\tilde{\mathbf{K}}_- + \sigma^2\mathbf{I}) = \tilde{\mathbf{K}}_-^2 + \sigma^2\tilde{\mathbf{K}}_- \succeq \mathbf{0}.$$

□

As already discussed in Section 3.3, the upper bound U_n in Eq. (3) and the lower bound L_n in Eq. (9) that we proved above can be monitored to choose the number of NG optimisation steps for $\mathbf{T}$. In particular, by subtracting U_n and L_n, we find the quantity

$$\begin{aligned}G_n &= U_n - L_n \\ &= k_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\mathbf{k}_{\mathbf{u}n} + \frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}\tilde{\mathbf{K}}_-\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\right) \\ &= \frac{1}{\sigma^2}\left(\mathbf{k}_{n\mathbf{u}}\mathbf{k}_{\mathbf{u}n} - 2\mathbf{k}_{n\mathbf{u}}\mathbf{T}(\tilde{\mathbf{K}}_- + \sigma^2\mathbf{I})\mathbf{k}_{\mathbf{u}n} + \mathbf{k}_{n\mathbf{u}}\mathbf{T}(\tilde{\mathbf{K}}_- + \sigma^2\mathbf{I})\tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}\right) \\ &= \frac{1}{\sigma^2}||\mathbf{k}_{\mathbf{u}n} - \tilde{\mathbf{K}}\mathbf{T}\mathbf{k}_{\mathbf{u}n}||^2 \\ &= \frac{1}{\sigma^2}||(\mathbf{I} - \tilde{\mathbf{K}}\mathbf{T})\mathbf{k}_{\mathbf{u}n}||^2. \end{aligned} \tag{S36}$$

By optimising $\mathbf{T}$ until $G_n < \epsilon'$, one can ensure that the R-SVGP predictive variance in Eq. (3) is at most ϵ' larger than the L-SVGP predictive variance $\sigma_n^{2(\mathrm{L})}$ in Eq. (2). Moreover, the only part of the SVGP ELBO in Eq. (1)

that depends directly on the predictive variance σ_n^2 of the parameterisation is the expectation component, which, in the case of Gaussian likelihoods, is given by

$$\sum_{n=1}^{N} \mathbb{E}_{\mathcal{N}(\mu_n,\sigma_n^2)} \left[\log p(y_n|f(\mathbf{x}_n))\right] = -\frac{1}{2\sigma_{\text{obs}}^2} \sum_{n=1}^{N} \sigma_n^2 + c_3, \tag{S37}$$

where σ_{obs}^2 is the likelihood variance and c_3 is a constant that does not depend on σ_n^2. Therefore, as mentioned in the main text, optimising $\mathbf{T}$ until the stopping criterion

$$\sum_{n=1}^{N} G_n \leq 2\sigma_{\text{obs}}^2 \epsilon \tag{S38}$$

is reached ensures that the slack in the ELBO of the R-SVGP parameterisation introduced by upper bounding the predictive variance $\sigma_n^{2(\text{L})}$ of the L-SVGP parameterisation is at most ϵ.

The careful reader may have noticed that the stopping criterion derived above resembles a commonly used approach for iterative GP approximations (MacKay and Gibbs, 1997; Davies, 2015). In particular, iterative frameworks for solving GPs approximate key quantities of exact GPs, such as their predictive mean

$$\mu_* = \mathbf{k}_{*\mathbf{u}} \mathbf{K}_{\mathbf{ff}}^{-1} \mathbf{y}, \tag{S39}$$

by using iterative solvers (e.g., conjugate gradients) for the linear system $\mathbf{K}_{\mathbf{ff}}\mathbf{x} = \mathbf{y}$, until a stopping criterion is reached. A common criterion (MacKay and Gibbs, 1997) monitors the slack due to the approximation in the quadratic term of the true GP log marginal likelihood, i.e.,

$$-\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\mathbf{ff}}^{-1} \mathbf{y}, \tag{S40}$$

by employing upper and lower bounds on Eq. (S40), which resemble the bounds U_n and L_n, with $\mathbf{k}_{\mathbf{u}n}$ replaced by $\mathbf{y}$.

