Fast Matrix Square Roots with Applications to Gaussian Processes and Bayesian Optimization

Geoff Pleiss
Columbia University
gmp2162@columbia.edu

Martin Jankowiak
The Broad Institute
mjankowi@broadinstitute.org

David Eriksson[*]
Facebook
deriksson@fb.com

Anil Damle
Cornell University
damle@cornell.edu

Jacob R. Gardner
University of Pennsylvania
jacobrg@seas.upenn.edu

Abstract

Matrix square roots and their inverses arise frequently in machine learning, e.g., when sampling from high-dimensional Gaussians $\mathcal{N}(\mathbf{0}, \mathbf{K})$ or "whitening" a vector $\mathbf{b}$ against covariance matrix $\mathbf{K}$. While existing methods typically require $\mathcal{O}(N^3)$ computation, we introduce a highly-efficient quadratic-time algorithm for computing $\mathbf{K}^{1/2}\mathbf{b}$, $\mathbf{K}^{-1/2}\mathbf{b}$, and their derivatives through *matrix-vector multiplication* (MVMs). Our method combines Krylov subspace methods with a rational approximation and typically achieves 4 decimal places of accuracy with fewer than 100 MVMs. Moreover, the backward pass requires little additional computation. We demonstrate our method's applicability on matrices as large as $50,000 \times 50,000$—well beyond traditional methods—with little approximation error. Applying this increased scalability to variational Gaussian processes, Bayesian optimization, and Gibbs sampling results in more powerful models with higher accuracy. In particular, we perform variational GP inference with up to 10,000 inducing points and perform Gibbs sampling on a 25,000-dimensional problem.

1 Introduction

High-dimensional Gaussian distributions arise frequently in machine learning, especially in the context of Bayesian modeling. For example, the prior of Gaussian process models is given by a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{K})$ governed by an $N \times N$ symmetric positive definite kernel matrix $\mathbf{K}$. Historically, $\mathcal{O}(N^3)$ computation and $\mathcal{O}(N^2)$ memory requirements have limited the tractability of inference for high-dimensional Gaussian latent variable models.

A growing line of research aims to reformulate many common covariance matrix operations—such as linear solves and log determinants—as iterative optimizations involving *matrix-vector multiplications* (MVMs) [e.g. 3, 11, 16, 29, 79]. MVM approaches have two primary advantages: 1) the covariance matrix need not be explicitly instantiated (so only $\mathcal{O}(N)$ memory is required) [11, 16, 79]; and 2) MVMs utilize GPU acceleration better than direct methods like Cholesky [3, 29]. Thus MVM methods can be scaled to much larger covariance matrices.

In this paper, we propose an MVM method that addresses a common computational bottleneck for high-dimensional Gaussians: computing $\mathbf{K}^{\pm 1/2}\mathbf{b}$. This operation occurs frequently in Gaussian process models and inverse problems. For example, if $\mathbf{b} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then $\mathbf{K}^{\frac{1}{2}}\mathbf{b} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$. This operation appears frequently in Bayesian optimization [e.g. 23, 42, 74, 80] and Gibbs sampling [e.g.

---

[*]This work was conducted while David Eriksson was at Uber AI.

6, 8, 33]. $\mathbf{K}^{-\frac{1}{2}}\mathbf{b}$ can be used to project parameters into a "whitened" coordinate space [50, 54]—a transformation that accelerates the convergence of variational Gaussian process approximations. To make these computations more efficient and scalable, we make the following contributions:

- We introduce a MVM approach for computing $\mathbf{K}^{\pm 1/2}\mathbf{b}$. The approach uses an insight from Hale et al. [35] that expresses the matrix square root as a sum of shifted matrix inverses.

- To efficiently compute these shifted inverses, we leverage a modified version of the MINRES algorithm [59] that performs *multiple shifted solves* through a single iteration of MVMs. We demonstrate that, surprisingly, **multi-shift MINRES (msMINRES)** convergence can be accelerated with a *single* preconditioner despite the presence of multiple shifts. Moreover, msMINRES only requires $\mathcal{O}(N)$ storage when used in conjunction with partitioned MVMs [11, 79]. Achieving 4 or 5 decimal places of accuracy typically requires *fewer than 100 matrix-vector multiplications*, which can be highly accelerated through GPUs.

- We derive a scalable backward pass for $\mathbf{K}^{\pm 1/2}\mathbf{b}$ that enables our approach to be used as part of learning and optimization.

- We apply our $\mathbf{K}^{-1/2}\mathbf{b}$ and $\mathbf{K}^{1/2}\mathbf{b}$ routines to three applications: 1) variational Gaussian processes with up to $M = 10^4$ inducing points (where we additionally introduce a $\mathcal{O}(M^2)$ MVM-based natural gradient update); 2) sampling from Gaussian process posteriors in the context of Bayesian optimization with up to 50,000 candidate points; and 3) an image reconstruction task where we perform Gibbs sampling in 25,600 dimensions.

Code examples for the GPyTorch framework are available at `bit.ly/ciq_svgp` and `bit.ly/ciq_sampling`.

2 Background

Existing Methods for Sampling and Whitening typically rely on the Cholesky factorization: $\mathbf{K} = \mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is lower triangular. Though $\mathbf{L}$ is not a square root of $\mathbf{K}$, $\mathbf{L}\mathbf{b}$ is equivalent to $\mathbf{K}^{1/2}\mathbf{b}$ up to an orthonormal rotation. Therefore, $\mathbf{L}\epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ can be used to draw samples from from $\mathcal{N}(\mathbf{0}, \mathbf{K})$ and $\mathbf{L}^{-1}\mathbf{b}$ can be used to "whiten" the vector $\mathbf{b}$. However, the Cholesky factor requires $\mathcal{O}(N^3)$ computation and $\mathcal{O}(N^2)$ memory for an $N \times N$ covariance matrix $\mathbf{K}$. To avoid this large complexity, randomized algorithms [58, 63], low-rank/sparse approximations [41, 61, 83], or alternative distributions [80] are often used to approximate the sampling and whitening operations.

Krylov Subspace Methods are a family of iterative algorithms for computing functions of matrices applied to vectors $f(\mathbf{K})\mathbf{b}$ [e.g. 65, 67, 77]. Crucially, $\mathbf{K}$ is only accessed through *matrix-vector multiplication* (MVM), which is beneficial for extremely large matrices that cannot be explicitly computed in memory. All Krylov algorithms share the same basic structure: each iteration j produces an estimate $\mathbf{c}_j \approx f(\mathbf{K})\mathbf{b}$ which falls within the jth *Krylov subspace* of $\mathbf{K}$ and $\mathbf{b}$:

$$\mathbf{c}_j \in \mathcal{K}_j(\mathbf{K}, \mathbf{b}) = \text{span}\left\{\mathbf{b}, \ \mathbf{K}\mathbf{b}, \ \mathbf{K}^2\mathbf{b}, \ \dots, \ \mathbf{K}^{j-1}\mathbf{b}\right\}. \tag{1}$$

Each iteration expands the Krylov subspace by one vector, requiring a single matrix-vector multiplication with $\mathbf{K}$. Many Krylov methods, such as linear conjugate gradients, can be reduced to computationally efficient vector recurrences. Krylov methods are exact after N iterations, though most methods offer extremely accurate solutions in $J \ll N$ iterations. There has been growing interest in applying Krylov methods to large-scale kernel methods [3, 4, 11, 13, 15, 16, 29, 30, 57, 61, 65, 70, 71, 82], especially due to their memory efficiency and amenability to GPU acceleration.

3 Contour Integral Quadrature (CIQ) via Matrix-Vector Multiplication

In this section we develop an MVM method to compute $\mathbf{K}^{-1/2}\mathbf{b}$ and $\mathbf{K}^{1/2}\mathbf{b}$ for sampling and whitening. Our approach scales better than existing methods (e.g. Cholesky) by: 1) reducing computation from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$; 2) reducing memory from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$; 3) more effectively using GPU acceleration; and 4) affording an efficient gradient computation.

Contour Integral Quadrature (CIQ). A standard result from complex analysis is that $\mathbf{K}^{-1/2}$ can be expressed through Cauchy's integral formula: $\mathbf{K}^{-1/2} = \frac{1}{2\pi i} \oint_\Gamma \tau^{-1/2} (\tau \mathbf{I} - \mathbf{K})^{-1} \, \mathrm{d}\tau$, where Γ

is a closed contour in the complex plane that winds once around the spectrum of $\mathbf{K}$ [18, 35, 44]. Applying a numerical quadrature scheme to the contour integral yields the rational approximations

$$\mathbf{K}^{-\frac{1}{2}} \approx \sum_{q=1}^{Q} w_q \left(t_q \mathbf{I} + \mathbf{K}\right)^{-1} \quad \text{and} \quad \mathbf{K}^{\frac{1}{2}} \approx \mathbf{K} \sum_{q=1}^{Q} w_q \left(t_q \mathbf{I} + \mathbf{K}\right)^{-1}, \tag{2}$$

where the weights w_q encapsulate the normalizing constant, quadrature weights, and the $t_q^{-\frac{1}{2}}$ terms. Hale et al. [35] introduce a real-valued quadrature strategy based on a change-of-variables formulation (described in Appx. B) that converges extremely rapidly—often achieving full machine precision with only $Q \approx 20$ quadrature points. For the remainder of this paper, applying Eq. (2) to compute $\mathbf{K}^{\pm 1/2}\mathbf{b}$ will be referred to as **Contour Integral Quadrature (CIQ)**.

3.1 An Efficient Matrix-Vector Multiplication Approach to CIQ with msMINRES.

Using the quadrature method of Eq. (2) for whitening and sampling requires solving several shifted linear systems. To compute the shifted solves required by Eq. (2) we leverage a variant of the minimum residuals algorithm (MINRES) developed by Paige and Saunders [59]. At step j MINRES approximates $\mathbf{K}^{-1}\mathbf{b}$ by the vector within the Krylov subspace $\mathbf{c} \in \mathcal{K}_j(\mathbf{K}, \mathbf{b})$ that minimizes the residual $\|\mathbf{K}\mathbf{c} - \mathbf{b}\|_2$.

msMINRES for multiple shifted solves. To efficiently compute all the shifted solves, we leverage techniques [e.g. 14, 17, 24, 25, 55] that exploit the shift-invariance property of Krylov subspaces: i.e. $\mathcal{K}_J(\mathbf{K}, \mathbf{b}) = \mathcal{K}_J(t\mathbf{I}+\mathbf{K}, \mathbf{b})$. We introduce a variant to MINRES, which we refer to as **multi-shift MINRES** or **msMINRES**, that re-uses the same Krylov subspace vectors $[\mathbf{b}, \mathbf{K}\mathbf{b}, \ldots, \mathbf{K}^{J-1}\mathbf{b}]$ for all shifted solves $(t\mathbf{I} + \mathbf{K})^{-1}\mathbf{b}$. In other words, using msMINRES we can get all $(t_q\mathbf{I} + \mathbf{K})^{-1}\mathbf{b}$ *essentially for free*, i.e. only requiring J MVMs for the Krylov subspace $\mathcal{K}_J(\mathbf{K}, \mathbf{b})$. As with standard MINRES, the msMINRES procedure for computing $(t_q\mathbf{I} + \mathbf{K})^{-1}$ from $[\mathbf{b}, \mathbf{K}\mathbf{b}, \ldots, \mathbf{K}^{J-1}\mathbf{b}]$ can be reduced to a simple vector recurrence (see Appx. C for details).

3.2 Computational Complexity and Convergence Analysis of msMINRES-CIQ

Pairing Eq. (2) with msMINRES is an efficient algorithm for computing $\mathbf{K}^{1/2}\mathbf{b}$ and $\mathbf{K}^{-1/2}\mathbf{b}$. Alg. 1 (see Appendix) summarizes this approach; below we highlight its computational properties:

Property 1 (Computation/Memory of msMINRES-CIQ). *J iterations of msMINRES requires exactly J MVMs with the input matrix $\mathbf{K}$, regardless of the number of quadrature points Q. The resulting runtime of msMINRES-CIQ is $\mathcal{O}(J\xi(\mathbf{K}))$, where $\xi(\mathbf{K})$ is the time to perform an MVM with $\mathbf{K}$. The memory requirement is $\mathcal{O}(QN)$ in addition to what is required to store $\mathbf{K}$.*

For arbitrary positive semi-definite $N \times N$ matrices, the runtime of msMINRES-CIQ is $\mathcal{O}(JN^2)$, where often $J \ll N$. Performing the MVMs in a map-reduce fashion [11, 79] avoids explicitly forming $\mathbf{K}$, which results in $\mathcal{O}(QN)$ total memory. This is in contrast to Cholesky, which produces an artifact that requires $\mathcal{O}(N^2)$ memory. Below we bound the error of msMINRES-CIQ:

Theorem 1. *Let $\mathbf{K} \succ 0$ and $\mathbf{b}$ be inputs to msMINRES-CIQ, producing $\mathbf{a}_J \approx \mathbf{K}^{1/2}\mathbf{b}$ after J iterations with Q quadrature points. The difference between $\mathbf{a}_J$ and $\mathbf{K}^{1/2}\mathbf{b}$ is bounded by:*

$$\left\|\mathbf{a}_J - \mathbf{K}^{\frac{1}{2}}\mathbf{b}\right\|_2 \leq \overbrace{\mathcal{O}\left(\exp\left(-\frac{2Q\pi^2}{\log \kappa(\mathbf{K})+3}\right)\right)}^{\textit{Quadrature error}} + \overbrace{\frac{2Q \log\left(5\sqrt{\kappa(\mathbf{K})}\right)\kappa(\mathbf{K})\sqrt{\lambda_{min}}}{\pi}\left(\frac{\sqrt{\kappa(\mathbf{K})}-1}{\sqrt{\kappa(\mathbf{K})}+1}\right)^{J-1}}^{\textit{msMINRES error}} \|\mathbf{b}\|_2.$$

where $\lambda_{max}, \lambda_{min}$ are the max and min eigenvalues of $\mathbf{K}$, and $\kappa(\mathbf{K}) \equiv \frac{\lambda_{max}}{\lambda_{min}}$ is the condition number.

For $\mathbf{a}'_J \approx \mathbf{K}^{-1/2}\mathbf{b}$, the bound incurs an additional factor of $1/\lambda_{\min}$. (See Appx. G for proofs.) Thm. 1 suggests that error in computing $(t_q\mathbf{I} + \mathbf{K})^{-1}\mathbf{b}$ will be the primary source of error as the quadrature error decays rapidly with Q. In many of our applications the rapid convergence of Krylov subspace methods for linear solves is well established, allowing for accurate solutions if desired. For covariance matrices up to $N = 50{,}000$, often $Q = 8$ and $J \leq 100$ suffices for 4 decimal places of accuracy and J can be further reduced with preconditioning (see Sec. 4 and Appx. D).

3.3 Efficient Vector-Jacobi Products for Backpropagation

In certain applications, such as variational Gaussian process inference, we have to compute gradients of the $\mathbf{K}^{-1/2}\mathbf{b}$ operation. This requires the vector-Jacobian product $\mathbf{v}^\top (\partial \mathbf{K}^{-1/2}\mathbf{b}/\partial \mathbf{K})$, where $\mathbf{v}$ is the back-propagated gradient. The form of the Jacobian is the solution to a Lyapunov equation, which requires expensive iterative methods or solving a $N^2 \times N^2$ Kronecker sum $(\mathbf{K}^{1/2} \oplus \mathbf{K}^{1/2})^{-1}$. Both of these options are much slower than the forward pass and are impractical for large N. Fortunately, our quadrature formulation affords a computationally efficient approximation to this vector-Jacobian product. If we back-propagate directly through each term in Eq. (2), we have

$$\mathbf{v}^\top \left(\frac{\partial \mathbf{K}^{-1/2}\mathbf{b}}{\partial \mathbf{K}} \right) \approx -\frac{1}{2} \sum_{q=1}^{Q} w_q \, (t_q \mathbf{I} + \mathbf{K})^{-1} \left(\mathbf{v}\mathbf{b}^\top + \mathbf{b}\mathbf{v}^\top \right) (t_q \mathbf{I} + \mathbf{K})^{-1} . \qquad (3)$$

Since the forward pass computes the solves with $\mathbf{b}$, the only additional work needed for the backward pass is computing the shifted solves $(t_q \mathbf{I} + \mathbf{K})^{-1}\mathbf{v}$, which can be computed with another call to the msMINRES algorithm. Thus the backward pass takes only $\mathcal{O}(J\xi(\mathbf{K}))$ (e.g. $\mathcal{O}(JN^2)$) time.

3.4 Preconditioning

Preconditioners are commonly applied to Krylov subspace methods like MINRES to improve the condition number $\kappa(\mathbf{K})$ and accelerate convergence. However, standard preconditioning techniques do not apply to msMINRES, as each shifted system $\mathbf{K} + t_q\mathbf{I}$ requires its own preconditioner (see Appx. D for details). Each separately preconditioned system would require separate MVMs, defeating the efficiency of msMINRES. Nevertheless, we can use a single preconditioner to compute rotationally-equivalent solutions to $\mathbf{K}^{\pm 1/2}\mathbf{b}$. If $\mathbf{P} \approx \mathbf{K}$ is a preconditioner matrix, we note that:

$$\mathbf{K}\mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{b}, \qquad \mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{b}$$

are equivalent to $\mathbf{K}^{1/2}\mathbf{b}$ and $\mathbf{K}^{-1/2}\mathbf{b}$ (respectively) up to an orthonormal rotation (see Appx. D). We can use msMINRES-CIQ to compute the $(\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2})^{-1/2}\mathbf{b}$ terms. Crucially, the convergence now depends on the condition number of $\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2}$, rather than that of $\mathbf{K}$.

3.5 Related Work

Other Krylov methods for $\mathbf{K}^{1/2}\mathbf{b}$ and $\mathbf{K}^{-1/2}\mathbf{b}$, often via polynomial approximations [e.g. 44], have been explored. Chow and Saad [13] compute $\mathbf{K}^{1/2}\mathbf{b}$ via a preconditioned Lanczos algorithm. Unlike msMINRES, however, they require storage of the entire Krylov subspace. Moreover this approach does not afford a simple gradient computation. Frommer et al. [26, 27] apply a similar Krylov/quadrature approach to a broad class of matrix functions. More similar to our work is [3, 4], which uses the quadrature formulation of Eq. (2) in conjunction with a shifted conjugate gradients solver. We expand upon their method by: 1) introducing a simple gradient computation; 2) proving a convergence guarantee; and 3) enabling the use of simple preconditioners (see Appx. D).

