MULTISCALE MODEL. SIMUL.
Vol. 15, No. 4, pp. 1584–1611

FAST SPATIAL GAUSSIAN PROCESS MAXIMUM LIKELIHOOD ESTIMATION VIA SKELETONIZATION FACTORIZATIONS*

VICTOR MINDEN†, ANIL DAMLE‡, KENNETH L. HO§, AND LEXING YING¶

Abstract. Maximum likelihood estimation for parameter fitting given observations from a Gaussian process in space is a computationally demanding task that restricts the use of such methods to moderately sized datasets. We present a framework for unstructured observations in two spatial dimensions that allows for evaluation of the log-likelihood and its gradient (i.e., the score equations) in $\tilde{O}(n^{3/2})$ time under certain assumptions, where n is the number of observations. Our method relies on the skeletonization procedure described by Martinsson and Rokhlin [*J. Comput. Phys.*, 205 (2005), pp. 1–23] in the form of the recursive skeletonization factorization of Ho and Ying [*Comm. Pure Appl. Math.*, 69 (2015), pp. 1415–1451]. Combining this with an adaptation of the matrix peeling algorithm of Lin, Lu, and Ying [*J. Comput. Phys.*, 230 (2011), pp, 4071–4087] for constructing $\mathcal{H}$-matrix representations of black-box operators, we obtain a framework that can be used in the context of any first-order optimization routine to quickly and accurately compute maximum likelihood estimates.

Key words. spatial Gaussian processes, kriging, hierarchical matrices, maximum likelihood estimation, fast algorithms

AMS subject classifications. 65F30, 65C60, 60G15

DOI. 10.1137/17M1116477

1. Introduction.

Gaussian processes are commonly used in the applied sciences as a statistical model for spatially indexed observations. In such applications, each observation $z_i \in \mathbb{R}$ of some quantity is associated with a corresponding location $x_i \in \Omega \subset \mathbb{R}^d$ with $d = 2$ or 3. Given a prescribed covariance kernel $K(\cdot, \cdot; \theta)$ that maps $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$ and is specified up to some parameter vector $\theta \in \mathbb{R}^p$, any vector of observations $z = [z_1, \ldots, z_n] \in \mathbb{R}^n$ (with associated locations $\{x_i\}_{i=1}^n$) is assumed to be randomly distributed as a multivariate Gaussian

$$z \sim N(0, \Sigma(\theta)) \text{ with covariances } [\Sigma(\theta)]_{ij} = K(x_i, x_j; \theta). \tag{1}$$

We assume the mean of the process to be 0 (i.e., known) for simplicity, though, as we discuss later, this is not strictly necessary.

*Received by the editors February 13, 2017; accepted for publication (in revised form) September 6, 2017; published electronically November 7, 2017.
http://www.siam.org/journals/mms/15-4/M111647.html

Funding: The first author was supported by a Stanford Graduate Fellowship in Science & Engineering and a U.S. Department of Energy Computational Science Graduate Fellowship (grant DE-FG02-97ER25308). The second author was supported by a Simons Graduate Research Assistantship and National Science Foundation Graduate Research Fellowship (grant DGE-1147470). The third author was supported by a National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship (grant DMS-1203554). The fourth author was supported by the National Science Foundation (grants DMS-1328230 and DMS-1521830) and U.S. Department of Energy Advanced Scientific Computing Research program (grants DE-FC02-13ER26134 and DE-SC0009409).

†Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (vminden@stanford.edu).

‡Department of Computer Science, Cornell University, Ithaca, NY 14850 (damle@cornell.edu).

§TSMC Technology Inc., San Jose, CA 95134 (klho@alumni.caltech.edu).

¶Department of Mathematics and Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (lexing@math.stanford.edu).