S4 DERIVATION OF ALTERNATIVE INVERSE-FREE BOUND

As in Section 2, start from M-SVGP and reparameterise the variational mean and covariance as $\mathbf{m} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{m}}$ and $\mathbf{S} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{S}}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}$, respectively, where $\tilde{\mathbf{S}}$ is PD and $\mathbf{R}$ is lower triangular with positive diagonal. By plugging these choices into the expressions for the M-SVGP predictive mean and variance, this yields

$$\mu_n = \mathbf{k}_{n\mathbf{u}}\mathbf{R}\tilde{\mathbf{m}}, \quad \sigma_n^2 = k_{nn} - \mathbf{k}_{n\mathbf{u}}(\mathbf{K}_{\mathbf{uu}}^{-1} - \mathbf{R}\tilde{\mathbf{S}}\mathbf{R}^\top)\mathbf{k}_{\mathbf{u}n}.$$

The predictive variance σ_n^2 above admits an inverse-free upper bound that depends on the auxiliary parameter $\mathbf{R}$. In particular, since $\mathbf{K}_{\mathbf{uu}}$ is PD, we have

$$(\mathbf{R}\mathbf{R}^\top - \mathbf{K}_{\mathbf{uu}}^{-1})\mathbf{K}_{\mathbf{uu}}(\mathbf{R}\mathbf{R}^\top - \mathbf{K}_{\mathbf{uu}}^{-1}) = \mathbf{R}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\mathbf{R}^\top - 2\mathbf{R}\mathbf{R}^\top + \mathbf{K}_{\mathbf{uu}}^{-1} \succeq \mathbf{0},$$

which implies that

$$-\mathbf{K}_{\mathbf{uu}}^{-1} \preceq \mathbf{R}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\mathbf{R}^\top - 2\mathbf{R}\mathbf{R}^\top.$$

By adding $\mathbf{R}\tilde{\mathbf{S}}\mathbf{R}^\top$ to both sides and plugging the resulting inequality into the expression for σ_n^2, we find that

$$\sigma_n^2 \leq k_{nn} + \mathbf{k}_{n\mathbf{u}}\mathbf{R}(\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R} - 2\mathbf{I} + \tilde{\mathbf{S}})\mathbf{R}^\top\mathbf{k}_{\mathbf{u}n},$$

with equality when $\mathbf{R} = \text{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$. As mentioned in the main text, working backwards, the upper bound above is exactly the predictive variance induced by the alternative covariance choice $\mathbf{S} = \mathbf{K}_{\mathbf{uu}} + \mathbf{K}_{\mathbf{uu}}\mathbf{R}(\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R} - 2\mathbf{I} + \tilde{\mathbf{S}})\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}$. This is easily checked by plugging such $\mathbf{S}$ into the expression for the M-SVGP predictive variance.

We now derive an upper bound on the resulting term $\text{KL}[q(\mathbf{u})||p(\mathbf{u})]$. To start, define the shorthand $\mathbf{A} = \mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}$, and rewrite $\mathbf{S}$ as

$$\mathbf{S} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\left(\mathbf{A} + \mathbf{A}^{-1} - 2\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}} = \mathbf{K}_{\mathbf{uu}}\mathbf{R}\mathbf{B}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}},$$

where $\mathbf{B} = \mathbf{A} + \mathbf{A}^{-1} - 2\mathbf{I} + \tilde{\mathbf{S}}$. By plugging the expressions for $\mathbf{m}$ and $\mathbf{S}$ into the M-SVGP KL term and rearranging, we find that