4 Benchmarking msMINRES-CIQ

In this section we empirically measure the convergence and speedup of msMINRES-CIQ applied to several types of covariance matrices.

Convergence of msMINRES-CIQ. In Fig. 1 we measure the relative error of computing $\mathbf{K}^{1/2}\mathbf{b}$ with msMINRES-CIQ on random matrices.[2] We vary 1) the number of quadrature points Q; 2) the size of the matrix N; and 3) the conditioning of the matrix. The left and middle plots display results for matrices with spectra that decay as $\lambda_t = 1/\sqrt{t}$ and $\lambda_t = 1/t^2$, respectively. The right plot displays results for one-dimensional Matérn kernel matrices (formed with random data), which have near-exponentially decaying spectra. Consequently, the $1/\sqrt{t}$ matrices are relatively well-conditioned, while the Matérn kernels are relatively ill-conditioned. Nevertheless, in all cases CIQ achieves 10^{-4} relative error with only $Q = 8$ quadrature points, regardless of the size of the matrix.

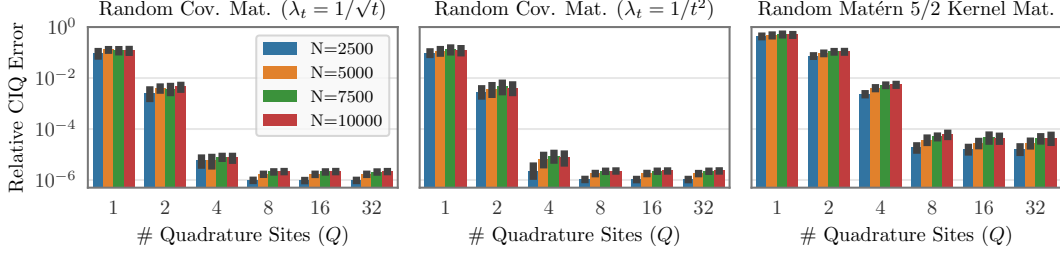[2]msMINRES is stopped after achieving a relative residual of 10^{-4} or after reaching $J = 400$ iterations.

Figure 1: msMINRES-CIQ relative error when computing $\mathbf{K}^{1/2}\mathbf{b}$ as a function of number of quadrature sites Q. We test random matrices with eigenvalues that scale as $\lambda_t = 1/\sqrt{t}$ (left) and $\lambda_t = 1/t^2$ (middle), as well as Matérn kernels (right). In all cases $Q = 8$ achieves $< 10^{-4}$ error. The error levels out at roughly 10^{-4} or 10^{-5}, which corresponds to the msMINRES tolerance. msMINRES is stopped after achieving a relative residual of 10^{-4} or $J = 400$ iterations.
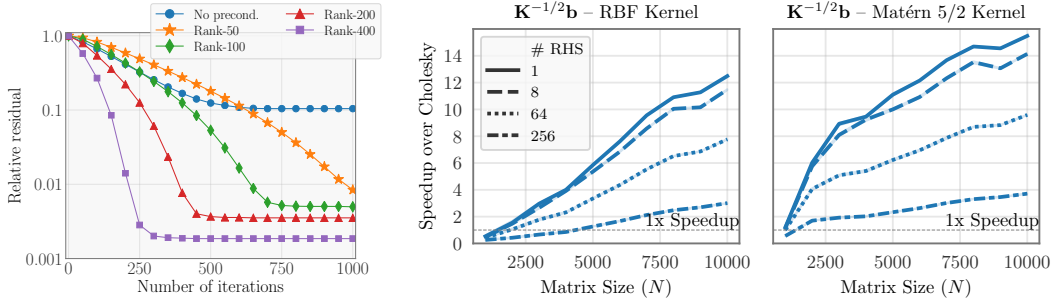

Figure 2: (**Left**:) Effect of preconditioning on msMINRES-CIQ convergence while performing Bayesian optimization. Samples are drawn from the $N = 50,000$ posterior covariance matrix of the ill-conditioned 6-dimensional Hartmann function (see Sec. 5.2), using the pivoted Cholesky preconditioner [29]. (**Middle/Right**:) Speedup of msMINRES-CIQ over Cholesky when computing forward/backward passes of $\mathbf{K}^{-1/2}\mathbf{b}$ with varying number of right-hand-sides $\mathbf{b}$ (RHS).

Additionally, Appx. A demonstrates that msMINRES-CIQ achieves orders of magnitude smaller error than approximation algorithms like randomized SVD [36] or random Fourier features [63].

To demonstrate the effect of preconditioning, we construct a posterior covariance matrix of size $N = 50,000$ points on the 6 dimensional Hartmann function (see Sec. 5.2 for a description). We note that this problem is particularly ill-conditioned ($\kappa(\mathbf{K}) \approx 10^8$), and thus represents an extreme test case. Fig. 2 (left) plots the convergence of msMINRES-CIQ (computing $\mathbf{K}^{1/2}\mathbf{b}$). Without preconditioning, it is difficult to achieve relative residuals less than 0.1. Using the pivoted Cholesky preconditioner of Gardner et al. [29]—a low-rank approximation of $\mathbf{K}$—not only accelerates the convergence but also reduces the final residual. With rank-200/rank-400 preconditioners, the final residual is cut by orders of magnitude, and msMINRES-CIQ converges $2\times/4\times$ faster.

Speedup over Cholesky. We compare the wall-clock speedup of msMINRES-CIQ over Cholesky in Fig. 2 (middle/right) on RBF/Matérn kernels.[3] We compute $\mathbf{K}^{-1/2}\mathbf{b}$ and its derivative on multiple right-hand-side (RHS) vectors. As N increases, msMINRES-CIQ incurs a larger speedup (up to $15\times$ faster than Cholesky). This speedup is less pronounced when computing many RHSs simultaneously, as the cubic complexity of Cholesky is amortized across each RHS. Nevertheless, msMINRES-CIQ is advantageous for matrices larger than $N = 3,000$ even when simultaneously whitening 256 vectors.

5   Applications

In previous sections we showed, both theoretically and empirically, that msMINRES-CIQ accurately computes $\mathbf{K}^{\pm 1/2}\mathbf{b}$ while scaling better than traditional (Cholesky-based) methods. In this section we

---

[3]$Q = 8$. msMINRES is stopped after a residual of 10^{-4}. Kernels are formed using data from the Kin40k dataset [2]. Timings are performed on a NVIDIA 1070 GPU.
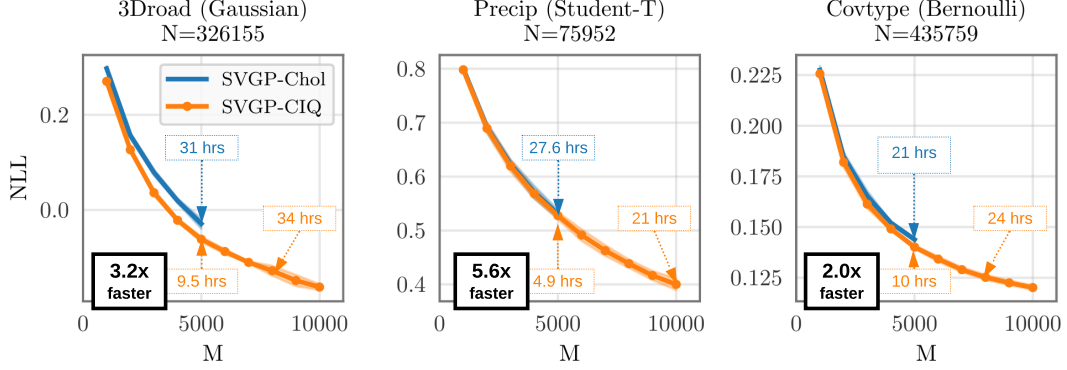
Figure 3: Negative log likelihood (NLL) comparison of Cholesky vs CIQ SVGP models. **Left:** 3DRoad dataset ($N = 326155, D = 2$, Gaussian likelihood). **Middle:** Precipitation dataset ($N = 75952, D = 3$, Student-T likelihood). **Right:** CoverType dataset ($N = 435759, D = 54$, Bernoulli likelihood). NLL improves with more inducing points (M), and Cholesky and msMINRES-CIQ models have similar performance. However CIQ models train faster than their Cholesky counterparts.

demonstrate applications of this increased speed and scalability. In particular, we show that using msMINRES-CIQ in conjunction with variational Gaussian processes, Bayesian optimization, and Gibbs sampling facilitates higher-fidelity models that can be applied to large-scale problems.

5.1 Whitened Stochastic Variational Gaussian Processes

As a first application, we demonstrate that the msMINRES-CIQ whitening procedure $\mathbf{K}^{-1/2}\mathbf{b}$ can increase the fidelity of **stochastic variational Gaussian processes (SVGP)** [39, 40, 54]. These models are used for non-conjugate likelihoods (e.g. binary classification) or for large datasets that do not fit into memory. SVGP forms an approximate posterior $p(f(\mathbf{x}) \mid \mathbf{X}, \mathbf{y}) \approx q(f(\mathbf{x})) = \mathbb{E}_{q(\mathbf{u})}\left[p\left(f(\mathbf{x}) \mid \mathbf{u}\right)\right]$, where $\mathbf{u} \in \mathbb{R}^M$ are inducing function values (see [40, 54] for a detailed derivation). $q(\mathbf{u})$ is a Gaussian variational distribution parameterized by mean $\mathbf{m} \in \mathbb{R}^M$ and covariance $\mathbf{S} \in \mathbb{R}^{M \times M}$. $\mathbf{m}$ and $\mathbf{S}$ (as well as the model's kernel/likelihood hyperparameters) are chosen to maximize the variational ELBO:

$$\mathcal{L}_{\text{ELBO}}\left\{q(\mathbf{u}) = \mathcal{N}\left(\mathbf{m}, \mathbf{S}\right)\right\} = \sum_{i=1}^{N} \mathbb{E}_{q(f(\mathbf{x}^{(i)}))}\left[\log p(y^{(i)} \mid f(\mathbf{x}^{(i)}))\right] - \text{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right].$$

Rather than directly learning $\mathbf{m}$ and $\mathbf{S}$, it is more common to learn the *whitened parameters* [50, 54]: $\mathbf{m}' = \mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{m}$ and $\mathbf{S}' = \mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{S}\mathbf{K}_{\mathbf{ZZ}}^{-1/2}$. Under these coordinates, the KL divergence term is $\frac{1}{2}(\mathbf{m}'^{\top}\mathbf{m}' + \text{Tr}(\mathbf{S}') - \log |\mathbf{S}'| - M)$, which doesn't depend on $p(\mathbf{u})$ and therefore is relatively simple to optimize. The posterior distribution $q(f(\mathbf{x})) = \mathcal{N}(\mu_{\text{aprx}}^*(\mathbf{x}), \text{Var}_{\text{aprx}}^*(\mathbf{x}))$ is given by

$$\mu_{\text{aprx}}^*(\mathbf{x}) = \mathbf{k}_{\mathbf{Zx}}^{\top}\mathbf{K}_{\mathbf{ZZ}}^{-\frac{1}{2}}\mathbf{m}', \quad \text{Var}_{\text{aprx}}^*(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{\mathbf{Zx}}^{\top}\mathbf{K}_{\mathbf{ZZ}}^{-\frac{1}{2}}\left(\mathbf{I} - \mathbf{S}'\right)\mathbf{K}_{\mathbf{ZZ}}^{-\frac{1}{2}}\mathbf{k}_{\mathbf{Zx}}. \quad (4)$$

Time and space complexity. During training, we repeatedly compute the ELBO and its derivative, which requires computing Eq. (4) and its derivative for a minibatch of data points. Optimization typically requires up to 10,000 iterations of training [e.g. 66]. We note that $\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{b}$ (and its derivative) is the most expensive numerical operation during each ELBO computation. If we use Cholesky to compute this operation, the time complexity of SVGP training is $\mathcal{O}(M^3)$.[4] On the other hand, msMINRES-CIQ-based SVGP training is only $\mathcal{O}(JM^2)$, where J is the number of msMINRES iterations. Both methods require $\mathcal{O}(M^2)$ storage for the $\mathbf{m}'$ and $\mathbf{S}'$ parameters.

Natural gradient descent with msMINRES-CIQ. The size of the variational parameters $\mathbf{m}'$ and $\mathbf{S}'$ grows quadratically with M. This poses a challenging optimization problem for standard gradient descent methods. To adapt to the large M regime, we rely on **natural gradient descent (NGD)** to optimize $\mathbf{m}'$ and $\mathbf{S}'$ [e.g. 38, 66]. At a high level, these methods perform the updates $[\mathbf{m}, \ \mathbf{S}] \leftarrow [\mathbf{m}, \ \mathbf{S}] - \varphi \, \boldsymbol{\mathcal{F}}^{-1} \, \nabla\mathcal{L}_{\text{ELBO}}$, where φ is a step size, $\nabla\mathcal{L}_{\text{ELBO}}$ is the ELBO gradient, and $\boldsymbol{\mathcal{F}}$ is the

---

[4]Note that Cholesky computes $\mathbf{K}^{-1/2}\mathbf{b}$ up to an orthogonal rotation, which is suitable for whitened SVGP.
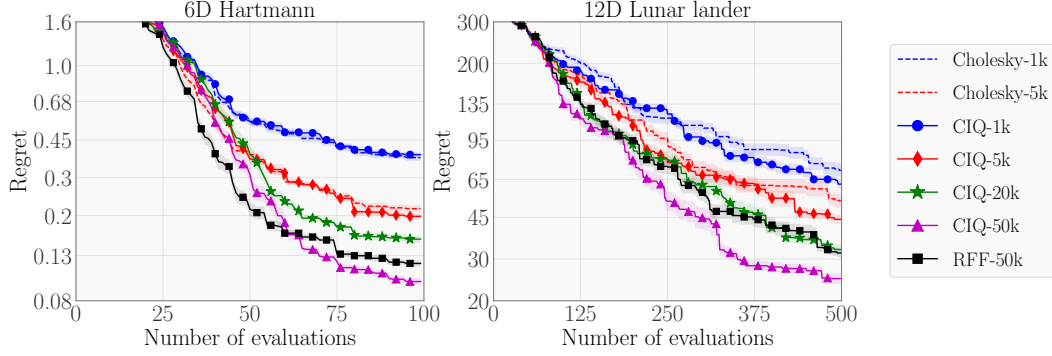
Figure 4: A comparison of sampling methods for Bayesian Optimization. BO is applied to the **(left)** Hartmann ($D = 6$) and **(right)** Lunar Lander ($D = 12$) problems. Methods: Cholesky-$\langle T \rangle$ draws posterior samples with Cholesky at T candidate points. CIQ-$\langle T \rangle$ draws posterior samples with msMINRES-CIQ. RFF-50k uses random Fourier features to draw approximate posterior samples at 50,000 candidate points. Larger T results in better optimization. msMINRES-CIQ enables scaling to $T \geq 50,000$. Each plot shows mean regret with standard error in log-scale based on 30 replications.

Fisher information matrix of the variational parameters. Naïvely, each NGD step requires $\mathcal{O}(M^3)$ computations with $\mathbf{m}'$ and $\mathbf{S}'$, which would dominate the cost of CIQ-based SVGP. Fortunately, we can derive a natural gradient update that only relies on matrix solves with $\mathbf{S}'$, which take $\mathcal{O}(JM^2)$ time using preconditioned conjugate gradients. Therefore, using NGD incurs the same *quadratic* asymptotic complexity as msMINRES-CIQ. See Appx. E for the $\mathcal{O}(M^2)$ NGD update equations.

Cholesky vs msMINRES-CIQ. We compare msMINRES-CIQ-SVGP against Cholesky-SVGP on 3 large-scale datasets: a GIS dataset (**3droad**, $D = 2$) [34], a monthly precipitation dataset (**Precipitation**, $D = 3$) [52, 53], and a tree cover dataset (**Covtype**, $D = 54$) [9].[5] Each task has between $N = 70,000$ and $500,000$ training data points. For 3droad we use a Gaussian observation model. The Precipitation dataset has noisier observations; therefore we apply a Student-T observation model. Finally, we reduce the CovType dataset to a binary classification problem and apply a Bernoulli observation model.[6] We train models with $10^3 \leq M \leq 10^4$. See Appx. F for details.

The two methods achieve very similar test-set negative log likelihood (Fig. 3). We note that there are small differences in the optimization dynamics, which is to be expected since $\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}$ can differ by an orthogonal transformation when computed with msMINRES-CIQ versus Cholesky. The key difference is the training time: with $M = 5,000$ inducing points, msMINRES-CIQ models are up to *5.6x faster* than Cholesky models (on a Titan RTX GPU). Moreover, msMINRES-CIQ models with $M = 8,000$-$10,000$ take roughly the same amount of time as $M = 5,000$ Cholesky models. This speed is due to the rapid convergence of msMINRES—on average $J = 100$ kernel-vector multiplies suffices to achieve 3 decimal places of error (see Appx. A). Note we do not train $M > 5,000$ Cholesky models as doing so would require 14GB of GPU memory and 2-10 days for training.

Effects of increased inducing points. We find that accuracy improves with increased M on all datasets. Scaling from $M = 5,000$ to $M = 10,000$ reduces test-set NLL by 0.1 nats on the 3droad and Precipitation datasets. We find similar reductions in predictive error (see Appx. A for plots). By scaling more readily to large M, msMINRES-CIQ enables high-fidelity variational approximations that would be computationally prohibitive with Cholesky.

5.2 Posterior Sampling for Bayesian Optimization

The second application of msMINRES-CIQ we explore is Gaussian process posterior sampling in the context of Bayesian optimization (BO) [e.g. 72]. Many acquisition functions require drawing samples from posteriors [e.g. 23, 42, 80]. One canonical example is **Thompson Sampling** (TS) [43, 47, 74]. TS trades off exploitation of existing minima for exploration of new potential minima. TS chooses the next query point $\widetilde{\mathbf{x}}$ as the minimizer of a sample drawn from the posterior. Let $\mathbf{X}^* = [\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*]$ be

---

[5]Details on these datasets (including how to acquire them) are in Appx. F.