Neglecting θ, common choices for the covariance kernel include the family of rational quadratic kernels

$$K(x,y) = \left(1 + \frac{\|x-y\|^2}{2\alpha}\right)^{-\alpha}, \tag{2}$$

which has corresponding processes that are infinitely differentiable in the mean-squared sense for all $\alpha > 0$. Notably, this family includes the Gaussian or squared exponential kernel as a limiting case as $\alpha \to \infty$. Another popular family is the Matérn family of kernels

$$K(x,y) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu} \cdot \|x-y\|\right)^{\nu} K_\nu \left(\sqrt{2\nu} \cdot \|x-y\|\right), \tag{3}$$

where $K_\nu(\cdot)$ is the modified second-kind Bessel function of order ν, $\Gamma(\cdot)$ is the gamma function, and the corresponding process is $\lfloor \nu \rfloor$ times mean-squared differentiable. To explicitly parameterize $K(\cdot,\cdot;\theta)$, we might introduce a correlation length parameter ρ leading to

$$K(x,y;\theta) = \left(1 + \frac{\|x-y\|^2}{2\alpha\rho^2}\right)^{-\alpha}$$

in the case of the rational quadratic kernel. The fundamental parameters of the kernel family, e.g., α or ν, may be considered as part of θ or fixed a priori.

Typically, the parameter vector θ is unknown and must be estimated from the data. For example, given a parameterized family of kernels and a set of observations, we might want to infer θ for later use in estimating the value of the field at other spatial locations as in kriging (see Stein [38]). In this paper we consider the general Gaussian process maximum likelihood estimation (MLE) problem for θ: given an observation vector $z \in \mathbb{R}^n$, find $\hat{\theta}_{\text{MLE}}$ maximizing the Gaussian process log-likelihood

$$\ell(\theta) \equiv -\frac{1}{2} z^T \Sigma^{-1} z - \frac{1}{2} \log|\Sigma| - \frac{n}{2} \log 2\pi, \tag{4}$$

where we have dropped the explicit dependence of Σ on θ for notational convenience.

If θ is unconstrained, then $\hat{\theta}_{\text{MLE}}$ is given by maximizing (4) over all of $\mathbb{R}^p$. In this case, it is possible under certain assumptions to perform MLE by solving the score equations $g(\theta) = 0$, where the gradient of the log-likelihood $g(\theta) \in \mathbb{R}^p$ is given componentwise by

$$g_i \equiv \frac{\partial \ell(\theta)}{\partial \theta_i} = \frac{1}{2} z^T \Sigma^{-1} \Sigma_i \Sigma^{-1} z - \frac{1}{2} \operatorname{Tr}(\Sigma^{-1}\Sigma_i), \quad i = 1, \ldots, p, \tag{5}$$

where $\Sigma_i \equiv \frac{\partial}{\partial \theta_i}\Sigma$. This may be accomplished without evaluating $\ell(\theta)$ as is done by, e.g., Anitescu, Chen, and Wang [2] and Stein, Chen, and Anitescu [40]. In contrast, we consider in this paper the use of first-order methods for nonlinear optimization that, at each iteration, use both gradient and log-likelihood evaluations to find a local optimum. This allows for the treatment of constraints if desired, though we note the methods here are equally applicable to the unconstrained case.

For any given θ, both $\ell(\theta)$ and the gradient $g(\theta)$ contain a number of terms whose evaluation is traditionally computationally expensive. For example, the Cholesky decomposition of Σ may be used to calculate $z^T\Sigma^{-1}z$ and $z^T\Sigma^{-1}\Sigma_i\Sigma^{-1}z$ as well as the log-determinant $\log|\Sigma|$ and trace $\operatorname{Tr}(\Sigma^{-1}\Sigma_i)$, but the asymptotic computation and storage complexities are $O(n^3)$ and $O(n^2)$, respectively. This is prohibitively expensive for datasets with a large number of observations, necessitating alternative approaches.