$$\begin{aligned}
\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] &= \frac{1}{2}\left(\mathrm{tr}(\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{S}) + \mathbf{m}^\top\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{m} - M + \log|\mathbf{K}_{\mathbf{uu}}| - \log|\mathbf{S}|\right) \\
&= \frac{1}{2}\left(\mathrm{tr}(\mathbf{R}\mathbf{B}\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}) + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}} - M - \log|\mathbf{A}| - \log|\mathbf{B}|\right) \\
&= \frac{1}{2}\left(\mathrm{tr}\left(\left(\mathbf{A} + \mathbf{A}^{-1} - 2\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{A}\right) - \log|\mathbf{B}| + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}} - M - \log|\mathbf{A}|\right) \\
&= \frac{1}{2}\left(\mathrm{tr}\left(\left(\mathbf{A} + \mathbf{A}^{-1} - 3\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{A}\right) - \log|\mathbf{B}| + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}} + \mathrm{tr}\left(\mathbf{A}\right) - M - \log|\mathbf{A}|\right) \\
&= \frac{1}{2}\left(\mathrm{tr}\left(\left(\mathbf{A} - 3\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{A}\right) + M - \log|\mathbf{B}| + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}} + \mathrm{tr}\left(\mathbf{A}\right) - M - \log|\mathbf{A}|\right).
\end{aligned}$$

By identifying the last three terms as $2 \cdot \mathrm{KL}\left[\mathcal{N}(\mathbf{0}, \mathbf{R}\mathbf{R}^\top)||\mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{-1})\right]$, we find that

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] = \frac{1}{2}\left(\mathrm{tr}\left(\left(\mathbf{A} - 3\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{A}\right) + M - \log|\mathbf{B}| + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}}\right) + \mathrm{KL}\left[\mathcal{N}(\mathbf{0}, \mathbf{R}\mathbf{R}^\top)||\mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{-1})\right].$$

Out of all other terms, only $\log|\mathbf{B}|$ requires matrix decompositions. However, since

$$\mathbf{A} + \mathbf{A}^{-1} - 2\mathbf{I} = \mathbf{A}^{-1/2}\left(\mathbf{A}^2 + \mathbf{I} - 2\mathbf{A}\right)\mathbf{A}^{-1/2} = \mathbf{A}^{-1/2}(\mathbf{A} - \mathbf{I})^2\mathbf{A}^{-1/2} \succeq \mathbf{0},$$

we have that $\mathbf{B} \succeq \tilde{\mathbf{S}}$, and thus $\log|\mathbf{B}| \geq \log|\tilde{\mathbf{S}}|$ with equality when $\mathbf{A} = \mathbf{I}$, which is implied by $\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$. By plugging this bound into the expression for $\mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})]$, we find that

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] \leq \frac{1}{2}\left(\mathrm{tr}\left(\left(\mathbf{A} - 3\mathbf{I} + \tilde{\mathbf{S}}\right)\mathbf{A}\right) + M - \log|\tilde{\mathbf{S}}| + \tilde{\mathbf{m}}^\top\mathbf{A}\tilde{\mathbf{m}}\right) + \mathrm{KL}\left[\mathcal{N}(\mathbf{0}, \mathbf{R}\mathbf{R}^\top)||\mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{-1})\right],$$

with equality when $\mathbf{R} = \mathrm{chol}(\mathbf{K}_{\mathbf{uu}}^{-1})$.

Finally, by plugging the expression for $\mathbf{A}$ back into the bound above, we obtain

$$\begin{aligned}
\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] \leq \frac{1}{2}\Big(&\mathrm{tr}\left(\left(\tilde{\mathbf{S}} - 3\mathbf{I} + \mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\right)\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\right) + M - \log|\tilde{\mathbf{S}}| + \tilde{\mathbf{m}}^\top\mathbf{R}^\top\mathbf{K}_{\mathbf{uu}}\mathbf{R}\tilde{\mathbf{m}}\Big) \\
&+ \mathrm{KL}\left[\mathcal{N}(\mathbf{0}, \mathbf{R}\mathbf{R}^\top)||\mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{-1})\right],
\end{aligned}$$

as stated in the main text.