[6]The task is predicting whether the primary tree cover at a given location is pine trees or other types of trees.
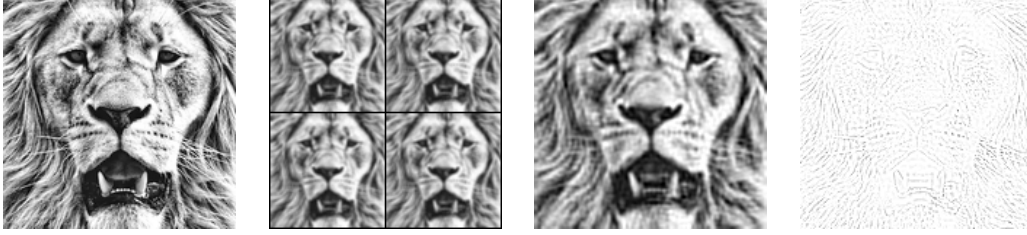
7

Figure 5: Using msMINRES-CIQ for solving problems in spatial statistics, such as image reconstruction. This requires sampling from a precision matrix of dimension $D = 25{,}600$. (**Left**) High-resolution image of dimension D. (**Middle Left**) Low-resolution images. (**Middle Right**) Reconstructed image. (**Right**) Delta between original image and reconstruction (darker colors correspond to larger deltas).

a *candidate set* of possible query points. To choose the next query point $\widetilde{\mathbf{x}}$, TS computes

$$\widetilde{\mathbf{x}} = \arg\min \left(\boldsymbol{\mu}^*(\mathbf{X}^*) + \mathbf{COV}^*(\mathbf{X}^*)^{\frac{1}{2}} \boldsymbol{\epsilon} \right), \quad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right). \tag{5}$$

where $\boldsymbol{\mu}^*(\mathbf{X}^*)$ and $\mathbf{COV}^*(\mathbf{X}^*)$ are the posterior mean and covariance of the Gaussian process at the candidate set. The candidate set is often chosen using a space-filling design, e.g. a Sobol sequence. The search space grows exponentially with the dimension; therefore, we need large values of T to more densely cover the search space for better optimization performance. Using Cholesky to compute Eq. (5) incurs a $\mathcal{O}(T^3)$ computational cost and $\mathcal{O}(T^2)$ memory, which severely limits the size of T. In comparison, msMINRES-CIQ only requires $\mathcal{O}(T^2)$ computation and $\mathcal{O}(T)$ memory.

We perform BO using TS on the classic test function (**Hartmann**, $D = 6$) and a reinforcement controller tuning problem (**Lunar Lander**, $D = 12$) from the OpenAI gym.[7] We provide more details in the supplementary material. For each problem we use exact Gaussian processes as the surrogate model and TS as the acquisition function. Our goal is to determine whether CIQ-based sampling is beneficial by enabling scaling to larger candidate set sizes.

Baselines. We measure the performance of TS as a function of the candidate set size T and consider $T \in \{1{,}000, 5{,}000, 20{,}000, 50{,}000\}$. We run Cholesky (**Cholesky**-T) for $T \in \{1{,}000, 5{,}000\}$ and msMINRES-CIQ (**CIQ**-T) for $T \geq 5{,}000$. Note that it would be very challenging and impractical to use Cholesky with $T \geq 10{,}000$, due to its quadratic memory and cubic time complexity. For example, running Cholesky for $T = 50{,}000$ would require ≥ 100 GB of GPU memory, and performing a single decomposition would take (at best) ≈ 30 seconds. In addition to Cholesky and CIQ with exact Gaussian processes as the surrogate model, we also compare to random Fourier features (RFF) [63] with $1{,}000$ random features.

Optimization performance. We plot the mean regret with standard error based on 30 replications in Fig. 4. By increasing $T = 1{,}000$ to $T = 50{,}000$, the final regret achieved by CIQ is significantly lower on both problems. We re-iterate that $T = 50{,}000$ is largely impractical with Cholesky. Large candidate sets have previously only been possible with approximate sampling methods like RFF. We note, however, that RFF with $T = 50{,}000$ is outperformed by CIQ-50k on both problems.

5.3 Gibbs Samplers and Image Reconstruction

High-dimensional Gaussian distributions are ubiquitous in Bayesian statistics, especially in the context of spatially structured data. Application areas are numerous, including disease mapping, archaeology, and image analysis [8, 49, 78]. Many of the models that arise in these applications are amenable to Gibbs sampling, a MCMC method for generating (approximate) samples from Bayesian posteriors. As such, sampling from high-dimensional Gaussian distributions is often the primary computational bottleneck for these methods.

To illustrate the utility of msMINRES-CIQ for constructing efficient Gibbs samplers for high-dimensional Gaussian latent variables, we consider an image reconstruction task [6]. We emphasize, however, the wide-ranging applicability of these methods, including for non-spatially structured data

---
[7]https://gym.openai.com/envs/LunarLander-v2

(e.g. for sparse linear regression [33]). We formulate an image analysis model as follows: we observe R low-resolution images $\{\mathbf{y}_r\}_{r=1}^R$, with each image of size $M \times M$. The goal is to reconstruct the unknown high-resolution image $\mathbf{x}$ of size $N \times N$ with $N > M$. The joint density is given by

$$p(\mathbf{x}, \mathbf{y}_{1:R}, \gamma_{\mathrm{obs}}, \gamma_{\mathrm{prior}}) = \mathcal{N}(\mathbf{y}_{1:R}|\mathbf{Ax}, \gamma_{\mathrm{obs}}^{-1}\mathbf{1})\mathcal{N}(\mathbf{x}|\mathbf{0}, \gamma_{\mathrm{prior}}^{-1}\mathbf{L})p(\gamma_{\mathrm{obs}})p(\gamma_{\mathrm{prior}}) \qquad (6)$$

where $\mathbf{A}$ is a $M^2 R \times N^2$ matrix that encodes how the high-resolution image is blurred and down-sampled to yield R low-resolution images and $\mathbf{L}$ is a $N^2 \times N^2$ discrete Laplace operator that encodes our prior smoothness assumptions about the image $\mathbf{x}$. Additionally, γ_{obs} and γ_{prior} are scalar hyperparameters that control the scale of the observation noise and strength of the image prior, respectively. For more details please refer to Appendix F. The computational bottleneck in the resulting Gibbs sampler is sampling from the conditional Gaussian distribution given by

$$p(\mathbf{x}|\mathbf{y}_{1:R}, \gamma_{\mathrm{obs}}, \gamma_{\mathrm{prior}}) = \mathcal{N}(\mathbf{x}|\mathbf{m}, \mathbf{\Lambda}^{-1}) \qquad \mathbf{m} = \gamma_{\mathrm{obs}}\mathbf{\Lambda}^{-1}\mathbf{A}^T\mathbf{y}_{1:R} \qquad \mathbf{\Lambda} = \gamma_{\mathrm{obs}}\mathbf{A}^T\mathbf{A} + \gamma_{\mathrm{prior}}\mathbf{L}$$

For a concrete demonstration we perform image reconstruction on the image depicted in Fig. 5. Here $N = 160$, $M = 80$, and $R = 4$, so that the precision matrix $\mathbf{\Lambda}$ is of size 25600×25600. Despite the extreme size, our implementation achieves ≈ 0.61 samples per second (using a TitanRTX GPU). We estimate that a Cholesky version of this method would achieve only ≈ 0.05 samples per second.

6   Discussion

We have introduced msMINRES-CIQ—a MVM-based method for computing $\mathbf{K}^{1/2}\mathbf{b}$ and $\mathbf{K}^{-1/2}\mathbf{b}$. In sampling and whitening applications, msMINRES-CIQ can be used as a $\mathcal{O}(N^2)$ drop-in replacement for the $\mathcal{O}(N^3)$ Cholesky decomposition. Its scalability and GPU utilization enable us to use more inducing points with SVGP models and larger candidate sets in Bayesian optimization. In all applications, such increased fidelity results in better performance.

Stability of msMINRES-CIQ. Krylov methods on symmetric matrices can be prone to numerical instabilities due to round-off errors [e.g. 60]. Our method has two key advantages that improve stability. First, we only use Krylov methods to solve linear systems rather than eigenvalue problems. Common numerical pitfalls that hinder Krylov eigen-solvers (e.g. loss of orthogonality between Lanczos vectors) have been shown to have little empirical effect on linear system solvers like MINRES and CG [e.g. 22, 75]. Second, each solve from msMINRES is inherently a shifted system $\mathbf{K} + t_q\mathbf{I}$. In practice these shifts dramatically improve the conditioning of $\mathbf{K}$, and allow us to work directly with the matrix $\mathbf{K}$ without having to add diagonal jitter for stability.

Comparison to other fast sampling methods. Historically, GP samples have been drawn using the Cholesky factor or finite-basis approximations like RFFs. Recently, a growing line of work investigates using inducing point methods for scalable sampling [61, 83]. We believe that CIQ-sampling can be used in conjunction with these inducing point approaches. For example, Wilson et al. [83] use RFFs to sample from the prior and an inducing point approximation of the conditional to convert prior samples into posterior samples. CIQ can augment this approach, allowing for more inducing points and/or replacing RFFs for prior sampling.

Advantages and disadvantages. One advantage of the Cholesky decomposition is its reusability. As discussed in Sec. 4, the cubic cost of computing $\mathbf{LL}^\top$ is amortized when drawing $\mathcal{O}(M)$ samples or whitening $\mathcal{O}(M)$ vectors. Conversely, applying msMINRES-CIQ to $\mathcal{O}(M)$ vectors would incur a $\mathcal{O}(M^3)$ cost, eroding its computational benefits. Thus, our method is primarily advantageous in scenarios with a small number of right hand sides or where $\mathbf{K}$ is too large to apply Cholesky. We also emphasize that msMINRES-CIQ—like all Krylov methods—can take advantage of fast MVMs afforded by structured covariances. Though this paper focuses on applying this algorithm to dense matrices, we suggest that future work explore applications involving sparse or structured matrices.

Broader Impact

This paper introduces an algorithm to improve the efficiency and scalability of a common-place computation. The results section highlights three common use cases of this algorithm: variational Gaussian processes, Bayesian optimization, and Gibbs sampling. While there are other potential use-cases of this method, we will focus on the broader impacts with respect to these three applications.

Variational Gaussian processes and Gibbs sampling are common methods. Other researchers have focused on domains like medicine [28, 68], geo-statistics [19, 73], and time-series modelling [64, 81] to motivate the need for increased scalability and efficiency. We believe that our proposed algorithm will make Gaussian process models and Gibbs sampling techniques increasingly applicable in these settings. Researchers/practitioners in these fields might have previously been unable to use Gaussian processes/Gibbs sampling due to scalability issues. While we believe increasing the scalability and usability of these probabilistic techniques is a worthwhile goal, we note that they require additional care when using. If a system is to rely on probabilistic methods for calibrated uncertainty estimates, it will no longer be sufficient to iterate on accuracy as a target method. We also note that performing meaningful probabilistic inferences requires some level of domain expertise regarding modeling priors and potential biases of sampling/variational approximations.

Bayesian optimization is a tool commonly used for hyperparameter optimization [72], A/B testing [5], and other black-box optimization problems. One of the most popular and best performing acquisition functions is Thompson sampling, which requires sampling the unknown function at a candidate set. The primary benefit of the proposed method is better optimization, which could lead to better machine learning models (via better hyperparameter searches) and faster experimental testing (via A/B testing). We would argue that improving the efficiency of such algorithms poses minimal risk beyond more general concerns about potential misapplications of the underlying technology to the optimization of nefarious objectives, intentionally or otherwise. However, we will make note here of some general risks associated with black-box optimization: a potential over-reliance on fully automated methods and computationally expensive searches for what might be marginal improvements.

We have release an open-sourced implementation of this algorithm to facilitate the adoption of this method.[8] Since our method relies on quadrature approximations and iterative refinement, one mode of failure is when such iterations fail to converge to a good estimate (for example, due to bad conditioning). However, there are several easy-to-perform convergence checks (e.g. the msMINRES residual), and such convergence checks are part of our implementation to catch such failure cases.

Acknowledgments and Disclosure of Funding

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1948.

[2] A. Asuncion and D. Newman. UCI machine learning repository. https://archive.ics.uci.edu/ml/, 2007. Last accessed: 2018-05-18.

[3] E. Aune, J. Eidsvik, and Y. Pokern. Iterative numerical methods for sampling from high dimensional Gaussian distributions. *Statistics and Computing*, 23(4):501–521, 2013.

[4] E. Aune, D. P. Simpson, and J. Eidsvik. Parameter estimation in high dimensional Gaussian distributions. *Statistics and Computing*, 24(2):247–263, 2014.

[5] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. Modular Bayesian optimization with BoTorch: An efficient differentiable monte-carlo approach. In *NeurIPS*, 2020.

[6] J. M. Bardsley. MCMC-based image reconstruction with uncertainty quantification. *SIAM Journal on Scientific Computing*, 34(3):A1316–A1332, 2012.

---

[8]See bit.ly/ciq_svgp and bit.ly/ciq_sampling.

[7] M. Bauer, M. van der Wilk, and C. E. Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *NeurIPS*, pages 1533–1541, 2016.

[8] J. Besag, J. York, and A. Mollié. Bayesian image restoration, with two applications in spatial statistics. *Annals of the institute of statistical mathematics*, 43(1):1–20, 1991.

[9] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

[10] B. C. Carlson and J. Todd. The degenerating behavior of elliptic functions. *SIAM Journal on Numerical Analysis*, 20(6):1120–1129, 1983. ISSN 00361429.

[11] B. Charlier, J. Feydy, J. Glaunès, F.-D. Collin, and G. Durif. Kernel operations on the GPU, with autodiff, without memory overflows. 2020.

[12] S.-C. Choi. *Iterative methods for singular linear equations and least-squares problems*. PhD thesis, 2006.

[13] E. Chow and Y. Saad. Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions. *Journal on Scientific Computing*, 36(2):A588–A608, 2014.

[14] N. Cundy, S. Krieg, G. Arnold, A. Frommer, T. Lippert, and K. Schilling. Numerical methods for the QCD overlap operator IV: Hybrid monte carlo. *Computer Physics Communications*, 180(1):26–54, 2009.

[15] J. P. Cunningham, K. V. Shenoy, and M. Sahani. Fast Gaussian process methods for point process intensity estimation. In *ICML*, 2008.

[16] K. Cutajar, M. Osborne, J. Cunningham, and M. Filippone. Preconditioning kernel matrices. In *ICML*, 2016.

[17] B. N. Datta and Y. Saad. Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment. *Linear Algebra and its Applications*, 154-156:225 – 244, 1991.

[18] P. I. Davies and N. J. Higham. Computing $f(A)b$ for matrix functions f. In *QCD and numerical analysis III*, pages 15–24. Springer, 2005.

[19] P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(3):299–350, 1998.

[20] K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. Scalable log determinants for Gaussian process kernel learning. In *NeurIPS*, 2017.

[21] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. In *NeurIPS*, 2019.

[22] D. C.-L. Fong and M. Saunders. CG versus MINRES: An empirical comparison. *Sultan Qaboos University Journal for Science*, 17(1):44–62, 2012.

[23] P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *Journal on Computing*, 21(4):599–613, 2009.

[24] R. Freund. On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices. *Numerische Mathematik*, 57(1):285–312, 1990.

[25] A. Frommer and U. Glässner. Restarted GMRES for shifted linear systems. *SIAM Journal on Scientific Computing*, 19(1):15–26, 1998.

[26] A. Frommer, S. Güttel, and M. Schweitzer. Convergence of restarted krylov subspace methods for stieltjes functions of matrices. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1602–1624, 2014.

[27] A. Frommer, S. Güttel, and M. Schweitzer. Efficient and stable arnoldi restarts for matrix functions based on quadrature. *SIAM Journal on Matrix Analysis and Applications*, 35(2):661–683, 2014.

[28] J. Futoma, S. Hariharan, and K. Heller. Learning to detect sepsis with a multitask Gaussian process RNN classifier. In *ICML*, 2017.

[29] J. R. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *NeurIPS*, pages 7576–7586, 2018.

[30] J. R. Gardner, G. Pleiss, R. Wu, K. Q. Weinberger, and A. G. Wilson. Product kernel interpolation for scalable Gaussian processes. In *AISTATS*, 2018.

[31] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

[32] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, 1997.

[33] J. Griffin, P. Brown, et al. Hierarchical shrinkage priors for regression models. *Bayesian Analysis*, 12(1): 135–159, 2017.

[34] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: evaluating models of vehicular environmental impact. In *International Conference on Advances in Geographic Information Systems*, 2012.

[35] N. Hale, N. J. Higham, and L. N. Trefethen. Computing A^α, $\log(A)$, and related matrix functions by contour integrals. *SIAM Journal on Numerical Analysis*, 46(5):2505–2523, 2008.

[36] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. 2009.

[37] H. Harbrecht, M. Peters, and R. Schneider. On the low-rank approximation by the pivoted Cholesky decomposition. *Applied Numerical Mathematics*, 4(62):428–440, 2012.

[38] J. Hensman, M. Rattray, and N. D. Lawrence. Fast variational inference in the conjugate exponential family. In *NeurIPS*, 2012.

[39] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *UAI*, 2013.

[40] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. 2015.

[41] J. Hensman, N. Durrande, and A. Solin. Variational Fourier features for Gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

[42] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NeurIPS*, 2014.

[43] J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *ICML*, 2017.

[44] N. J. Higham. *Functions of matrices: theory and computation*, volume 104. SIAM, 2008.

[45] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[46] B. Jegerlehner. Krylov space solvers for shifted linear systems. *arXiv preprint hep-lat/9612014*, 1996.

[47] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *AISTATS*, 2018.

[48] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[49] L. Knorr-Held and H. Rue. On block updating in Markov random field models for disease mapping. *Scandinavian Journal of Statistics*, 29(4):597–614, 2002.

[50] M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6(Oct):1679–1704, 2005.

[51] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators1. *Journal of Research of the National Bureau of Standards*, 45(4), 1950.

[52] B. Lyon. The strength of el niño and the spatial extent of tropical drought. *Geophysical Research Letters*, 31(21), 2004.

[53] B. Lyon and A. G. Barnston. Enso and the spatial extent of interannual precipitation extremes in tropical land areas. *Journal of Climate*, 18(23):5095–5109, 2005.

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

[55] K. Meerbergen. The solution of parametrized symmetric linear systems. *SIAM journal on matrix analysis and applications*, 24(4):1038–1059, 2003.

[56] K. R. Meyer. Jacobi elliptic functions from a dynamical systems point of view. *The American Mathematical Monthly*, 108(8):729–737, 2001.

[57] I. Murray. Gaussian processes and fast matrix-vector multiplies. In *ICML Workshop on Numerical Mathematics in Machine Learning*, 2009.

[58] M. Mutny and A. Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. In *NeurIPS*, 2018.