1.1. Our method. The contribution of this paper is a framework for efficiently finding $\hat{\theta}_{\text{MLE}}$ by taking advantage of fast hierarchical matrix algorithms developed in the numerical linear algebra community. Such algorithms exploit the fact that linear operators defined in terms of pairwise kernel evaluations between points embedded in $\mathbb{R}^d$ frequently exhibit *hierarchical rank structure*, as we discuss in section 2 (briefly, many different-sized off-diagonal blocks of the matrix are close to low rank and thus compressible). Our framework has two key parts:

(I) Construct a fast approximate hierarchical factorization of the covariance matrix Σ and its derivatives Σ_i for $i = 1, \ldots, p$ in a matrix-free fashion using the kernel function $K(\cdot, \cdot; \theta)$ and the points $\{x_i\}_{i=1}^n$.

(II) Additionally factor the derivatives of the covariance matrix Σ_i for $i = 1, \ldots, p$ through the same approach as in (I). Use the hierarchical factorizations as a fast black-box operator for approximately applying $\Sigma^{-1}\Sigma_i$ for $i = 1, \ldots, p$, and use this operator and the points $\{x_i\}_{i=1}^n$ to compute the traces $\text{Tr}(\Sigma^{-1}\Sigma_i)$ through a randomized "matrix peeling" scheme for efficiently extracting the trace of a hierarchically rank-structured linear operator.

In both parts, the approximation accuracy is well-controlled by specified tolerance parameters intrinsic to the algorithms.

For any θ, the approximate factorization of part (I) can be used to efficiently evaluate the terms composing $\ell(\theta)$ (including the log-determinant), which overlaps with recent work by Ambikasaran et al. [1] that addresses the use of hierarchical factorizations for kernelized covariance matrices for computing these terms (see also Khoromskij, Litvinenko, and Matthies [28] and Börm and Garcke [4] for earlier work on $\mathcal{H}$-matrix techniques for fast computation of matrix-vector products with Σ in a Gaussian process context). This piece of the framework alone gives sufficient machinery to perform black-box optimization using numerical derivatives. However, using finite differences of an approximate log-likelihood can magnify approximation errors, which can lead to larger inaccuracies in the approximate gradient depending on, e.g., the conditioning of Σ (see subsection 5.4 for a relatively benign example). Therefore, central to our framework is the computation in (II) of the gradient components g_i for $i = 1, \ldots, p$, which requires the trace terms $\text{Tr}(\Sigma^{-1}\Sigma_i)$.

In the simplest form detailed in this paper, our framework employs the *recursive skeletonization factorization* [25] as the approximate hierarchical factorization variant of choice for Σ and Σ_i, $i = 1, \ldots, p$. We then compute the trace terms in the gradient using an adaptation of the matrix peeling algorithm of Lin, Lu, and Ying [29]. Combining these two tools, we obtain an efficient method for evaluating (4) and (5)—and, ultimately, finding $\hat{\theta}_{\text{MLE}}$—with high and controllable accuracy using a black-box first-order optimization package (e.g., `fminunc` or `fmincon` in MATLAB).

While the framework of this paper technically applies to observations in d dimensions for general d, the computational complexity increases in high dimensions due to how the runtime of hierarchically rank-structured factorizations depends on the numerical rank of off-diagonal blocks. For example, applying these methods in the case $d = 1$ is essentially optimal in the sense that off-diagonal matrix blocks have numerical rank that is not strongly dependent on the number of observations. We direct the reader to Ambikasaran et al. [1] for extensive numerical examples of factoring kernel matrices in this case. In contrast, for $d = 3$ the observed rank growth is in general much larger and leads to greater asymptotic complexities. We focus on the case $d = 2$ in the remainder of this paper, but note the broader applicability.

1.2. Alternative approaches. Due to the prevalance of Gaussian process models, a number of methods exist in the literature for fast computations involving kernelized covariance matrices. To decrease apply, solve, and storage costs, Σ can be replaced with a sparser "tapered" approximant as described by Furrer, Genton, and Nychka [12], wherein the desired covariance kernel is multiplied pointwise with a compactly supported tapering function to attain sparsity. Of course, the computational benefit of tapering depends on the sparsity of the resulting approximant, which is limited by the desired accuracy if the correlation length of the kernel is not small.