S5 EXPERIMENTAL DETAILS

S5.1 Implementation

We implement our methods in Python using TensorFlow (Abadi et al., 2016). W-SVGP is the default SVGP parameterisation for GPflow (Matthews et al., 2017), and we follow its structure to implement R-SVGP and L-SVGP. Their extension to deep architectures is based on GPflux (Dutordoir et al., 2021). Our implementation is available at https://github.com/stefanocortinovis/relaxedgp/.

The experiments on toy datasets (Section 4.1) were run on an Apple Silicon M4 Pro CPU with 24GB of memory, while all other experiments (Section 4) were run on a single NVIDIA V100 GPU with 32GB of memory.

S5.2 Datasets

For the experiments presented in Section 4, we use datasets from four sources:

- **UCI repository**[3]: the ELEVATORS ($N = 16599$, $D = 18$) and KIN40K ($N = 40000$, $D = 8$) scalar regression datasets;

[3] https://archive.ics.uci.edu/datasets

- **OpenML**[4]: the BANANA ($N = 5300$, $D = 2$) classification dataset;
- **Keras**[5]: the MNIST ($N = 70000$, $D = 784$) classification dataset;
- **Other**: the SNELSON ($N = 200$, $D = 1$) regression dataset (Snelson and Ghahramani, 2005).

For SNELSON and BANANA we fit on the full, unnormalised datasets with no held-out split. For ELEVATORS and KIN40K we use a random 90/10 train/test split and we standardise the data to have zero mean and unit variance feature-wise using train statistics. For MNIST we use the standard 60k/10k train/test split and scale pixels to $[0, 1]$ with no augmentation.

S5.3 Model Setup

Here, we describe the model setup and initialisation for all the bounds considered (i.e., W-SVGP, L-SVGP, and R-SVGP) in the experiments presented in Section 4. Unless otherwise specified, model parameters shared by all bounds (e.g., kernel variances and likelihood variance) are initialised using the GPflow defaults.

For all the experiments, we use ARD RBF kernels (Williams and Rasmussen, 2006). For the multi-class classification experiment on MNIST (Section 4.3), we also include results for the weighted convolutional kernel, as implemented by van der Wilk et al. (2017).

Depending on the experiment, we use a suitable GPflow likelihood: Gaussian for regression, Bernoulli for binary classification, and RobustMax for multi-class classification.

Model setup specific to W-SVGP:

- The variational mean $\tilde{\mathbf{m}}$ is initialised to $\tilde{\mathbf{m}} = \mathbf{0}$;
- The variational covariance $\tilde{\mathbf{S}}$ is parameterised via its Cholesky factor $\tilde{\mathbf{L}}$ with $\tilde{\mathbf{S}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top$, and $\tilde{\mathbf{L}}$ is initialised to $\tilde{\mathbf{L}} = \mathbf{I}$.

Model setup specific to L-SVGP:

- Apart from the efficacy experiments on toy datasets (Section 4.1), we only consider the preconditioned version of the bound (Section 3.2);
- The variational mean $\tilde{\mathbf{m}}$ is initialised to $\tilde{\mathbf{m}} = \mathbf{0}$;
- The variational covariance $\tilde{\mathbf{S}}$ is parameterised via its diagonal $\tilde{\mathbf{s}}$ with $\tilde{\mathbf{S}} = \mathrm{diag}(\tilde{\mathbf{s}})$, and $\tilde{\mathbf{s}}$ is initialised to $\tilde{\mathbf{s}} = \alpha\mathbf{1}$ with $\alpha = 10^{-4}$.

Model setup specific to R-SVGP:

- Apart from the efficacy experiments on toy datasets (Section 4.1), we only consider the preconditioned version of the bound (Section 3.2);
- The variational parameters $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{S}}$ are initialised as for L-SVGP;
- The matrix $\mathbf{T}$ is parameterised via its Cholesky factor $\mathbf{L}$ with $\mathbf{T} = \mathbf{L}\mathbf{L}^\top$, and $\mathbf{L}$ is initialised to $\mathbf{L} = \beta\mathbf{I}$ with $\beta = 10^{-3}$.
- For the efficiency experiments on the UCI datasets (Section 4.2), we estimate the trace terms in the bound using $K = 256$ random probe vectors, as discussed in Section 3.4.