[59] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *Journal on numerical analysis*, 12(4):617–629, 1975.

[60] B. N. Parlett and D. S. Scott. The lanczos algorithm with selective orthogonalization. *Mathematics of computation*, 33(145):217–238, 1979.

[61] G. Pleiss, J. R. Gardner, K. Weinberger, and A. G. Wilson. Constant-time predictive distributions for Gaussian processes. In *ICML*, 2018.

[62] S. Qiu, M. K. Vamanamurthy, and M. Vuorinen. Some inequalities for the growth of elliptic integrals. *SIAM journal on mathematical analysis*, 29(5):1224–1237, 1998.

[63] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2008.

[64] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions. Series A: Mathematical, Physical and Engineering Sciences*, 371 (1984), 2013.

[65] Y. Saad. *Iterative methods for sparse linear systems*, volume 82. SIAM, 2003.

[66] H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *AISTATS*, 2018.

[67] M. K. Schneider and A. S. Willsky. Krylov subspace estimation. *SIAM Journal on Scientific Computing*, 22(5):1840–1864, 2001.

[68] P. Schulam and S. Saria. A framework for individualizing predictions of disease trajectories by exploiting multi-resolution structure. In *NeurIPS*, 2015.

[69] J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[70] D. P. Simpson, I. W. Turner, and A. N. Pettitt. Fast sampling from a gaussian markov random field using krylov subspace approaches. 2008.

[71] D. P. Simpson, I. W. Turner, C. M. Strickland, and A. N. Pettitt. Scalable iterative methods for sampling from massive gaussian random vectors. *arXiv preprint arXiv:1312.1476*, 2013.

[72] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

[73] M. L. Stein. *Interpolation of spatial data: some theory for Kriging*. Springer Science & Business Media, 2012.

[74] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[75] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

[76] S. Ubaru, J. Chen, and Y. Saad. Fast estimation of $\mathrm{Tr}(f(A))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

[77] H. A. Van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press, 2003.

[78] L. A. Waller, B. P. Carlin, H. Xia, and A. E. Gelfand. Hierarchical spatio-temporal mapping of disease rates. *Journal of the American Statistical association*, 92(438):607–617, 1997.

[79] K. A. Wang, G. Pleiss, J. R. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact Gaussian processes on a million data points. In *NeurIPS*, 2019.

[80] Z. Wang and S. Jegelka. Max-value entropy search for efficient Bayesian optimization. In *ICML*, 2017.

[81] A. G. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, 2013.

[82] A. G. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *ICML*, 2015.

[83] J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *ICML*, 2020.

[84] Z.-H. Yang and J.-F. Tian. Convexity and monotonicity for elliptic integrals of the first kind and applications. *Applicable Analysis and Discrete Mathematics*, 13(1):240–260, 2019.

Supplementary Information for: Fast Matrix Square Roots with Applications to Gaussian Processes and Bayesian Optimization

Geoff Pleiss
Columbia University
gmp2162@columbia.edu

Martin Jankowiak
The Broad Institute
mjankowi@broadinstitute.org

David Eriksson*
Facebook
deriksson@fb.com

Anil Damle
Cornell University
damle@cornell.edu

Jacob R. Gardner
University of Pennsylvania
jacobrg@seas.upenn.edu

Algorithm 1: Computing $\mathbf{K}^{-\frac{1}{2}}\mathbf{b}$ with MVM-based Contour Integral Quadrature (CIQ)

Input : `mvm_K` $(\cdot)$ – function for matrix-vector multiplication (MVM) with matrix $\mathbf{K}$
        $\mathbf{b}$ – right hand side, J – number of `msMINRES` iterations, Q – number of quad. points
Output: $\mathbf{a} \approx \mathbf{K}^{-\frac{1}{2}}\mathbf{b}$

$[w_1, \ldots, w_Q], [t_1, \ldots, t_Q] \leftarrow$ `compute_quad(` `mvm_K` `(·),` Q `)` `// Weights` (w_i) `and`
   `shifts` (t_i) `for quadrature – details in Appx. B.`
$(t_1\mathbf{I} + \mathbf{K})^{-1}\mathbf{b}, \ldots (t_Q\mathbf{I} + \mathbf{K})^{-1}\mathbf{b} \leftarrow$ `msMINRES(` `mvm_K` `(·),` $\mathbf{b}$`,` J`,` t_1`, ...,` t_Q`)`
   `// msMINRES computes all solves simultaneously – details in`
   `Appx. C.`

return $\sum_{q=1}^{Q} w_q \left(t_q\mathbf{I} + \mathbf{K}\right)^{-1}\mathbf{b}$ `// CIQ estimate of`
$\frac{1}{2\pi i} \int \tau^{-1/2}(\tau\mathbf{I} - \mathbf{K})^{-1}\mathbf{b}\,\mathrm{d}\tau = \mathbf{K}^{-1/2}\mathbf{b}$
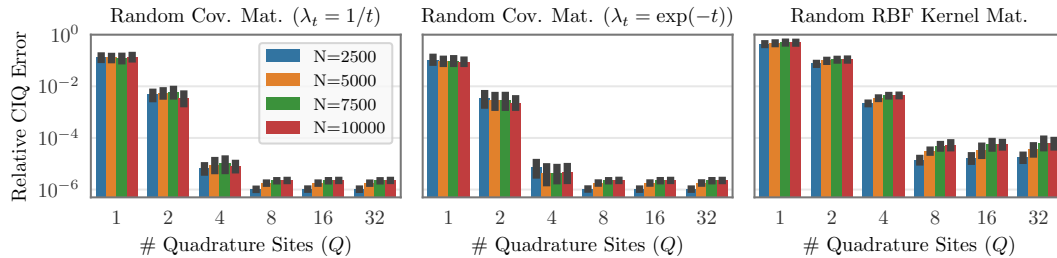

A   Additional Results

Figure S1: CIQ relative error at computing $\mathbf{K}^{1/2}\mathbf{b}$ as a function of number of quadrature points Q. In all cases $Q = 8$ achieves $< 10^{-4}$ error.

Fig. S1 and Fig. S2 are continuations of Fig. 1. They plots CIQ convergence and randomized SVD convergence as a function of Q and R for covariance matrices whose eigenvalues decay as $\lambda_t = \frac{1}{\sqrt{t}}$, $\lambda_t = \frac{1}{t}$, $\lambda_t = \frac{1}{t^2}$, and $\lambda_t = \exp(-t)$ in addition to the kernel matrix results already presented. The
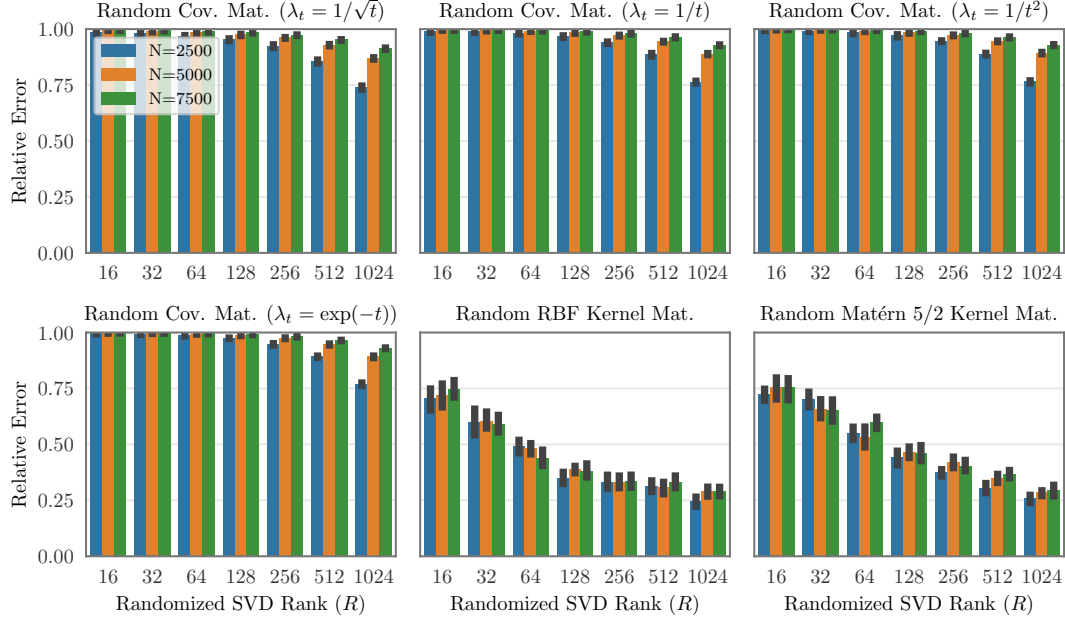

Figure S2: Randomized SVD relative error at computing $\mathbf{K}^{1/2}\mathbf{b}$ as a function of approximation rank R. In all cases, randomized SVD is unable to achieve a relative error better than about 0.25.

results for CIQ demonstrate that it is relatively invariant to the eigenvalue decay speed, and does not require approximately low rank structure. Randomized SVD on the other hand incurs an order of magnitude more error; a rank of $1{,}024$ is unable to reduce the relative error to a single decimal point.
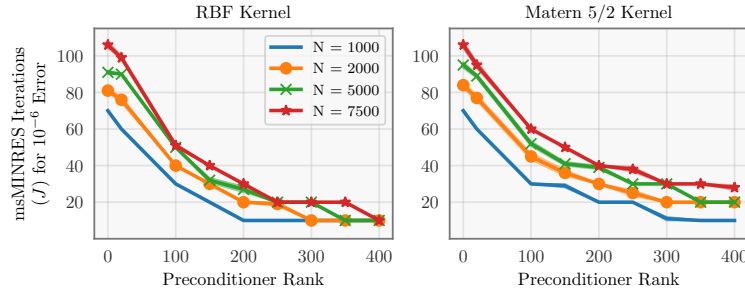

Figure S3: Effect of preconditioning on CIQ convergence (random RBF and Matérn-5/2 kernels with a pivoted Cholesky preconditioner [29]).

Fig. S3 further demonstrates the effect of preconditioning on msMINRES-CIQ. We construct random $N \times N$ RBF/Matérn kernels, applying msMINRES-CIQ to a set of N orthonormal vectors ($[\mathbf{K}^{1/2}\mathbf{b}_1, \ldots, \mathbf{K}^{1/2}\mathbf{b}_N]$), and compute the empirical covariance. We plot the number of msMINRES iterations needed to achieve a relative error of 10^{-4}. The pivoted Cholesky preconditioner of Gardner et al. [29]—which forms a low-rank approximation of $\mathbf{K}$—accelerates convergence of msMINRES. Without preconditioning (i.e. rank=0), $J = 100$ iterations are required for $N = 7{,}500$ matrices. With rank-100/rank-400 preconditioners, iterations are cut by a factor of two/four.

To further compare msMINRES-CIQ to randomized methods, Fig. S4 plots the empirical covariance matrix of $1{,}000$ Gaussian samples drawn from a Gaussian process prior $\mathcal{N}(\mathbf{0}, \mathbf{K})$. We construct the RBF covariance matrices $\mathbf{K}$ using subsets of the Protein and Kin40k datasets[9] [2]. We note that all methods incur some sampling error, regardless of the subset size (N). msMINRES-CIQ and Cholesky-based sampling tend to have very similar empirical covariance error. On the other hand, the

---

[9]Both datasets are originally from the UCI repository and can be downloaded from https://github.com/gpleiss/ciq_experiments/tree/main/svgp/data.
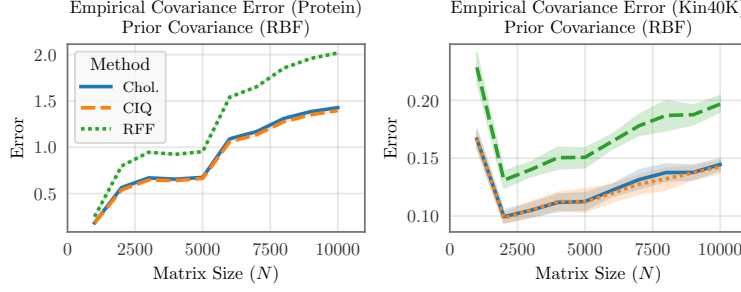
Figure S4: Empirical covariance error (relative norm) for various sampling methods (Cholesky, msMINRES-CIQ, and 1,000 Random Fourier Features [63]). Empirical covariances are measured from 1,000 samples. RBF matrices are constructed from data in the Protein and Kin40k datasets [2].

Random Fourier Features method [63] (with 1,000 random features) incurs errors up to $2\times$ as large. This additional error is due to the randomness in the RFF approximation.
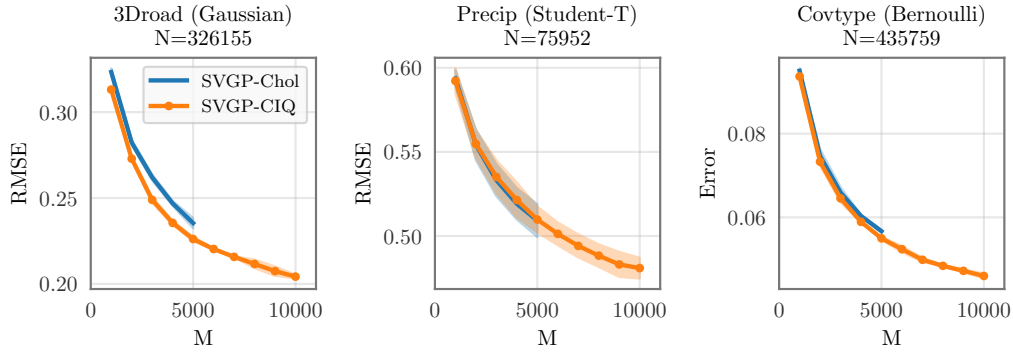

Figure S5: Error comparison of Cholesky-whitened vs CIQ-whitened SVGP models. **Left:** 3DRoad dataset RMSE ($N = 326155, D = 2$, Gaussian likelihood). **Middle:** Precipitation dataset RMSE ($N = 75952, D = 3$, Student-T likelihood). **Right:** CoverType dataset $0/1$ error ($N = 435759, D = 54$, Bernoulli likelihood). Error improves with more inducing points (M), and Cholesky and CIQ models have similar performance. However CIQ scales to larger values of M.

In Fig. S5 we plot the predictive error of CIQ-SVGP and Chol-SVGP models as a function of M. For the two regression datasets (3droad and Precipitation) error is measured by test set root mean squared error (RMSE). On the Covtype classification dataset error is measured by the test set $0/1$ loss. As with the NLL results in Fig. 3 we find that the CIQ-SVGP and Chol-SVGP perform similarly, despite the fact that CIQ-SVGP can be up to $5.6\times$ faster. Moreover, we see that error continuously decreases with more inducing points up to $M = 10,000$.

In Fig. S6 we plot the learned hyperparameters of the Precipitation SVGP models: 1) o^2 (the kernel outputscale)—which roughly corresponds to variance explained as "signal" in the data; 2) σ_{obs}^2—which roughly corresponds to variance explained away as observational noise; and 3) ν (degrees of freedom)—which controls the tails of the noise model (lower ν corresponds to heavier tails). As M increases, we find that the observational noise parameter decreases by a factor of 4—down from 0.19 to 0.05—while the ν parameter also decreases. Models with larger M values can more closely approximate the true posterior [39]; therefore, we expect that the parameters from the larger-M likelihoods more closely correspond to the true dataset noise. This confirms findings from Bauer et al. [7], who argue that variational approximations with small M can tend to overestimate the amount of noise in datasets.

Fig. S7 is a histogram displaying the msMINRES iterations needed to achieve a relative residual of 10^{-3} when training a $M = 5,000$ SVGP model on the 3droad dataset (subsampled to 30,000 data points). Most msMINRES calls converge in fewer than 100 iterations; almost no calls require more than 200 iterations. We hypothesize that this fast convergence is due to solving shifted systems
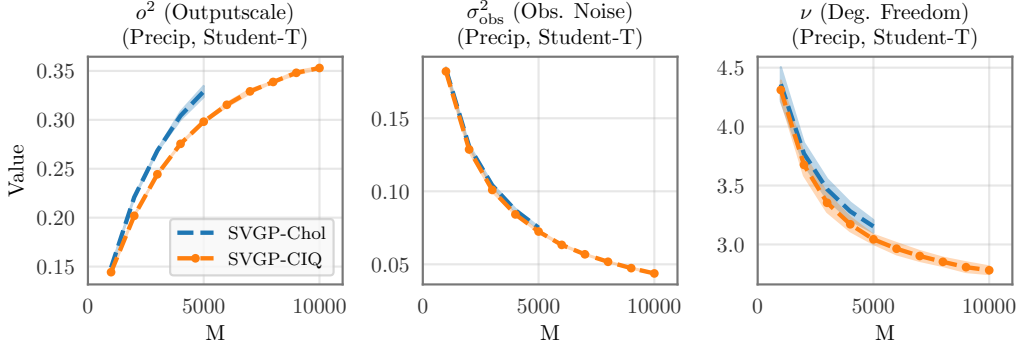
Figure S6: Hyperparameters versus number of inducing points (M) for Chol-SVGP and CIQ-SVGP (Precipitation dataset, Student-T likelihood). As M increases, the kernel outputscale (left) also increases. At the same time, the estimated observational noise (middle) decreases as does the estimated degrees of freedom (right), reflecting a heavier-tailed noise distribution. This suggests that, with larger M, SVGP models can find more signal in the data.
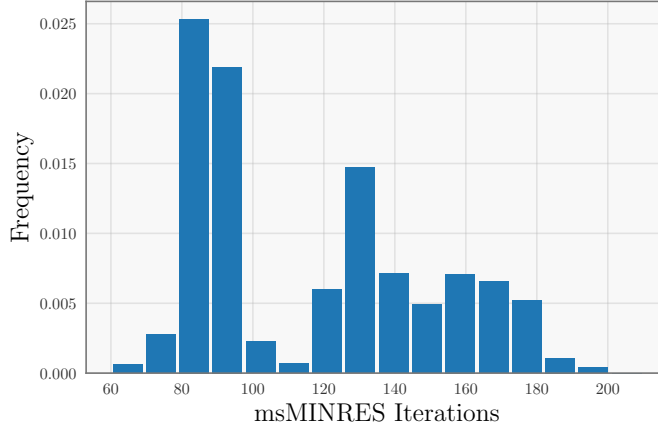
Figure S7: Number of msMINRES iterations needed to achieve a relative residual of 10^{-3}. Histogram captures training a $M = 5,000$ SVGP model on the 3droad dataset (subsampled to 30,000 data points).