If Σ decomposes naturally into the sum of a diagonal and a numerically low-rank matrix then such a decomposition can be quite efficient for computation (see Cressie and Johannesson [10]), but this representation is too simple for the applications and kernel functions we consider. Extending this to a general sparse-plus-low-rank model by replacing the diagonal piece with a tapered covariance kernel can perform better than either a tapered or low-rank Σ alone, as shown by Sang and Huang [36] (see also Vanhatalo, Pietiläinen, and Vehtari [43]).

For cases where the underlying process is stationary and the observations lie on a regular grid it is possible to directly approximate the log-likelihood using spectral approximations due to Whittle [45] or to quickly apply the covariance matrix in Fourier space to solve linear systems with an iterative method. Further, in such cases these systems can be preconditioned using the method of Stein, Chen, and Anitescu [39] yielding efficient methods for many important problem classes. For irregularly spaced data such as we consider in this paper, however, these approaches do not apply directly. One approach for log-likelihood evaluation (and thus derivative-free optimization) with generally distributed data is that of Aune, Simpson, and Eidsvik [3], which offers an involved framework for approximating the log-determinant using Krylov methods, assuming the covariance matrix or its inverse can be efficiently applied. More recently, Castrillón-Candás, Genton, and Yokota [5] demonstrated a combination of multilevel preconditioning and tapering for fast derivative-free restricted MLE, though gradient computation is not discussed.

An alternative approach to approximating the Gaussian process log-likelihood directly is to explicitly construct and solve a different set of estimating equations that is less computationally cumbersome. For example, the Hutchinson-like sample average approximation estimator [2, 40] falls into this category, as do the composite likelihood methods described by, e.g., Vecchia [44] and Stein, Chi, and Welty [41] and their extension, block composite likelihood methods (see, e.g., Eidsvik et al. [11]). Another notable effort based on a modified model is the work of Lindgren, Rue, and Lindström [30], which gives a way of approximating Gaussian processes by Gaussian Markov random fields for specific kernels in the Matérn family. In practice, these methods perform quite well, but in our approach, we consider maximizing the true likelihood as opposed to alternative models or estimators.

1.3. Outline. The remainder of the paper is organized as follows. In section 2 we review hierarchical matrix structure and outline the recursive skeletonization factorization [25, section 3], which is our hierarchical matrix format of choice for fast MLE. In section 3, we discuss a modification of the matrix peeling algorithm by Lin, Lu, and Ying [29], which we use to quickly evaluate the gradient of the log-likelihood. In section 4, we succinctly summarize our framework. In section 5 we present numerical results on a number of test problems and demonstrate the scaling of our approach. Finally, in section 6, we make some concluding remarks.

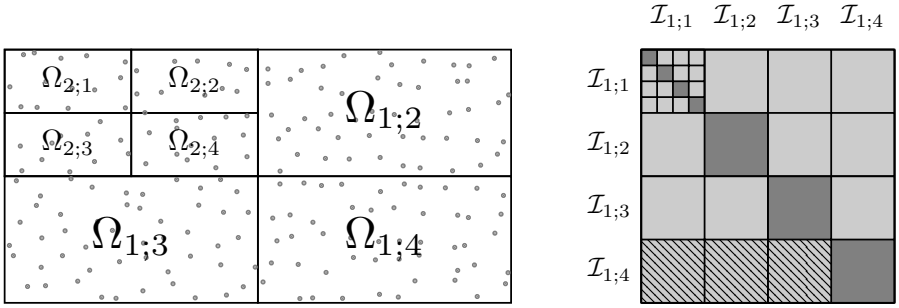

FIG. 1. *To expose low-rank structure in the matrix Σ, consider the partitions $\Omega = \cup_{i=1}^{4}\Omega_{1;i}$ and $\Omega_{1;1} = \cup_{i=1}^{4}\Omega_{2;i}$ (left), with corresponding index sets $\mathcal{I}_{\ell;i}$ for $\ell = 1, \ldots, 2$, $i = 1, \ldots, 4$. Due to smoothness properties of $K(\cdot, \cdot; \theta)$, the blocks of Σ are rank structured (right), where dark gray blocks on the diagonal are full rank and light gray off-diagonal blocks are numerically low rank. Representing each off-diagonal block in low-rank form independently leads to the HODLR format. In contrast, the HBS format uses one low-rank representation for all off-diagonal blocks for each block row, e.g., the patterned blocks in the bottom row are aggregated into a single low-rank matrix.*