The number M of inducing points and the initialisation of the corresponding inducing location $\mathbf{Z}$ depend on the experiment performed:

[4] https://www.openml.org/search?type=data
[5] https://keras.io/api/datasets

- Shallow regression/classification on toy datasets (Section 4.1): $M = 10$ for SNELSON, initialised on a uniform grid spanning the range of the training inputs, and $M = 64$ for BANANA, initialised using k-means++ (Arthur and Vassilvitskii, 2007) on the training inputs;
- Shallow regression on UCI datasets (Section 4.2): M ranging from 1000 to 4000 for both ELEVATORS and KIN40K, initialised using k-means++ on the training inputs;
- Deep regression on KIN40K (Section 4.3): $M = 500$ for each layer, initialised using the GPflux default (based on k-means++);
- Multi-class classification on MNIST (Section 4.3): $M = 750$ initialised by sampling uniformly without replacement from the training inputs.

As discussed in Section 4.3, for deep GP experiments, we range the number of layers from 1 to 3, with each intermediate layer ℓ having $Q^{(\ell)} = D$ outputs, where D is the input dimension of the dataset. Kernel hyperparameters and inducing locations are shared between outputs within each layer, as per GPflux defaults. As discussed in Section 4.3, for deep GPs based on L-SVGP and R-SVGP, we also tie the variational covariance matrices $\tilde{\mathbf{S}}$ within each layer to match the cost of W-SVGP-based models.

S5.4 Training

Training details are specific to each experiment, and are described below.

Shallow Regression/Classification on Toy Datasets (Section 4.1). We keep the inducing locations $\mathbf{Z}$ fixed during training, use a mini-batch size B equal to the number of inducing points M, and perform 10000 training iterations. The learning rate of Adam is kept fixed during training to 5×10^{-3} for the SNELSON dataset, and to 1×10^{-2} for the BANANA dataset. For $\mathbf{T}$-optimisation in R-SVGP, a single NG update with step size 1.0 is used at every training iteration.

Shallow Regression on UCI Datasets (Section 4.2). We use a mini-batch size $B = 100$ and perform 20000 training iterations. The learning rate of Adam is kept fixed during training to 5×10^{-3}. For $\mathbf{T}$-optimisation in R-SVGP, we take advantage of the training heuristics discussed in Section 3.4. In particular, $\mathbf{Z}$ is kept fixed for the first 1000 training iterations, before being trained jointly with the other parameters using a $\mathbf{Z}$-specific Adam configuration with $\beta_1 = 0.99$ and learning rate starting 1×10^{-3} and multiplied by 0.95 every time a plateau in the training loss is reached for 100 iterations. As a result of these heuristics, $\mathbf{T}$ can be optimised using a single NG update with step size 1.0 at every training iteration.

Deep Regression (Section 4.3). We use a mini-batch size $B = 1000$ and perform 200000 training iterations. The learning rate of Adam is initialised to 5×10^{-3} and multiplied by 0.95 every time a plateau in the training loss is reached for 1000 iterations. For $\mathbf{T}$-optimisation in R-SVGP, we use the stopping criterion (8) in Section 3.3 with tolerance $\epsilon = 10^{-9}$ and a log-linear step-size schedule (Salimbeni et al., 2018) starting from 1×10^{-5} and increasing to 1.0 over the first 10 NG updates.

Multi-class Classification (Section 4.3). We follow the training setup of van der Wilk et al. (2017). In particular, we use a mini-batch size $B = 200$ and perform 450000 training iterations. The learning rate of Adam is initialised to 10^{-3} and annealed using a piecewise-constant schedule with boundaries every 30000 iterations and values multiplied by $10^{-1/3} \approx 0.464$ at each boundary. For $\mathbf{T}$-optimisation in R-SVGP, we use the stopping criterion (8) in Section 3.3 with tolerance $\epsilon = 10^{-6}$ and a log-linear step-size schedule starting from 1×10^{-5} and increasing to 1.0 over the first 10 NG updates.