$(\mathbf{K} + t_q \mathbf{I})$. The minimum eigenvalues of the shifted matrix are lower-bounded by t_q, and therefore shifted systems have a better condition number than the unshifted matrix $\mathbf{K}$.

B Quadrature for Matrix Square Roots

Here we briefly describe the quadrature formula derived by Hale et al. [35] for use with Cauchy's integral formula and refer the reader to the original publication for more details.

Assume that $\mathbf{K}$ is a positive definite matrix, and thus has real positive eigenvalues. Our goal is to approximate Cauchy's integral formula with a quadrature estimate:

$$f(\mathbf{K}) = \frac{1}{2\pi i} \oint_{\Gamma} f(\tau) \left(\tau \mathbf{I} - \mathbf{K}\right)^{-1} \mathrm{d}\tau \tag{S1}$$

$$\approx \frac{1}{2\pi i} \sum_{q=1}^{Q} \widetilde{w}_q f(\tau_q) \left(\tau_q \mathbf{I} - \mathbf{K}\right)^{-1} , \tag{S2}$$

18

where $f(\cdot)$ is analytic on and within Γ, and $\widetilde{w}_q$ and τ_q are quadrature weights and nodes respectively. Note that Eq. (S1) holds true for any closed contour Γ in the complex plane that winds once (counterclockwise) around the spectrum of $\mathbf{K}$.

A naïve approach with uniformly-spaced quadrature. For now, assume that $\lambda_{\min}$ and $\lambda_{\max}$—the minimum and maximum eigenvalues of $\mathbf{K}$—are known. (We will later address how they can be efficiently estimated.) A naïve first approach to Eq. (S2) is to uniformly place the quadrature locations in a circle that surrounds the eigenvalues and avoids crossing the negative real axis, where we anticipate f may be singular:

$$\tau_q = \frac{\lambda_{\max} + \lambda_{\min}}{2} + \frac{\lambda_{\max}}{2} e^{2i\pi(q/Q)}, \quad \widetilde{w}_q = \frac{1}{Q}, \quad q = 0, 1, \ldots, Q-1.$$

This corresponds to a standard trapezoid quadrature rule. However, Hale et al. [35] demonstrate that the convergence of this quadrature rule depends linearly on the condition number $\kappa(\mathbf{K}) = \lambda_{\max}/\lambda_{\min}$. In particular, this is because the integrand is only analytic in a narrow region around the chosen contour. As many kernel matrices tend to be approximately low-rank and therefore ill-conditioned, this simple quadrature rule requires large Q to achieve the desired numerical accuracy.

Improving convergence with conformal mappings. Rather than uniformly spacing the quadrature points, it makes more sense to place more quadrature points near $\lambda_{\min}$ and fewer near $\lambda_{\max}$. This can be accomplished by using the above trapezoid quadrature rule in a transformed parameter space that is "stretched" near $\lambda_{\min}$ and contracted near $\lambda_{\max}$. Mathematically, this is accomplished by applying a conformal mapping that moves the singularities to the upper and lower boundaries of a periodic rectangle. We may then apply the trapezoid rule along a contour traversing the middle of the rectangle—maximizing the region in which the function we are integrating is analytic around the contour.

B.1 A Specific Quadrature Formula for $f(\mathbf{K}) = \mathbf{K}^{-1/2}$

Hale et al. [35] suggest performing a change of variables that projects Eq. (S1) onto an annulus. Uniformly spaced quadrature points inside the annulus will cluster near $\lambda_{\min}$ when projected back into the complex plane. This change of variables has a simple analytic formula involving Jacobi elliptic functions (see [35, Sec. 2] for details.) In the special case of $f(\mathbf{K}) = \mathbf{K}^{-1/2}$, we can utilize an additional change of variables for an even more efficient quadrature formulation [35, Sec. 4]. Setting $\sigma = \tau^{1/2}$, we have

$$\mathbf{K}^{-\frac{1}{2}} = \frac{1}{\pi i} \oint_{\Gamma_s} \left(\sigma^2 \mathbf{I} - \mathbf{K}\right)^{-1} \mathrm{d}\sigma.$$

$$\approx \frac{1}{\pi i} \sum_{q=1}^{Q} \widetilde{w}_q \left(\sigma_q^2 \mathbf{I} - \mathbf{K}\right)^{-1}, \tag{S3}$$

where Γ_σ is a contour that surrounds the spectrum of $\mathbf{K}^{1/2}$. Since the integrand is symmetric with respect to the real axis, we only need to consider the imaginary portion of Γ_σ. Consequently, all the τ_q quadrature locations (back in the original space) will be real-valued and negative. Combining this square-root change-of-variables with the annulus change-of-variables results in the following quadrature weights/locations:

$$\sigma_q^2 = \lambda_{\min}\left(\mathrm{sn}(iu_q \mathcal{K}'(k) \mid k)\right)^2,$$

$$\widetilde{w}_q = -\frac{2\sqrt{\lambda_{\min}}}{\pi Q} \left[\mathcal{K}'(k) \quad \mathrm{cn}\left(iu_q \mathcal{K}'(k) \mid k\right) \quad \mathrm{dn}\left(iu_q \mathcal{K}'(k) \mid k\right)\right], \tag{S4}$$

where we adopt the following notation:

- $k = \sqrt{\lambda_{\min}/\lambda_{\max}} = 1/\sqrt{\kappa(\mathbf{K})}$;
- $\mathcal{K}'(k)$ is the complete elliptic integral of the first kind with respect to the complimentary elliptic modulus $k' = \sqrt{1-k^2}$;
- $u_q = \frac{1}{Q}(q - \frac{1}{2})$; and

- $\text{sn}(\cdot \mid k)$, $\text{cn}(\cdot \mid k)$, and $\text{dn}(\cdot \mid k)$ are the Jacobi elliptic functions with respect to elliptic modulus k.

The weights $\widetilde{w}_q$ and locations σ_q^2 from Eq. (S4) happen to be real-valued and negative. Setting $t_q = -\sigma_q^2$ and $w_q = -\widetilde{w}_q$ gives us:

$$\mathbf{K}^{-\frac{1}{2}} \approx \sum_{q=1}^{Q} w_q \left(t_q \mathbf{I} + \mathbf{K}\right)^{-1}, \quad w_q = -\widetilde{w}_q > 0, \quad t_q = -\sigma_q^2 > 0. \tag{S5}$$

An immediate consequence of this is that the shifted matrices $(t_q \mathbf{I} + \mathbf{K})$ are all positive definite.

Convergence of the quadrature approximation. Due to the double change-of-variables, the convergence of this quadrature rule in Eq. (S4) is extremely rapid—even for ill-conditioned matrices. Hale et al. prove the following error bound:

Lemma 1 (Hale et al. [35], Thm. 4.1). *Let $t_1, \ldots, t_Q > 0$ and $w_1, \ldots, w_Q > 0$ be the locations and weights of Hale et al.'s quadrature procedure. The error of Eq. (2) is bounded by:*

$$\left\| \mathbf{K} \sum_{q=1}^{Q} w_q \left(t_q \mathbf{I} + \mathbf{K}\right)^{-1} - \mathbf{K}^{\frac{1}{2}} \right\|_2 \leq \mathcal{O}\left(\exp\left(-\frac{2Q\pi^2}{\log \kappa(\mathbf{K}) + 3} \right) \right),$$

where $\kappa(\mathbf{K}) = \lambda_{max}/\lambda_{min}$ is the condition number of $\mathbf{K}$.

Remarkably, the error of Eq. (2) is *logarithmically* dependent on the conditioning of $\mathbf{K}$. Consequently, $Q \approx 8$ quadrature points is even sufficient for ill-conditioned matrices (e.g. $\kappa(\mathbf{K}) \approx 10^4$).

B.2 Estimating the Minimum and Maximum Eigenvalues

The equations for the quadrature weights/locations depend on the extreme eigenvalues λ_{max} and λ_{min} of $\mathbf{K}$. Using the Lanczos algorithm [51]—which is a Krylov subspace method—we can obtain accurate estimates of these extreme eigenvalues using relatively few matrix-vector multiplies with $\mathbf{K}$.

The Lanczos algorithm is a method for computing an orthonormal basis for Krylov subspaces of a symmetric matrix $\mathbf{K}$ and, simultaneously, projections of A onto that subspace. Given an initial vector $\mathbf{b}$, the algorithm iteratively factorizes $\mathbf{K}$ as:

$$\mathbf{K}\mathbf{Q}_J = \mathbf{Q}_J \mathbf{T}_J + \mathbf{r}_J \mathbf{e}_J^\top$$

where $\mathbf{e}_J$ is a unit vector, and

- $\mathbf{Q}_J \in \mathbb{R}^{N \times J}$ is an orthonormal basis of the Jth Krylov subspace $\mathcal{K}(\mathbf{K}, \mathbf{b})$,
- $\mathbf{T}_J \in \mathbb{R}^{J \times J}$ is a symmetric tridiagonal matrix, and
- $\mathbf{r}_J \in \mathbb{R}^J$ is a residual term.

At a high level, the Lanczos iterations form the Krylov subspaces while simultaneously performing a process akin to modified Gram Schmidt orthogonalization:

$$\text{span}\{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(J)}\} = \mathcal{K}(\mathbf{K}, \mathbf{b}) = \text{span}\{\mathbf{b}, \mathbf{Kb}, \mathbf{K}^2\mathbf{b}, \ldots, \mathbf{K}^{J-1}\mathbf{b}\}.$$

The orthogonal basis vectors are collected into $\mathbf{Q}$ and the orthogonalization coefficients are collected into $\mathbf{T}$. Due to the symmetry of $\mathbf{K}$ a three term recurrence exists for this process and each vector $\mathbf{q}^{(j)}$ only has to be orthogonalized against the two previous basis vectors $\mathbf{q}^{(j-1)}$, $\mathbf{q}^{(j-2)}$—resulting in a tridiagonal $\mathbf{T}$.

Estimating Extreme Eigenvalues from Lanczos. To estimate λ_{min} and λ_{max} from Lanczos, we perform an eigendecomposition of $\mathbf{T}_J$. If J is small (i.e. $J \approx 10$) then this eigendecomposition requires minimal computational resources. In fact, as $\mathbf{T}_J$ is tridiagonal invoking standard routines allows computation of all the eigenvalues in $\mathcal{O}(J^2)$ time. A well-known convergence result of the Lanczos algorithm is that the extreme eigenvalues of $\mathbf{T}_J$ tend to converge rapidly to λ_{min} and λ_{max} [e.g. 31, 65]. Since the Lanczos algorithm always produces underestimates of the largest eigenavlue and overestimates of the smallest it is reasonable to use slightly larger and smaller values in the construction of the quadrature scheme—as we see in Lemma 1, the necessary number of quadrature nodes is insensitive to small overestimates of the condition number.

Algorithm 2: Computing w_q and t_q for Contour Integral Quadrature

---

Input : `mvm_K` $(\cdot)$ – function for matrix-vector multiplication (MVM) with matrix $\mathbf{K}$
   Q – number of quad. points
Output : $w_1, \ldots, w_Q, t_1, \ldots, t_Q$

```
// Estimate extreme eigenvalues with Lanczos.
```
$\_, \mathbf{T} \leftarrow$ `lanczos(mvm_K`$(\cdot)$`)` `// Lanczos w/ rand.  init.  vector`
$\lambda_{\min}, \cdots, \lambda_{\max} \leftarrow$ `symeig(`$\mathbf{T}$`)`
```
// Compute elliptic integral of the first kind.
// We use the relation 𝒦'(k) = 𝒦(k'), where k' = √(1−k²) is the
   complementary elliptic modulus.
```
$k^2 \leftarrow \lambda_{\min}/\lambda_{\max}$ `// The squared elliptic modulus.`
$k'^2 \leftarrow \sqrt{1-k^2}$ `// The squared complementary elliptic modulus.`
$\mathtt{K}' \leftarrow$ `ellipke(`k'^2`)` `// K' = 𝒦'(k)`
```
// Compute each quadrature weight/location.
```
for $q \leftarrow 1$ **to** Q **do**
 $\quad u_q \leftarrow (q - 1/2)/Q$
 $\quad$ `// Compute Jacobi elliptic fn's via Jacobi's imaginary transform.`
 $\quad$ `// First we compute` $\overline{\mathtt{sn}}_q = \mathrm{sn}(u_q \mathcal{K}'(k)|k'),\ \overline{\mathtt{cn}}_q = \mathrm{cn}(u_q \mathcal{K}'(k)|k'),$
 $\qquad \overline{\mathtt{dn}}_q = \mathrm{dn}(u_q \mathcal{K}'(k)|k').$
 $\quad \overline{\mathtt{sn}}_q, \overline{\mathtt{cn}}_q, \overline{\mathtt{dn}}_q \leftarrow$ `ellipj(`$u_q \mathtt{K}', k'^2$`)`
 $\quad$ `// Use identities to convert` $\overline{\mathtt{sn}}_q,\ \overline{\mathtt{cn}}_q,\ \overline{\mathtt{dn}}_q$ `values into`
 $\quad$ `//` $\mathtt{sn}_q = \mathrm{sn}(iu_q\mathcal{K}'(k)|k),\ \mathtt{cn}_q = \mathrm{cn}(iu_q\mathcal{K}'(k)|k),\ \mathtt{dn}_q = \mathrm{dn}(iu_q\mathcal{K}'(k)|k).$
 $\quad \mathtt{sn}_q \leftarrow i\left[\overline{\mathtt{sn}}_q/\overline{\mathtt{cn}}_q\right]$
 $\quad \mathtt{dn}_q \leftarrow \left[\overline{\mathtt{dn}}_q/\overline{\mathtt{cn}}_q\right]$
 $\quad \mathtt{cn}_q \leftarrow \left[1/\overline{\mathtt{cn}}_q\right]$
 $\quad$ `// Quadrature weight` w_q `and location` t_q
 $\quad w_q \leftarrow (-2\lambda_{\min}^{1/2})/(\pi Q)\, \mathtt{K}'\, \mathtt{cn}_q\, \mathtt{dn}_q$
 $\quad t_q \leftarrow \lambda_{\min}\left(\mathtt{sn}_q\right)^2$
end
return $w_1, \ldots, w_Q, t_1, \ldots, t_Q$

B.3 The Complete Quadrature Algorithm

Alg. 2 obtains the quadrature weights w_q and locations t_q corresponding to Eqs. (S4) and (S5). Computing these weights requires ≈ 10 matrix-vector multiplies with $\mathbf{K}$—corresponding to the Lanczos iterations—for a total time complexity of $\mathcal{O}(N)$. All computations involving elliptic integrals can be readily computed using routines available in e.g. the SciPy library.

C   The msMINRES Algorithm

Before introducing the msMINRES algorithm, we will first introduce MINRES as proposed by Paige and Saunders [59]; MINRES can be derived from the Lanczos algorithm [51] and, therefore, is able to take advantage of the same three term vector recurrence when building the necessary Krylov subspaces. We will then describe how msMINRES can be derived as a straightforward extension. Notably, we present this section assuming our best initial guess for the linear system we seek to solve is zero. If this is not the case a single step of iterative refinement can be used and the resulting residual system is solved with zero as the initial guess.

C.1   Standard MINRES

The method of minimum residuals (MINRES) [59] is an alternative to linear conjugate gradients, with the advantage that it can be applied to indefinite and singular symmetric matrices $\mathbf{K}$. Paige and Saunders [59] formulate MINRES to solve the least-squares problem $\arg\min_{\mathbf{c}} \|\mathbf{K}\mathbf{c} - \mathbf{b}\|_2$. Each

iteration J produces a solution $\mathbf{c}_J$ which is optimal within the Jth Krylov subspace:

$$\mathbf{c}_J^{(\text{MINRES})} = \underset{\mathbf{c} \in \mathcal{K}_J(\mathbf{K}, \mathbf{b})}{\arg\min} \ \|\mathbf{K}\mathbf{c} - \mathbf{b}\|_2. \tag{S6}$$

Using the Lanczos matrices and some mathematical manipulation, Eq. (S6) can be re-formulated as an unconstrained optimization problem:

$$\mathbf{c}_J^{(\text{MINRES})} = \|\mathbf{b}\|_2 \mathbf{Q}_J \mathbf{z}_J$$

$$\mathbf{z}_J = \underset{\mathbf{y} \in \mathbb{R}^J}{\arg\min} \left\| \left(\widetilde{\mathbf{T}}_J\right) \mathbf{y} - \mathbf{e}_1 \right\|_2, \quad \widetilde{\mathbf{T}}_J = \begin{bmatrix} \mathbf{T}_J \\ \|\mathbf{r}_J\|_2 \mathbf{e}_J^\top \end{bmatrix}, \tag{S7}$$

where $\mathbf{e}_1, \mathbf{e}_J$ are unit vectors, and $\mathbf{Q}_J$, $\mathbf{T}_J$, and $\mathbf{r}_J$ are the outputs from the Lanczos algorithm. Since Eq. (S7) is a least-squares problem (guaranteed to be full column-rank unless $\mathbf{b}$ lives in the Jth Krylov subspace—at which point we would exactly solve the problem), we can write the analytic solution to it using the reduced QR factorization of $\widetilde{\mathbf{T}}_J = \mathcal{Q}_J \mathbf{R}_J$ [e.g. 31]:

$$\mathbf{c}_J^{(\text{MINRES})} = \|\mathbf{b}\|_2 \, \mathbf{Q}_J \left(\mathbf{R}^{-1} \mathcal{Q}_J^\top \right) \mathbf{e}_1. \tag{S8}$$

One way to perform MINRES is first running J iterations of the Lanczos algorithm, computing $\widetilde{\mathbf{T}}_J = \mathcal{Q}_J \mathbf{R}_J$, and then plugging the resulting $\mathbf{Q}_J$, $\mathcal{Q}_J$, and $\mathbf{R}_J$ into Eq. (S8). However, this is unsatisfactory as, naïvely it requires storing the $N \times J$ matrix $\mathbf{Q}_J$ [e.g. 31] so that $\mathbf{c}_J$ can be formed. Paige and Saunders instead introduce a vector recurrence to iteratively compute $\mathbf{c}_J^{(\text{MINRES})}$. This is possible because the QR factorizations of of successive $\widetilde{\mathbf{T}}_J$ may be related, allowing for the derivation of a simple update $\mathbf{c}_{J-1} \to \mathbf{c}_J$. This recurrence relation, which is given by Alg. 3 and broadly described below is exactly equivalent to Eq. (S8); however it uses careful bookkeeping to avoid storing any $N \times J$ terms.