2. Factorization of the covariance matrix. Consider the kernelized covariance matrix Σ as in (1), and assume for simplicity of exposition that the points $\{x_i\}_{i=1}^{n}$ are uniformly distributed inside a rectangular domain Ω. Partitioning Ω into four equal rectangular subdomains $\Omega_{1;i}$ for $i = 1, \ldots, 4$, it has been observed that the corresponding block partitioning of Σ exposes low-rank structure of off-diagonal blocks when $K(x, y; \theta)$ is sufficiently nice as a function of $\|x - y\|$.

Concretely, we define the set $[n] \equiv \{1, 2, \ldots, n\}$ and let $\mathcal{I}_{1;i} \subset [n]$ denote the index set indexing degrees of freedom (DOFs) located inside $\Omega_{1;i}$ for $i = 1, \ldots, 4$ such that $\{x_j\}_{j \in \mathcal{I}_{1;i}} \subset \Omega_{1;i}$ and $\bigcup_{i=1}^{4} \mathcal{I}_{1;i} = [n]$. In a self-similar fashion, we further partition $\Omega_{1;1}$ into the four subdomains $\Omega_{2;i}$ (with corresponding DOFs $\mathcal{I}_{2;i}$) for $i = 1, \ldots, 4$ and obtain the decomposition shown in Figure 1 (left). Assuming that the covariance kernel is smooth away from $x = y$ and does not exhibit high-frequency oscillations, the off-diagonal blocks $\Sigma(\mathcal{I}_{1;i}, \mathcal{I}_{1;j})$ for $i \neq j$ and $\Sigma(\mathcal{I}_{2;i}, \mathcal{I}_{2;j})$ for $i \neq j$ in the corresponding partitioning shown in the same figure (right) tend to be *numerically low rank* and thus compressible.

DEFINITION 1 (numerically low rank). *We call a matrix $\mathsf{A} \in \mathbb{R}^{m_1 \times m_2}$* numerically low rank *with respect to a specified tolerance ϵ if for some $r < \min(m_1, m_2)$ there exist matrices $\mathsf{V}_1 \in \mathbb{R}^{m_1 \times r}$ and $\mathsf{V}_2 \in \mathbb{R}^{m_2 \times r}$ such that $\|\mathsf{A} - \mathsf{V}_1\mathsf{V}_2^T\|_2 \leq \epsilon \|A\|_2$.*

The rank structure of Figure 1 includes numerically low-rank blocks at multiple spatial scales *independent of the length-scale of the underlying Gaussian process*, i.e., we may continue to recursively subdivide the domain and expose more compressible blocks of Σ. Explicitly representing each of these blocks in low-rank form leads to the so-called hierarchical off-diagonal low-rank (HODLR) matrix format that has been used by Ambikasaran et al. [1] to compress various families of covariance kernels with application to Gaussian processes.

Remark 1. A sufficient condition to ensure this rank structure is that for any pair of distinct subdomains on the same level $\Omega_{\ell;i}$ and $\Omega_{\ell;j}$ there exists an approximation $K(x_i, x_j; \theta) \approx \sum_{k=1}^{r} f_k(x_i) g_k(x_j)$ for any $x_i \in \mathcal{I}_{\ell;i}$ and $x_j \in \mathcal{I}_{\ell;j}$, where the number of terms r is relatively small (via, e.g., Chebyshev polynomials). An important example where this is not typically the case is periodic kernels with a short period relative to the size of the domain. Further, the techniques we discuss here are less relevant to

compactly supported kernels, for which standard sparse linear algebra is already a computationally efficient approach.