S6 ADDITIONAL EXPERIMENTS

S6.1 Approximate Bound Evaluation via Hutchinson's Method

This section examines the runtime–accuracy trade-off of using Hutchinson's method to estimate the trace terms in the R-SVGP bound, as discussed in Section 3.4. To this end, we consider the ELEVATORS and KIN40K UCI

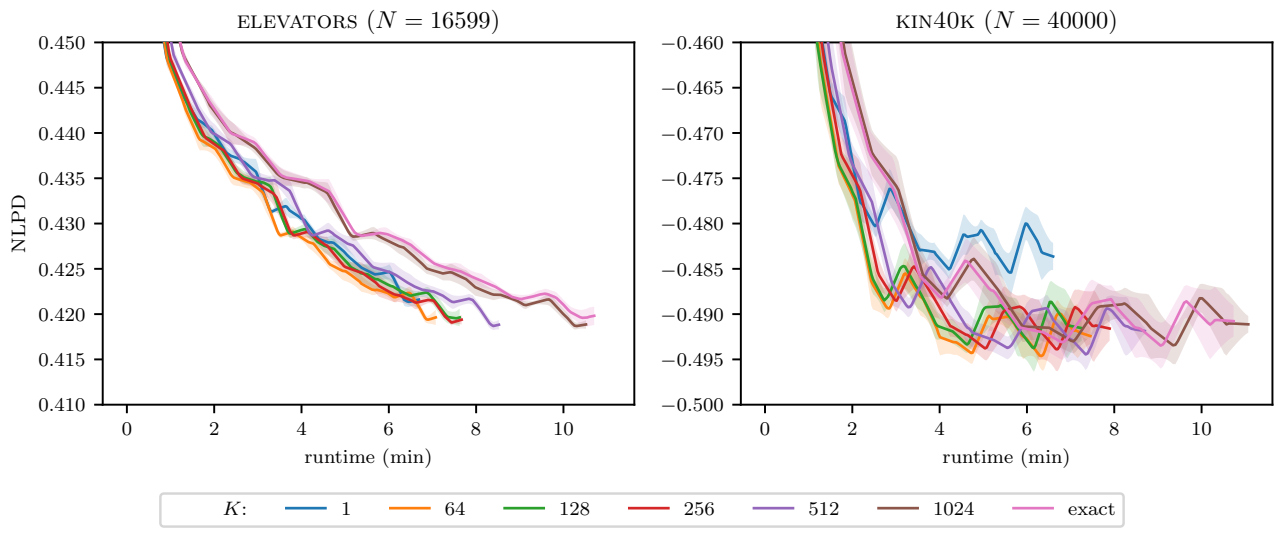

Figure S1: NLPD/runtime on ELEVATORS and KIN40K datasets for different choices of the number of probes K, with $M = 2000$ and $B = 100$. Lines show the mean over 5 seeds; shaded regions indicate ± 1 standard error.

regression datasets, and train R-SVGP using the same setup as in Section 4.2 with $M = 2000$ inducing points and batch size $B = 100$, while varying the number of probe vectors K used for trace estimation. Figure S1 tracks the negative log-predictive density (NLPD) against training time on both datasets for different choices of K. As expected, using a single probe vector ($K = 1$) leads to noticeably noisier optimisation, especially on KIN40K, and to worse final NLPD. In contrast, for $K \geq 64$, the final NLPD after training is very similar between runs using approximate and exact trace estimation, while runtime is substantially reduced relative to larger-K runs. This indicates that Hutchinson's method can provide stable training and good predictive performance with a moderate number of probe vectors in this setting. For instance, the choice $K = 256$ used in Section 4.2 reduces total runtime by approximately 25–30% relative to exact trace estimation.