First we note that the $\widetilde{\mathbf{T}}_J$ matrices are formed recursively, and thus their QR factorizations are also recursive:

$$\mathcal{Q}^\top \widetilde{\mathbf{T}}_J = \begin{bmatrix} \mathcal{Q}_{J-1}^\top & \mathcal{Q}^{\top(J,1:J-1)} \\ \mathcal{Q}^{\top(1:J-1,J+1)} & \mathcal{Q}^{(J,J+1)} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{T}}_{J-1} & \mathbf{t}^{(J)} \\ \mathbf{0}^\top & \|\mathbf{r}_J\| \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{J-1} & \mathbf{r}^{(J,1:J-1)} \\ \mathbf{0} & R^{(J,J)} \end{bmatrix} = \mathbf{R}_J$$

where $\mathbf{t}^{(J)}$ and $[\mathbf{r}^{(J,1:J-1)}; R^{(J,J)}]$ are the last columns of $\mathbf{T}_J$ and $\mathbf{R}_J$ respectively. Moreover, if we recursively form $\mathbf{R}_J^{-1}$ as

$$\mathbf{R}_J^{-1} = \begin{bmatrix} \mathbf{R}_{J-1} & \mathbf{r}^{(J,1:J-1)} \\ \mathbf{0} & R^{(J,J)} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}_{J-1}^{-1} & \left(\mathbf{R}_{J-1}^{-1} \mathbf{r}^{(J,1:J-1)} \right) / R^{(J,J)} \\ \mathbf{0} & 1/R^{(J,J)} \end{bmatrix},$$

then Eq. (S8) can be re-written in a decent-style update:

$$\mathbf{c}_J^{(\text{MINRES})} = \|\mathbf{b}\|_2 \begin{bmatrix} \mathbf{Q}_{J-1} & \mathbf{q}^{(J)} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{J-1}^{-1} & \frac{\mathbf{R}_{J-1}^{-1} \mathbf{r}^{(J,1:J-1)}}{R^{(J,J)}} \\ \mathbf{0} & 1/R^{(J,J)} \end{bmatrix} \begin{bmatrix} \mathcal{Q}_{J-1}^\top & \mathcal{Q}^{\top(J,1:J-1)} \\ \mathcal{Q}^{\top(1:J-1,J+1)} & \mathcal{Q}^{(J,J+1)} \end{bmatrix} \mathbf{e}_1$$

$$= \|\mathbf{b}\|_2 \begin{bmatrix} \mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1} & \frac{\mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1}\mathbf{r}^{(J,1:J-1)}}{R^{(J,J)}} \\ \mathbf{0} & 1/R^{(J,J)}\mathbf{q}_{J-1} \end{bmatrix} \begin{bmatrix} \mathcal{Q}_{J-1}^\top \mathbf{e}_1 \\ \mathcal{Q}^{\top(1,J+1)} \end{bmatrix}$$

$$= \underbrace{\left(\|\mathbf{b}\|_2 \mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1}\mathcal{Q}_{J-1}\mathbf{e}_1 \right)}_{\mathbf{c}_{J-1}^{(\text{MINRES})}} + \underbrace{\frac{\|\mathbf{b}\|_2 \mathcal{Q}^{\top(1,J+1)}}{R^{(J,J)}}}_{\varphi_J} \underbrace{\begin{bmatrix} \mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1}\mathbf{r}^{(J,1:J-1)} \\ \mathbf{q}_{J-1} \end{bmatrix}}_{\mathbf{d}_J}. \tag{S9}$$

Thus $\mathbf{c}_J^{(\text{MINRES})} = \mathbf{c}_{J-1}^{(\text{MINRES})} + \varphi_J \mathbf{d}_J$. The only seemingly expensive part of this update is computing $\mathbf{d}_J$, as we need to compute $\mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1}\mathbf{r}^{(J,1:J-1)}$. $\mathbf{r}^{(J,1:J-1)}$, which is the next entry in the QR factorization of $\widetilde{\mathbf{T}}_J$, can be cheaply computed using Givens rotations (see [e.g. 31, Ch. 11.4.1]). Moreover, only the last two entries of $\mathbf{r}^{(J,1:J-1)}$ will be non-zero (due to the tridiagonal structure of $\widetilde{\mathbf{T}}_J$). Consequently, we only need to store the last two vectors of $\mathbf{Q}_{J-1}\mathbf{R}_{J-1}^{-1}$, which again can be computed recursively.

In total, the whole procedure only requires the storage of ≈ 6 vectors. Each iteration requires a single MVM with $\mathbf{K}$ (to form the next Lanczos vector $\mathbf{q}_J$); and all subsequent operations are $\mathcal{O}(N)$. The entire procedure is given by Alg. 3. For simplicity, we have presented the algorithm as if run for a fixed number of steps J. In practice, the MINRES procedure admits inexpensive computation of the residual at each iteration [59] allowing for robust stopping criteria to be used.

Algorithm 3: Method of Minimum Residuals (MINRES).

Input : mvm_**K** ($\cdot$) – function for MVM with matrix **K**
    b – vector to solve against
Output: **c** = **K**$^{-1}$**b**.

$\mathbf{c}_1 \leftarrow \mathbf{0}$ // Current solution.
$\mathbf{d}_1, \mathbf{d}_0 \leftarrow \mathbf{0}$ // Current & prev. "search" direction.
$\varphi_2 \leftarrow \|\mathbf{b}\|_2$ // Current "step" size.

$\mathbf{q}_1 \leftarrow \mathbf{b}/\|\mathbf{b}\|_2$ // Current Lanczos vector.
$\mathbf{v}_1 \leftarrow$ mvm_**K**($\mathbf{q}_0$) // Buffer for MVM output.
$\delta_1 \leftarrow \|\mathbf{b}\|_2$ // Current Lanczos residual/sub-diagonal.
$\delta_0 \leftarrow 1$ // Prev. Lanczos residual/sub-diagonal.
$\eta_1 \leftarrow 1$ // Current scaling term.
$\eta_0 \leftarrow 0$ // Prev. scaling term.

for $j \leftarrow 2$ **to** J **do**
 // Run one iter of Lanczos. Gets next vector of **Q** matrix, and next
   diag/sub-diag (γ, δ) entries of **T** matrix.
 $\mathbf{q}_j \leftarrow \mathbf{v}_j/\delta_j$
 $\mathbf{v}_j \leftarrow$ mvm_**K**($\mathbf{q}_j$) $-\delta_j\mathbf{q}_{j-1}$
 $\gamma_j \leftarrow \mathbf{q}_j\mathbf{v}_j$
 $\mathbf{v}_j \leftarrow \mathbf{v}_j - \gamma_j\mathbf{q}_j$
 $\delta_j \leftarrow \|\mathbf{v}_j\|$
 // Compute the next $\mathbf{r}^{(J)}$ (part of QR) via Givens rotations. There
   are three non-0 entries: $\mathbf{R}^{(J,J-2:J)} = [\epsilon_J, \zeta_J, \eta_J]$.
 $\epsilon_j \leftarrow \delta_{j-1}\left(\delta_{j-2}/\sqrt{\delta_{j-2}^2 + \eta_{j-2}^2}\right)$
 $\zeta_j \leftarrow \delta_{j-1}\left(\eta_{j-2}/\sqrt{\delta_{j-2}^2 + \eta_{j-2}^2}\right)$
 $\eta_j \leftarrow \gamma_j\left(\eta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^2}\right) + \zeta_j\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^2}\right)$
 $\zeta_j \leftarrow \zeta_j\left(\eta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^2}\right) + \gamma_j\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^2}\right)$
 $\eta_j \leftarrow \eta_j\left(\eta_j/\sqrt{\delta_j^2 + \eta_j^2}\right)$
 // Compute "step" size $\varphi_J = \mathcal{Q}^{(1,J+1)}/R^{(J,J)}$.
 $\varphi_j \leftarrow \varphi_{j-1}\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^2}\right)\left(\eta_j/\sqrt{\delta_j^2 + \eta_j^2}\right)$
 // Update the current solution based on the $\mathbf{r}^{(J)}$ entries ($\epsilon_J, \zeta_J, \eta_J$)
   and previous search vectors $\mathbf{d}_{j-1}$, $\mathbf{d}_{j-2}$.
 $\mathbf{d}_j \leftarrow (\mathbf{q} - \zeta_j\mathbf{d}_{j-1} - \epsilon_j\mathbf{d}_{j-2})/\eta_j$
 $\mathbf{c}_j \leftarrow \mathbf{c}_{j-1} + \varphi_j\mathbf{d}_j$
end

return $\|\mathbf{b}\|_2\,\mathbf{c}_j$

C.2 Multi-Shift MINRES (msMINRES)

To adapt MINRES to multiple shifts (i.e. msMINRES), we exploit a well-established fact about the shift invariance of Krylov subspaces (see [e.g. 17, 24, 46, 65]).

Observation 1. *Let* $\mathbf{K}\mathbf{Q}_J = \mathbf{Q}_J\mathbf{T}_J + \mathbf{r}_J\mathbf{e}_J^\top$ *be the Lanczos factorization for* **K** *given the initial vector* **b**. *Then*

$$(\mathbf{K} + t\mathbf{I})\mathbf{Q}_J = \mathbf{Q}_J(\mathbf{T}_J + t\mathbf{I}) + \mathbf{r}_J\mathbf{e}_J^\top$$

is the Lanczos factorization for matrix $(\mathbf{K} + t\mathbf{I})$ *with initial vector* **b**.

In other words, if we run Lanczos on **K** and **b**, then we get the Lanczos factorization of $(\mathbf{K} + t\mathbf{I})$ *for free*, without any additional MVMs! Consequently, we can re-use the $\mathbf{Q}_J$ and $\mathbf{T}_J$ Lanczos matrices

Algorithm 4: Multi-shift MINRES (msMINRES). Differences from MINRES (Alg. 3) are in blue. Blue `for` loops are parallelizable.

Input : `mvm_K`($\cdot$) – function for MVM with matrix $\mathbf{K}$
  $\mathbf{b}$ – vector to solve against
  $t_1, \ldots, t_Q$ – shifts
Output : $\mathbf{c}_1 = (\mathbf{K} + t_1)^{-1}\mathbf{b}, \ldots, \mathbf{c}_Q = (\mathbf{K} + t_Q)^{-1}\mathbf{b}$.

$\mathbf{q}_1 \leftarrow \mathbf{b}/\|\mathbf{b}\|_2$ // Current Lanczos vector.
$\mathbf{v}_1 \leftarrow$ `mvm_K`($\mathbf{q}_0$) // Buffer for MVM output.
$\delta_1 \leftarrow \|\mathbf{b}\|_2, \delta_0 \leftarrow 1$ // Current/prev. Lanczos residual/sub-diagonal.
for $q \leftarrow 1$ **to** Q **do**
  $\mathbf{c}_1^{(q)} \leftarrow \mathbf{0}$ // Current solution.
  $\mathbf{d}_1^{(q)}, \mathbf{d}_0^{(q)} \leftarrow \mathbf{0}$ // Current & prev. "search" direction.
  $\varphi_2^{(q)} \leftarrow \|\mathbf{b}\|_2$ // Current "step" size.
  $\eta_1^{(q)} \leftarrow 1, \eta_0^{(q)} \leftarrow 0$ // Current/prev. scaling term.
end
for $j \leftarrow 2$ **to** J **do**
  $\mathbf{q}_j \leftarrow \mathbf{v}_j/\delta_j$
  $\mathbf{v}_j \leftarrow$ `mvm_K`($\mathbf{q}_j$) $-\delta_j\mathbf{q}_{j-1}$
  $\gamma_j \leftarrow \mathbf{q}_j\mathbf{v}_j$
  $\mathbf{v}_j \leftarrow \mathbf{v}_j - \gamma_j\mathbf{q}_j$
  $\delta_j \leftarrow \|\mathbf{v}_j\|$
  for $q \leftarrow 1$ **to** Q **do**
    $\epsilon_j^{(q)} \leftarrow \delta_{j-1}\left(\delta_{j-2}/\sqrt{\delta_{j-2}^2 + \eta_{j-2}^{(q)2}}\right)$
    $\zeta_j^{(q)} \leftarrow \delta_{j-1}\left(\eta_{j-2}^{(q)}/\sqrt{\delta_{j-2}^2 + \eta_{j-2}^{(q)2}}\right)$
    $\eta_j^{(q)} \leftarrow (\gamma_j + t_q)\left(\eta_{j-1}^{(q)}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^{(q)2}}\right) + \zeta_j^{(q)}\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^{(q)2}}\right)$
    $\zeta_j^{(q)} \leftarrow \zeta_j^{(q)}\left(\eta_{j-1}^{(q)}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^{(q)2}}\right) + (\gamma_j + t_q)\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^{(q)2}}\right)$
    $\eta_j^{(q)} \leftarrow \eta_j^{(q)}\left(\eta_j^{(q)}/\sqrt{\delta_j^2 + \eta_j^{(q)2}}\right)$
    $\varphi_j^{(q)} \leftarrow \varphi_{j-1}^{(q)}\left(\delta_{j-1}/\sqrt{\delta_{j-1}^2 + \eta_{j-1}^{(q)2}}\right)\left(\eta_j^{(q)}/\sqrt{\delta_j^2 + \eta_j^{(q)2}}\right)$
    $\mathbf{d}_j^{(q)} \leftarrow \left(\mathbf{q} - \zeta_j^{(q)}\mathbf{d}_{j-1}^{(q)} - \epsilon_j^{(q)}\mathbf{d}_{j-2}^{(q)}\right)/\eta_j^{(q)}$
    $\mathbf{c}_j^{(q)} \leftarrow \mathbf{c}_{j-1}^{(q)} + \varphi_j^{(q)}\mathbf{d}_j^{(q)}$
  end
end
return $\|\mathbf{b}\|_2\,\mathbf{c}_j$

to compute *multiple shifted solves*.

$$(\mathbf{K} + t\mathbf{I})^{-1}\mathbf{b} \approx \|\mathbf{b}\|_2\,\mathbf{Q}_J\left(\mathbf{R}_J^{(t)-1}\boldsymbol{\mathcal{Q}}_J^{(t)\top}\right)\mathbf{e}_1, \quad \boldsymbol{\mathcal{Q}}_J^{(t)}\mathbf{R}_J^{(t)} = \begin{bmatrix}\mathbf{T}_J + t\mathbf{I} \\ \|\mathbf{r}_J\|_2\mathbf{e}_J^\top\end{bmatrix}, \qquad \text{(S10)}$$

Assuming $\mathbf{Q}$ and $\mathbf{T}$ have been previously computed, Eq. (S10) requires no additional MVMs with $\mathbf{K}$. We refer to this multi-shift formulation as **Multi-Shift MINRES**, or **msMINRES**.

A simple vector recurrence for msMINRES. Just as with standard MINRES, Eq. (S10) can also be computed via a vector recurrence. We can derive a msMINRES algorithm simply by modifying the existing MINRES recurrence. Before the QR step in Alg. 3, we add t to the Lanczos diagonal terms ($\gamma_j + t$, where $\gamma_j = T^{(j,j)}$). This can be extended to simultaneously handle *multiple shifts* $t_1, \ldots, t_Q$. Each shift would compute its own QR factorization, its own step size $\varphi_j^{(t_q)}$, and its own

search vector $\mathbf{d}_j^{(t_q)}$. However, all shifts share the same Lanczos vectors $\mathbf{q}_j$ and therefore share the same MVMs. The operations for each shift can be vectorized for efficient parallelization.

To summarize: the resulting algorithm—msMINRES—gives us approximations to $(t_1\mathbf{I} + \mathbf{K})^{-1}\mathbf{b}$, $\ldots$, $(t_Q\mathbf{I} + \mathbf{K})^{-1}$ *essentially for free* by leveraging the information we needed anyway to compute $\mathbf{K}^{-1}\mathbf{b}$. Alg. 4 outlines the procedure; below we re-highlight its computational properties:

Property 1 (Restated) (Computation/Memory of msMINRES-CIQ). *J iterations of msMINRES requires exactly J matrix-vector multiplications (MVMs) with the input matrix $\mathbf{K}$, regardless of the number of quadrature points Q. The resulting runtime of msMINRES-CIQ is $\mathcal{O}(J\xi(\mathbf{K}))$, where $\xi(\mathbf{K})$ is the time to perform an MVM with $\mathbf{K}$. The memory requirement is $\mathcal{O}(QN)$ in addition to what's required to store $\mathbf{K}$.*

D Preconditioning msMINRES-CIQ

To improve the convergence of Thm. 1, we can introduce a preconditioner $\mathbf{P}$ where $\mathbf{P}^{-1}\mathbf{K} \approx \mathbf{I}$. For standard MINRES, applying a preconditioner is straightforward. We simply use MINRES to solve the system

$$\left(\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2}\right)\mathbf{P}^{1/2}\mathbf{c} = \mathbf{P}^{-1/2}\mathbf{b},$$

which has the same solution $\mathbf{c}$ as the original system. In practice the preconditioned MINRES vector recurrence does not need access to $\mathbf{P}^{-1/2}$—it only needs access to $\mathbf{P}^{-1}$ (see [12, Ch. 3.4] for details).