The HODLR format is only one of many hierarchical matrix formats, appearing as a special case of the $\mathcal{H}$- and $\mathcal{H}^2$-matrices of Hackbusch and collaborators [17, 20, 19]. This format is particularly simple as at each level it compresses *all off-diagonal blocks* of Σ, including those corresponding to domains that are adjacent (e.g., $\Omega_{1;2}$ and $\Omega_{1;3}$). Matrices compressible in this way are referred to as *weakly admissible* [21], in contrast to *strongly admissible* matrices which compress only a subset of off-diagonal blocks at each level. Closely related literature includes work on hierarchically semiseparable matrices [47, 7, 6] and hierarchically block separable (HBS) matrices [31, 13] which offer simplified representations for weakly admissible matrices with improved runtime.

For matrices where the entries are explicitly generated by an underlying kernel function such as Σ and its derivatives, specific factorization algorithms have been developed to exploit this additional structure for increased efficiency [15, 13, 24, 25, 9]. These "skeletonization-based" algorithms, based on the framework introduced by Martinsson and Rokhlin [31] stemming from observations by Starr and Rokhlin [37] and Greengard and Rokhlin [16], construct low-rank representations of certain off-diagonal blocks using the skeletonization process described by Cheng et al. [8].

Our framework is agnostic to the choice of hierarchical factorization used for Σ and its derivatives, provided that the factorization admits fast linear algebra computations (including computation of the log-determinant) with the underlying operator. The recursive skeletonization factorization that we use in this paper was first introduced by Ho and Ying [25, section 3] as a multiplicative factorization based on skeletonization [31]. In the remainder of this section we provide a brief review of the algorithm.

Remark 2. In what follows, we will continue to assume that the DOFs $\{x_i\}_{i=1}^n$ are uniformly distributed inside a rectangular subdomain Ω for simplicity of exposition. Further, we will describe the algorithm as though the quadtree representing the hierarchical partitioning of space is *perfect*, i.e., every subdomain is subdivided into four child subdomains at every level. In practice an adaptive decomposition of space is used to avoid subdividing the domain in regions of low observation density.

2.1. Block compression through skeletonization. We begin by recursively subdividing Ω into four subdomains until each leaf-level subdomain contains a constant number of observations independent of n. This leads to a quadtree data structure with levels labeled $\ell = 0$ through $\ell = L$, where $\ell = 0$ refers to the entire domain Ω and $\ell = L$ refers to the collection of subdomains $\Omega_{L;i}$ for $i = 1, \ldots, 4^L$. Considering factorization of the covariance matrix Σ, the basic intuition of the method is to first compress all blocks of Σ corresponding to covariances between observations in distinct subdomains at the leaf level, and then to recurse in a bottom-up traversal.

Consider first a single leaf-level subdomain containing observations indexed by $\mathcal{I} \subset [n]$ and define the complement DOF set $\mathcal{I}^c \equiv [n] \setminus \mathcal{I}$. Given a specified tolerance $\epsilon > 0$, the algorithm proceeds by compressing the off-diagonal blocks $\Sigma(\mathcal{I}, \mathcal{I}^c)$ and $\Sigma(\mathcal{I}^c, \mathcal{I})$ as in the HBS format (see Figure 1) through the use of an *interpolative decomposition* (ID) [8].