However, it is not immediately straightforward to apply preconditioning to msMINRES, as preconditioners break the shift-invariance property that is necessary for the $\mathcal{O}(JN^2)$ shifted solves [3, 46]. More specifically, if we apply $\mathbf{P}$ to msMINRES, then we obtain the solves

$$\mathbf{P}^{-1/2}(\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2} + t_q\mathbf{I})^{-1}(\mathbf{P}^{-1/2}\mathbf{b}).$$

Plugging these shifted solves into the quadrature equation Eq. (2) therefore gives us

$$\widetilde{\mathbf{a}}_J \approx \mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{b}). \tag{S11}$$

In general, we cannot recover $\mathbf{K}^{-1/2}$ from Eq. (S11). Nevertheless, we can still obtain preconditioned solutions that are equivalent to $\mathbf{K}^{-1/2}\mathbf{b}$ and $\mathbf{K}^{1/2}\mathbf{b}$ up to an orthogonal rotation. Let $\mathbf{R} = \mathbf{K}\mathbf{P}^{-1/2}(\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2})^{-1/2}$. We have that

$$\mathbf{R}\mathbf{R}^\top = \mathbf{K}\left(\mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\right)\left((\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{K} = \mathbf{K}.$$

Thus $\mathbf{R}$ is equivalent to $\mathbf{K}^{1/2}$ up to orthogonal rotation. We can compute $\mathbf{R}\mathbf{b}$ (e.g. for sampling) by applying Eq. (S11) to the initial vector $\mathbf{P}^{1/2}\mathbf{b}$:

$$\mathbf{R}\mathbf{b} = \mathbf{K}\underbrace{\left[\mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{P}^{-\frac{1}{2}}\right]\left(\mathbf{P}^{\frac{1}{2}}\mathbf{b}\right)}_{\text{Applying preconditioned msMINRES to } \mathbf{P}^{1/2}\mathbf{b}}. \tag{S12}$$

Similarly, $\mathbf{R}' = \mathbf{P}^{-1/2}\left(\mathbf{P}^{-1/2}\mathbf{K}\mathbf{P}^{-1/2}\right)^{-1/2}$ is equivalent to $\mathbf{K}^{-1/2}$ up to orthogonal rotation:

$$\mathbf{R}'\mathbf{R}'^\top = \left(\mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\right)\left((\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{P}^{-\frac{1}{2}}\right) = \mathbf{K}^{-1}.$$

We can compute $\mathbf{R}'\mathbf{b}$ (e.g. for whitening) via:

$$\mathbf{R}'\mathbf{b} = \underbrace{\left[\mathbf{P}^{-\frac{1}{2}}(\mathbf{P}^{-\frac{1}{2}}\mathbf{K}\mathbf{P}^{-\frac{1}{2}})^{-\frac{1}{2}}\mathbf{P}^{-\frac{1}{2}}\right]\left(\mathbf{P}^{\frac{1}{2}}\mathbf{b}\right)}_{\text{Applying preconditioned msMINRES to } \mathbf{P}^{1/2}\mathbf{b}}. \tag{S13}$$

Crucially, the convergence of Eqs. (S12) and (S13) depends on the conditioning $\kappa(\mathbf{P}^{-1}\mathbf{K}) \ll \kappa(\mathbf{K})$.

As with standard MINRES, msMINRES only requires access to $\mathbf{P}^{-1}$, not $\mathbf{P}^{-1/2}$. Note however that Eqs. (S12) and (S13) both require multiplies with $\mathbf{P}^{1/2}$. If a preconditioner $\mathbf{P}$ does not readily decompose into $\mathbf{P}^{1/2}\mathbf{P}^{1/2}$, we can simply run the CIQ algorithm on $\mathbf{P}$ to compute $\mathbf{P}^{1/2}\mathbf{b}$. Thus our requirements for a preconditioner are:

1) it affords efficient solves (ideally $o(N^2)$), and

2) it affords efficient MVMs (also ideally $o(N^2)$) for computing $\mathbf{P}^{1/2}\mathbf{b}$ via CIQ.

In our experiments we use the partial pivoted Cholesky preconditioner proposed by Gardner et al. [29], which satisfies the above requirements. The form of $\mathbf{P}$ is $\bar{\mathbf{L}}\bar{\mathbf{L}}^\top + \sigma^2\mathbf{I}$, where $\bar{\mathbf{L}}$ is a low-rank factor (produced by the partial pivoted Cholesky factorization [37]) and $\sigma^2\mathbf{I}$ is a small diagonal component. This preconditioner affords $\approx \mathcal{O}(N)$ MVMs by exploiting its low rank structure and $\approx \mathcal{O}(N)$ solves using the matrix inversion lemma. Moreover, this preconditioner is highly effective on many Gaussian covariance matrices [29, 79].

E $\quad \mathcal{O}(M^2)$ Natural Gradient Updates

When performing variational inference, we must optimize the $\mathbf{m}'$ and $\mathbf{S}'$ parameters of the whitened variational distribution $q(\mathbf{u}') = \mathcal{N}(\mathbf{m}', \mathbf{S}')$. Rather than using standard gradient descent methods on these parameters, many have suggested that **natural gradient descent (NGD)** is better suited for variational inference [38, 45, 66]. NGD performs the following update:

$$[\mathbf{m}' \quad \mathbf{S}'] \leftarrow [\mathbf{m}' \quad \mathbf{S}'] - \varphi \boldsymbol{\mathcal{F}}^{-1} \left[\frac{\partial \text{ELBO}}{\partial \mathbf{m}'} \quad \frac{\partial \text{ELBO}}{\partial \mathbf{S}'} \right] \tag{S14}$$

where φ is a step size, $\left[\frac{\partial \text{ELBO}}{\partial \mathbf{m}'} \quad \frac{\partial \text{ELBO}}{\partial \mathbf{S}'} \right]$ is the ELBO gradient, and $\boldsymbol{\mathcal{F}}$ is the *Fisher information matrix* of the variational parameters. Conditioning the gradient with $\boldsymbol{\mathcal{F}}^{-1}$ results in descent directions that are better suited towards distributional parameters [45].

For Gaussian distributions (and other exponential family distributions) the Fisher information matrix does not need to be explicitly computed. Instead, there is a simple closed-form update that relies on different parameterizations of the Gaussian $\mathcal{N}(\mathbf{m}', \mathbf{S}')$:

$$[\boldsymbol{\theta} \quad \boldsymbol{\Theta}] \leftarrow [\boldsymbol{\theta} \quad \boldsymbol{\Theta}] - \varphi \left[\frac{\partial \text{ELBO}}{\partial \boldsymbol{\eta}} \quad \frac{\partial \text{ELBO}}{\partial \mathbf{H}} \right]. \tag{S15}$$

$[\boldsymbol{\theta}, \ \boldsymbol{\Theta}]$ are the Gaussian's *natural parameters* and $[\boldsymbol{\eta}, \ \mathbf{H}]$ are the Gaussian's *expectation parameters*:

$$\boldsymbol{\theta} = \mathbf{S}'^{-1}\mathbf{m}', \quad \boldsymbol{\Theta} = -\frac{1}{2}\mathbf{S}'^{-1},$$

$$\boldsymbol{\eta} = \mathbf{m}', \quad \mathbf{H} = \mathbf{m}'\mathbf{m}'^\top + \mathbf{S}'$$

In many NGD implementations, it is common to store the variational parameters via their natural representation $(\boldsymbol{\theta}, \boldsymbol{\Theta})$, compute the ELBO via the standard parameters $(\mathbf{m}', \mathbf{S}')$, and then compute the derivative via the expectation parameters $(\boldsymbol{\eta}, \mathbf{H})$. Unfortunately, converting between these three parameterizations requires $\mathcal{O}(M^3)$ computation. (To see why this is the case, note that computing $\mathbf{S}'$ essentially requires inverting the $\boldsymbol{\Theta}$ matrix.)

A $\mathcal{O}(M^2)$ NGD update. In what follows, we will demonstrate that the ELBO and its derivative can be computed from $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ in $\mathcal{O}(M^2)$ time via careful bookkeeping. Consequently, NGD updates have the same asymptotic complexity as the other computations required for SVGP. Recall that the ELBO is given by

$$\text{ELBO} = \sum_{i=1}^{N} \overbrace{\underset{q(f(\mathbf{x}^{(i)}))}{\mathbb{E}} \left[\log p(y^{(i)} \mid f(\mathbf{x}^{(i)})) \right]}^{\text{expected log likelihood}} - \text{KL}\left[q(\mathbf{u}) \| p(\mathbf{u}) \right]$$

We will separately analyze the expected log likelihood and KL divergence computations.

E.1 The Expected Log Likelihood and its Gradient

Assume we are estimating the ELBO from a single data point $\mathbf{x}, y$. The expected log likelihood term of the ELBO is typically computed via Gauss-Hermite quadrature or Monte Carlo integration [40]:[10]

$$\underset{q(f(\mathbf{x}))}{\mathbb{E}} \left[\log p(y \mid f(\mathbf{x})) \right] = \sum_{s=1}^{S} w_s p(y \mid f_s), \quad f_s = \mu^*_{\text{aprx}}(\mathbf{x}) + \text{Var}^*_{\text{aprx}}(\mathbf{x})^{1/2} \varepsilon_s$$

---

[10]It can also be computed analytically for Gaussian distributions [39]. The analytic form achieves the same derivative decomposition as in Eq. (S16) and so the following analysis will still apply.

where w_s are the quadrature weights (or $1/S$ for MC integration) and ε_s are the quadrature locations (or samples from $\mathcal{N}(0,1)$ for MC integration). Therefore, the variational parameters only interact with the expected log likelihood term via $\mu_{\text{aprx}}^*(\mathbf{x})$ and $\text{Var}_{\text{aprx}}^*(\mathbf{x})$. We can write its gradients via chain rule as:

$$\frac{\partial\,\mathbb{E}_{q(f(\mathbf{x}))}\left[\log p(y|f(\mathbf{x}))\right]}{\partial\boldsymbol{\eta}} = c_1\,\frac{\partial\mu_{\text{aprx}}^*(\mathbf{x})}{\partial\boldsymbol{\eta}} + c_2\,\frac{\partial\text{Var}_{\text{aprx}}^*(\mathbf{x})}{\partial\boldsymbol{\eta}}$$

$$\frac{\partial\,\mathbb{E}_{q(f(\mathbf{x}))}\left[\log p(y|f(\mathbf{x}))\right]}{\partial\mathbf{H}} = c_3\,\frac{\partial\mu_{\text{aprx}}^*(\mathbf{x})}{\partial\mathbf{H}} + c_4\,\frac{\partial\text{Var}_{\text{aprx}}^*(\mathbf{x})}{\partial\mathbf{H}} \tag{S16}$$

for some constants c_1, c_2, c_3, and c_4 that do not depend on the variational parameters. It thus suffices to show that the posterior mean/variance and their gradients can be computed from $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ in $\mathcal{O}(M^2)$ time.

The predictive distribution and its gradient. All expensive computations involving $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ are written in blue.

$\mu_{\text{aprx}}^*(\mathbf{x})$ and its derivative can be written as:

$$\mu_{\text{aprx}}^*(\mathbf{x}) = \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{m}' \qquad\qquad \text{(standard parameters)}$$

$$= \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\boldsymbol{\eta} \qquad\qquad \text{(expectation parameters)}$$

$$= \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}(-2\boldsymbol{\Theta})^{-1}\boldsymbol{\theta}, \tag{S17}$$

$$\frac{\partial\mu_{\text{aprx}}^*(\mathbf{x})}{\partial\boldsymbol{\eta}} = \mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}, \tag{S18}$$

$$\frac{\partial\mu_{\text{aprx}}^*(\mathbf{x})}{\partial\mathbf{H}} = \mathbf{0}.$$

$\text{Var}_{\text{aprx}}^*(\mathbf{x})$ and its derivative can be written as:

$$\text{Var}_{\text{aprx}}^*(\mathbf{x}) = \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\left(\mathbf{S}'-\mathbf{I}\right)\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}} \qquad\qquad \text{(standard parameters)}$$

$$= \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\left(\mathbf{H}-\boldsymbol{\eta}\boldsymbol{\eta}^\top-\mathbf{I}\right)\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}} \qquad \text{(expectation parameters)}$$

$$= \mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\left((-2\boldsymbol{\Theta})^{-1}-\mathbf{I}\right)\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}, \tag{S19}$$

$$\frac{\partial\text{Var}_{\text{aprx}}^*(\mathbf{x})}{\partial\boldsymbol{\eta}} = -2\left(\mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}(-2\boldsymbol{\Theta})^{-1}\boldsymbol{\theta}\right)\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}, \tag{S20}$$

$$\frac{\partial\text{Var}_{\text{aprx}}^*(\mathbf{x})}{\partial\mathbf{H}} = \left(\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}^\top\right)\left(\mathbf{k}_{\mathbf{Zx}}^\top\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\right). \tag{S21}$$

In Eqs. (S17) to (S21), the only expensive operation involving $\mathbf{K}_{\mathbf{ZZ}}$ is $\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}$, which can be computed with CIQ. The only expensive operation involving the variational parameters is $(-2\boldsymbol{\Theta})^{-1}\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}$, which can be computed with preconditioned conjugate gradients after computing $\mathbf{K}_{\mathbf{ZZ}}^{-1/2}\mathbf{k}_{\mathbf{Zx}}$.[11] Those operations only need to be computed once, and then they can be reused across Eqs. (S17) to (S21). In total, the entire computation for the expected log likelihood and its derivative is $\mathcal{O}(M^2)$.

E.2 The KL Divergence and its Gradient

We will demonstrate that the KL divergence and its gradient can be computed from $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ in $\mathcal{O}(M^2)$ time. All expensive computations involving $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ are written in blue.

The whitened KL divergence from Sec. 5.1 is given by:

$$\text{KL}\left[\,q(\mathbf{u}')\|p(\mathbf{u}')\,\right] = \frac{1}{2}\left[\mathbf{m}'^\top\mathbf{m}' + \text{Tr}\left(\mathbf{S}'\right) - \log|\mathbf{S}'| - M\right] \qquad \text{(standard parameters)}$$

$$= \frac{1}{2}\left[\text{Tr}\left(\mathbf{H}\right) - \log|\mathbf{H}-\boldsymbol{\eta}\boldsymbol{\eta}^\top| - M\right] \qquad \text{(expectation parameters)}$$

$$= \frac{1}{2}\left[\boldsymbol{\theta}^\top(-2\boldsymbol{\Theta})^{-2}\boldsymbol{\theta} + \text{Tr}\left((-2\boldsymbol{\Theta})^{-1}\right) + \log|-2\boldsymbol{\Theta}| - M\right]. \tag{S22}$$

---

[11] We typically apply a Jacobi preconditioner to these solves.

The KL derivative with respect to $\boldsymbol{\eta}$ and $\mathbf{H}$ is surprisingly simple when re-written in terms of the natural parameters

$$\frac{\partial \operatorname{KL}\left[\,q(\mathbf{u}')\|p(\mathbf{u}')\,\right]}{\partial \boldsymbol{\eta}} = \left(\mathbf{H} - \boldsymbol{\eta}\boldsymbol{\eta}^\top\right)^{-1}\boldsymbol{\eta} = (\mathbf{S}')^{-1}\boldsymbol{\eta}$$
$$= \boldsymbol{\theta} \tag{S23}$$

$$\frac{\partial \operatorname{KL}\left[\,q(\mathbf{u}')\|p(\mathbf{u}')\,\right]}{\partial \mathbf{H}} = \frac{1}{2}\mathbf{I} - \frac{1}{2}\left(\mathbf{H} - \boldsymbol{\eta}\boldsymbol{\eta}^\top\right)^{-1} = \frac{1}{2}\mathbf{I} - \frac{1}{2}(\mathbf{S}')^{-1}$$
$$= \frac{1}{2}\mathbf{I} + \boldsymbol{\Theta}. \tag{S24}$$

Thus the derivative of the KL divergence only takes $\mathcal{O}(M^2)$ time to compute. The forward pass can also be computed in $\mathcal{O}(M^2)$ time—using stochastic trace estimation for the trace term [16, 29], stochastic Lanczos quadrature for the log determinant [20, 76], and CG for the solves. However, during training the forward pass can be omitted as only the gradient is needed for NGD steps.

F   Experimental Details

SVGP experiments. Each dataset is randomly split into 75% training, 10% validation, and 15% testing sets; $\mathbf{x}$ and y values are scaled to be zero mean and unit variance. All models use a constant mean and a Matérn 5/2 kernel, with lengthscales initialized to 0.01 and inducing points initialized by K-means clustering. Each model is trained for 20 epochs with a minibatch size of 256.[12] We alternate between optimizing $\mathbf{m}'/\mathbf{S}'$ and the other parameters, using NGD for the former and Adam [48] for the latter. Each optimizer uses an initial learning rate of 0.01[13], decayed by $10\times$ at epochs 1, 5, 10, and 15. For CIQ we use $Q = 15$ quadrature points. msMINRES terminates when the $\mathbf{c}_j$ vectors achieve a relative norm of 0.001 or after $J = 200$ iterations. We experimented with tighter tolerances and found no difference in the models' final accuracy. (Note that $J = 200$ is almost always enough to achieve the desired 0.001 tolerance; see Fig. S7.) Results are averaged over three trials.

The 3DRoad [34] and CovType [9] datasets are available from the UCI repository [2]. For 3Droad, we only use the first two features—corresponding to latitude and longitude. For CovType, we reduce the 7-way classification problem to a binary problem (Cover_Type $\in \{2,3\}$ versus Cover_Type $\in \{0, 1, 4, 5, 6\}$). The Precipitation dataset [52, 53] is available from the IRI/LDEO Climate Data Library.[14] This spatio-temporal dataset aims to predict the "WASP" index (Weighted Anomaly Standardized Precipitation) at various latitudes/longitudes. Each data point corresponds to the WASP index for a given year (between 2010 and 2019)—which is the average of monthly WASP indices. In total, there are 10 years and $10{,}127$ latitude/longitude coordinates, for a total dataset size of $101{,}270$.

Bayesian optimization experiments. The 6-dimensional Hartmann function is a classical test problem in global optimization[15]. There are 6 local minima and a global optimal value is -3.32237. We use a total of 100 evaluations with 10 initial points. The 10 initial points are generated using a Latin hypercube design and we use a batch size of 5. In each iteration, we draw 5 samples and select 5 new trials to evaluate in parallel.

We consider the same setup and controller as in [21] for the 12-dimensional Lunar Lander problem. The goal is to learn a controller that minimizes fuel consumption and distance to a given landing target while also preventing crashes. The state of the lunar lander is given by its angle and position, and their time derivatives. Given this state vector, the controller chooses one of the following four actions: $a \in \{$do nothing, booster left, booster right, booster down$\}$. The objective is the average final reward over a fixed constant set of 50 randomly generated terrains, initial positions, and initial velocities. The optimal controller achieves an average reward of ≈ 309 over the 50 environments.

---

[12]The batch size is 512 on the Covtype dataset due to its larger size.

[13]On the Precipitation dataset, the initial learning rate is 0.005 for NGD stability with the Student-T likelihood.

[14]A processed version of the dataset is available at https://github.com/gpleiss/ciq_experiments/tree/main/svgp/data. Original source of data: http://iridl.ldeo.columbia.edu/maproom/Global/Precipitation/WASP_Indices.html.

[15]https://www.sfu.ca/~ssurjano/hart6.html

For both problems, we use a Matérn-5/2 kernel with ARD and a constant mean function. The domain is scaled to $[0, 1]^d$ and we standardize the function values before fitting the Gaussian process. The kernel hyperparameters are optimized using L-BFGS-B and we use the following bounds: (lengthscale) $\ell \in [0.01, 2.0]$, (signal variance) $s^2 \in [0.05, 50.0]$, (noise variance) $\sigma^2 \in [1e{-}6, 1e{-}2]$. Additionally, we place a horseshoe prior on the noise variance as recommended in [72]. We add $1e{-}4$ to the diagonal of the kernel matrix to improve the conditioning and use a preconditioner of rank 200 for CIQ.

Image reconstruction experiments. The matrix $\mathbf{A} = \boldsymbol{D}\boldsymbol{B}$ is given as the product of two matrices $\boldsymbol{D}$ and $\boldsymbol{B}$. Here $\boldsymbol{B}$ is a $N^2 \times N^2$ Gaussian blur matrix with a blur radius of 2.5 pixels and filter size of 5 pixels. The binary matrix $\boldsymbol{D}$ is a $KM^2 \times N^2$ downsampling or decimation matrix that connects the $N \times N$ high-resolution image to the $M \times M$ low-resolution images. For the hyperparameters γ_{obs} and γ_{prior} we choose Jeffrey's hyperpriors, i.e.

$$p(\gamma_{\mathrm{obs}}) \propto \gamma_{\mathrm{obs}}^{-1} \qquad \text{and} \qquad p(\gamma_{\mathrm{prior}}) \propto \gamma_{\mathrm{prior}}^{-1} \tag{S25}$$

In order to conduct the experiment we use the observation likelihood with $\gamma_{\mathrm{obs}} = 1$ to sample $K = 4$ low-resolution images $\mathbf{y}_{1:K}$ from the high-resolution image. The discrete Laplacian matrix $\mathbf{L}$ is defined by the following isotropic filter:

$$\mathbf{L}_{\mathrm{filter}} = \frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{S26}$$

For both $\mathbf{L}$ and $\boldsymbol{B}$ we implicitly use reflected (i.e. non-periodic) boundary conditions. We use a CG tolerance of 0.001 and a maximum of $J = 400$ msMINRES iterations. We use a Jacobi preconditioner for CG. We draw 1000 samples from the Gibbs sampler and treat the first 200 samples as burn-in. The reconstructed image depicted in the main text is the (approximate) posterior mean. In the main text we provided the conditional posterior for the latent image $\mathbf{x}$. To complete the specification of the Gibbs sampler we also need the posterior conditionals for γ_{obs} and γ_{prior}, both of which are given by gamma distributions:

$$\begin{aligned} p(\gamma_{\mathrm{obs}}|\mathbf{x}, \mathbf{y}_{1:K}) &= \mathrm{Ga}(\gamma_{\mathrm{obs}}|\alpha = 1 + \tfrac{KM^2}{2}, \beta = 2/||\mathbf{y}_{1:K} - \mathbf{A}\mathbf{x}||^2) \\ p(\gamma_{\mathrm{prior}}|\mathbf{x}) &= \mathrm{Ga}(\gamma_{\mathrm{prior}}|\alpha = 1 + \tfrac{N^2-1}{2}, \beta = 2/||\mathbf{L}\mathbf{x}||^2) \end{aligned} \tag{S27}$$

G Proof of Theorem 1

To prove the convergence result in Thm. 1, we first prove the following lemmas.

Lemma 2. *Let* $\mathbf{K} \succ 0$ *be symmetric positive definite and let shifts* $t_1, \ldots, t_Q > 0$ *be real-valued and positive. After* J *iterations of msMINRES, all shifted solve residuals are bounded by:*

$$\left\| (\mathbf{K} + t_q\mathbf{I})\mathbf{c}_J^{(q)} - \mathbf{b} \right\|_2 \leq \left(\frac{\sqrt{\kappa(\mathbf{K} + t_q\mathbf{I})} - 1}{\sqrt{\kappa(\mathbf{K} + t_q\mathbf{I})} + 1} \right)^J \|\mathbf{b}\|_2 \leq \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^J \|\mathbf{b}\|_2,$$

where $\mathbf{b}$ *is the vector to solve against,* $\mathbf{c}_J^{(1)}, \ldots, \mathbf{c}^{(Q)}$ *are the msMINRES outputs, and* $\kappa(\mathbf{K})$ *is the condition number of* $\mathbf{K}$.

Proof. The convergence proof uses a polynomial bound, which is the standard approach for Krylov algorithms. See [e.g. 65, 69, 75] for an analogous proof for the conjugate gradients method and [e.g. 32] for a treatment of MINRES applied to both positive definite and indefinite systems.

At iteration J, the msMINRES algorithm produces:

$$\mathbf{c}_J^{(q)} = \operatorname*{arg\,min}_{\mathbf{c}^{(q)} \in \mathcal{K}_J(\mathbf{K}, \mathbf{b})} \left[\left\| (\mathbf{K} + t_q\mathbf{I})\mathbf{c}^{(q)} - \mathbf{b} \right\|_2 \right], \quad q = 1, \ldots Q, \tag{S28}$$

where without loss of generality we assume $\mathbf{c}_0^{(q)} = \mathbf{0}$ for simplicity. Using the fact that Krylov subspaces are shift invariant, we immediately have that

$$\mathbf{c}_J^{(q)} = \operatorname*{arg\,min}_{\mathbf{c}^{(q)} \in \mathcal{K}_J(\mathbf{K} + t_q\mathbf{I}, \mathbf{b})} \left[\left\| (\mathbf{K} + t_q\mathbf{I})\mathbf{c}^{(q)} - \mathbf{b} \right\|_2 \right], \quad q = 1, \ldots Q. \tag{S29}$$

Since $(\mathbf{K} + t_q \mathbf{I}) \succ 0$ we may invoke a result on MINRES error bounds for symmetric positive definite matrices [32, Chapter 3] to conclude that

$$\left\| (\mathbf{K} + t_q \mathbf{I}) \mathbf{c}_J^{(q)} - \mathbf{b} \right\|_2 \le \left(\frac{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} - 1}{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} + 1} \right)^J \|\mathbf{b}\|_2.$$

Observing that $\kappa(\mathbf{K} + t_q \mathbf{I}) \ge \kappa(\mathbf{K})$ for all q since $t_q > 0$ concludes the proof.

$\square$

Lemma 2 is a very loose bound, as it doesn't assume anything about the spectrum of $\mathbf{K}$ (which is standard for generic Krylov method error bounds) and upper bounds the residual error for every shift using the most ill-conditioned system. In practice, we find that smMINRES converges for many covariance matrices with $J \approx 100$, even when the conditioning is on the order of $\kappa(\mathbf{K}) \approx 10^4$ and this convergence can be further improved with preconditioning.

Lemma 3. *For any positive definite $\mathbf{K}$ and positive t, we have*

$$\frac{\sqrt{\kappa(\mathbf{K} + t\mathbf{I})} - 1}{\sqrt{\kappa(\mathbf{K} + t\mathbf{I})} + 1} = \frac{\sqrt{\lambda_{max} + t} - \sqrt{\lambda_{min} + t}}{\sqrt{\lambda_{max} + t} + \sqrt{\lambda_{min} + t}} < \frac{\lambda_{max}}{4t} \tag{S30}$$

Proof. We can upper bound the numerator

$$\sqrt{\lambda_{\max} + t} - \sqrt{\lambda_{\min} + t} \le \sqrt{\lambda_{\max} + t} - \sqrt{t}$$
$$= \sqrt{\lambda_{\max}} \left(\sqrt{1 + t/\lambda_{\max}} - \sqrt{t/\lambda_{\max}} \right) \le \sqrt{\lambda_{\max}} \frac{1}{2\sqrt{t/\lambda_{\max}}} = \frac{\lambda_{\max}}{2\sqrt{t}}.$$

where we have applied the standard inequality $\sqrt{(\cdot) + 1} - \sqrt{(\cdot)} < \frac{1}{2\sqrt{(\cdot)}}$. The denominator can be (loosely) lower-bounded as $2\sqrt{t}$. Combining these two bounds completes the proof. $\square$

Lemma 4. *Let σ_q^2 and $\widetilde{w}_q$ be defined as in Eq. (S4). Then*

$$\sum_{q=1}^Q \frac{|w_q|}{|t_q|} = \sum_{q=1}^Q \frac{|\widetilde{w}_q|}{|\sigma_q^2|} < \frac{4Q \log \left(5\sqrt{\kappa(\mathbf{K})} \right)}{\pi \sqrt{\lambda_{min}}}$$

where $w_q = -\widetilde{w}_q$ and $t_q = -\sigma_q^2$ as used in Eq. (S5).

Proof. Using facts about elliptical integrals we have

$$\mathcal{K}'(k) < \log(1 + 4/k) \le \log(5/k) \qquad k \in (0, 1) \qquad ([62, \text{Thm. } 1.7] \text{ and } [84, \text{Thm. } 2])$$
$$\frac{\pi}{2} \le \mathcal{K}(k) \qquad k \in [0, 1] \qquad ([\text{e.g. } 62])$$

where in the first statement we have used that $\mathcal{K}'(k) = \mathcal{K}(k')$. For Jacobi elliptic functions we have that

$$0 < \text{dn}(u\mathcal{K}(k)|k) < 1 \qquad u \in (0, 1),\ k \in (0, 1) \qquad ([\text{e.g. } 56])$$
$$0 < \text{sn}(u\mathcal{K}(k)|k) < 1 \qquad u \in (0, 1),\ k \in (0, 1) \qquad ([\text{e.g. } 56])$$
$$\text{sn}(\pi u/2|0) < \text{sn}(u\mathcal{K}(k)|k) < 1 \qquad u \in (0, 1),\ k \in (0, 1) \qquad ([10, \text{Thm. } 1])$$

where in the last inequality we have used that $\mathcal{K}(0) = \pi/2$ [e.g. 1]. Coupling the final inequality above with $\text{sn}(\pi u/2|0) = \sin(\pi u/2)$ for $u \in (0, 1)$ we have that

$$\sin(\pi u/2) < \text{sn}(u\mathcal{K}(k)|k) < 1 \qquad u \in (0, 1),\ k \in (0, 1).$$

Now, for each q we have that

$$\frac{w_q}{t_q} = \frac{\widetilde{w}_q}{\sigma_q^2} = \left(\frac{-2\sqrt{\lambda_{\min}}}{\pi Q \lambda_{\min}} \right) \frac{\mathcal{K}'(k)\text{cn}\left(iu_q\mathcal{K}'(k) \mid k\right) \text{dn}\left(iu_q\mathcal{K}'(k) \mid k\right)}{\text{sn}(iu_q\mathcal{K}'(k) \mid k)^2}$$
$$= \left(\frac{2\mathcal{K}'(k)}{\pi Q \lambda_{\min}} \right) \frac{\text{dn}\left(u_q\mathcal{K}(k') \mid k'\right)}{\text{sn}(u_q\mathcal{K}(k') \mid k')^2} \qquad \text{(via Jacobi imaginary transforms [e.g. 1])}$$

30

Consequently, we may conclude that

$$\frac{|w_q|}{|t_q|} = \left(\frac{2\mathcal{K}'(k)}{\pi Q \lambda_{\min}} \right) \frac{\mathrm{dn}\left(u_q \mathcal{K}(k') \mid k' \right)}{\mathrm{sn}(u_q \mathcal{K}(k') \mid k')^2}$$

$$\leq \frac{2 \log(5/k)}{\pi Q \lambda_{\min}} \left(\frac{1}{\sin^2(\pi u_q/2)} \right)$$

where we note that all quantities on the right hand side are positive. Plugging in the values of $k = 1/\sqrt{\kappa(\mathbf{K})}$, $u_q = (q - 1/2)/Q$ and summing over u_q we see that

$$\sum_{q=1}^{Q} \frac{|w_q|}{|t_q|} < \sum_{q=1}^{Q} \frac{2 \log \left(5\sqrt{\kappa(\mathbf{K})} \right)}{\pi Q \sqrt{\lambda_{\min}} \sin^2 \left(\frac{\pi(q-1/2)}{2Q} \right)}. \tag{S31}$$

Through trigonometric identities $\sum_{q=1}^{Q} 1/(Q \sin^2 \frac{\pi(q-1/2)}{2Q}) = 2Q$ and, therefore,

$$\sum_{q=1}^{Q} \frac{|w_q|}{|t_q|} < \frac{4Q \log \left(5\sqrt{\kappa(\mathbf{K})} \right)}{\pi \sqrt{\lambda_{\min}}}.$$

$\square$

With these lemmas we are now able to prove Theorem 1:

Theorem 1 (Restated). *Let $\mathbf{K} \succ 0$ and $\mathbf{b}$ be inputs to msMINRES-CIQ, producing $\mathbf{a}_J \approx \mathbf{K}^{1/2}\mathbf{b}$ after J iterations with Q quadrature points. The difference between $\mathbf{a}_J$ and $\mathbf{K}^{1/2}\mathbf{b}$ is bounded by:*

$$\left\| \mathbf{v}_J - \mathbf{K}^{\frac{1}{2}}\mathbf{b} \right\|_2 \leq \overbrace{\mathcal{O}\left(\exp\left(-\frac{2Q\pi^2}{\log \kappa(\mathbf{K}) + 3} \right) \right)}^{\text{Quadrature error}} + \overbrace{\frac{2Q \log \left(5\sqrt{\kappa(\mathbf{K})} \right) \kappa(\mathbf{K}) \sqrt{\lambda_{min}}}{\pi} \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2}^{\text{msMINRES error}}.$$

where $\lambda_{max}, \lambda_{min}$ are the max and min eigenvalues of $\mathbf{K}$, and $\kappa(\mathbf{K})$ is the condition number of $\mathbf{K}$.

Proof. First we note that the msMINRES-CIQ solution $\mathbf{a}_J$ can be written as $\sum_{i=1} w_q \mathbf{c}_J^{(q)}$, where $\mathbf{c}_J^{(q)}$ is the qth shifted solve $\approx (t_q \mathbf{I} + \mathbf{K})^{-1}\mathbf{b}$ from msMINRES. Applying the triangle inequality we have:

$$\left\| \mathbf{a}_J - \mathbf{K}^{\frac{1}{2}}\mathbf{b} \right\|_2 = \left\| \overbrace{\sum_{q=1}^{Q} w_q \mathbf{c}_J^{(q)} - \left(\mathbf{K} \sum_{q=1}^{Q} w_q (t_q \mathbf{I} + \mathbf{K})^{-1} \right) \mathbf{b}}^{\text{msMINRES error}} \right.$$

$$\left. + \underbrace{\left(\mathbf{K} \sum_{q=1}^{Q} w_q (t_q \mathbf{I} + \mathbf{K})^{-1} \right) \mathbf{b} - \mathbf{K}^{\frac{1}{2}}\mathbf{b}}_{\text{Quadrature error}} \right\|_2$$

$$\leq \sum_{q=1}^{Q} |w_q| \left\| \mathbf{c}_J^{(q)} - \mathbf{K} (t_q \mathbf{I} + \mathbf{K})^{-1} \mathbf{b} \right\|_2$$

$$+ \left\| \mathbf{K} \left(\sum_{q=1}^{Q} w_q (t_q \mathbf{I} + \mathbf{K})^{-1} \right) \mathbf{b} - \mathbf{K}^{\frac{1}{2}}\mathbf{b} \right\|_2 \tag{S32}$$

31

Plugging Lemma 2 into the msMINRES part of the bound bound, we have:

$$\sum_{q=1}^{Q} |w_q| \left(\frac{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} - 1}{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} + 1} \right)^{J} \|\mathbf{b}\|_2$$

$$\leq \sum_{q=1}^{Q} |w_q| \left(\frac{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} - 1}{\sqrt{\kappa(\mathbf{K} + t_q \mathbf{I})} + 1} \right) \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2 \qquad \text{(via Lemma 2)}$$

$$\leq \sum_{q=1}^{Q} |w_q| \left(\frac{\lambda_{\max}}{4 t_q} \right) \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2 \qquad \text{(via Lemma 3)}$$

$$\leq \frac{2Q \log\left(5\sqrt{\kappa(\mathbf{K})}\right) \lambda_{\max}}{\pi \sqrt{\lambda_{\min}}} \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2 \qquad \text{(via Lemma 4)}$$

$$\leq \frac{2Q \log\left(5\sqrt{\kappa(\mathbf{K})}\right) \sqrt{\lambda_{\min}} \kappa(\mathbf{K})}{\pi} \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2 .$$

Plugging this bound and Lemma 1 into Eq. (S32) completes the proof. $\qquad \square$

We can also prove this simple corollary:

Corollary 1. *Let $\mathbf{K} \succ 0$ and $\mathbf{b}$ be the inputs to Alg. 1, producing the output $\mathbf{a}'_J \approx \mathbf{K}^{-1/2}\mathbf{b}$ after J iterations with Q quadrature points. The difference between $\mathbf{a}_J$ and $\mathbf{K}^{1/2}\mathbf{b}$ is bounded by:*

$$\left\| \mathbf{a}'_J - \mathbf{K}^{-\frac{1}{2}}\mathbf{b} \right\|_2 \leq \overbrace{\mathcal{O}\left(\frac{1}{\lambda_{min}} \exp\left(-\frac{2Q\pi^2}{\log \kappa(\mathbf{K}) + 3} \right) \right)}^{\text{Quadrature error}} + \overbrace{\frac{2Q \log\left(5\sqrt{\kappa(\mathbf{K})}\right) \kappa(\mathbf{K})}{\sqrt{\lambda_{min}}\pi} \left(\frac{\sqrt{\kappa(\mathbf{K})} - 1}{\sqrt{\kappa(\mathbf{K})} + 1} \right)^{J-1} \|\mathbf{b}\|_2}^{\text{msMINRES error}} .$$

where $\lambda_{max}, \lambda_{min}$ are the maximal and minimal eigenvalues of $\mathbf{K}$, and $\kappa(\mathbf{K})$ is the condition number of $\mathbf{K}$.

Proof. Note that $\mathbf{a}'_J = \mathbf{K}^{-1}\mathbf{a}_J$, where $\mathbf{a}_J$ is the msMINRES-CIQ estimate of $\mathbf{K}^{1/2}\mathbf{b}$. Using the sub-multiplicative property of the induced matrix 2-norm we see that

$$\left\| \mathbf{a}'_J - \mathbf{K}^{-\frac{1}{2}}\mathbf{b} \right\|_2 \leq \left\| \mathbf{K}^{-1} \right\|_2 \left\| \mathbf{a}_J - \mathbf{K}^{\frac{1}{2}}\mathbf{b} \right\|_2 = \frac{1}{\lambda_{\min}} \left\| \mathbf{a}_J - \mathbf{K}^{\frac{1}{2}}\mathbf{b} \right\|_2 ,$$

where the final term is bounded by Thm. 1. $\qquad \square$