DEFINITION 2 (interpolative decomposition). *Given a matrix* $\mathsf{A} \in \mathbb{R}^{m \times |\mathcal{I}|}$ *with columns indexed by* $\mathcal{I}$ *and a tolerance* $\epsilon > 0$*, an* ϵ*-accurate ID of* A *is a partitioning of* $\mathcal{I}$ *into DOF sets associated with so-called skeleton columns* $\mathcal{S} \subset \mathcal{I}$ *and redundant columns* $\mathcal{R} = \mathcal{I} \setminus \mathcal{S}$ *and a corresponding interpolation matrix* $\mathsf{T}_{\mathcal{I}}$ *such that*

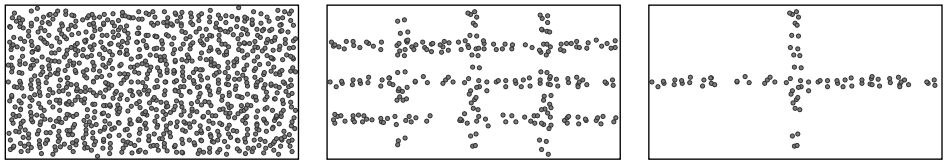

FIG. 2. *Shown here are the DOFs to be compressed at each level of the recursive skeletonization factorization. At the leaf level (left), all DOFs are involved in skeletonization. At each subsequent level (center, right), only skeleton DOFs from the previous level are involved in further skeletonization. We see that the skeleton DOFs tend to line the boundaries of their corresponding subdomains.*

$\|\mathsf{A}(:,\mathcal{R}) - \mathsf{A}(:,\mathcal{S})\mathsf{T}_{\mathcal{I}}\|_2 \leq \epsilon\|\mathsf{A}\|_2$, *where* $\mathsf{A}(:,\mathcal{R}) \in \mathbb{R}^{m\times|\mathcal{R}|}$ *is given by subselecting the columns of* A *indexed by* $\mathcal{R}$, *and* $\mathsf{A}(:,\mathcal{S})$ *is defined analogously.*

It is desirable in Definition 2 to take $|\mathcal{S}|$ as small as possible for a given ϵ.

Given an ID of $\Sigma(\mathcal{I}^c,\mathcal{I})$ such that $\Sigma(\mathcal{I}^c,\mathcal{R}) \approx \Sigma(\mathcal{I}^c,\mathcal{S})\mathsf{T}_{\mathcal{I}}$, Σ can be written in block form (up to a permutation) as

$$\left[\begin{array}{cc|c} \Sigma(\mathcal{I}^c,\mathcal{I}^c) & \Sigma(\mathcal{I}^c,\mathcal{S}) & \Sigma(\mathcal{I}^c,\mathcal{R}) \\ \Sigma(\mathcal{S},\mathcal{I}^c) & \Sigma(\mathcal{S},\mathcal{S}) & \Sigma(\mathcal{S},\mathcal{R}) \\ \hline \Sigma(\mathcal{R},\mathcal{I}^c) & \Sigma(\mathcal{R},\mathcal{S}) & \Sigma(\mathcal{R},\mathcal{R}) \end{array}\right] \approx \left[\begin{array}{cc|c} \Sigma(\mathcal{I}^c,\mathcal{I}^c) & \Sigma(\mathcal{I}^c,\mathcal{S}) & \Sigma(\mathcal{I}^c,\mathcal{S})\mathsf{T}_{\mathcal{I}} \\ \Sigma(\mathcal{S},\mathcal{I}^c) & \Sigma(\mathcal{S},\mathcal{S}) & \Sigma(\mathcal{S},\mathcal{R}) \\ \hline \mathsf{T}_{\mathcal{I}}^T\Sigma(\mathcal{S},\mathcal{I}^c) & \Sigma(\mathcal{R},\mathcal{S}) & \Sigma(\mathcal{R},\mathcal{R}) \end{array}\right].$$

Using a sequence of block row and column operations, we first eliminate the blocks $\Sigma(\mathcal{I}^c,\mathcal{S})\mathsf{T}_{\mathcal{I}}$ and $\mathsf{T}_{\mathcal{I}}^T\Sigma(\mathcal{S},\mathcal{I}^c)$ and then decouple the bottom-right block to obtain

$$\mathsf{L}_{\mathcal{I}}^{-1}\Sigma\mathsf{L}_{\mathcal{I}}^{-T} \approx \left[\begin{array}{cc|c} \Sigma(\mathcal{I}^c,\mathcal{I}^c) & \Sigma(\mathcal{I}^c,\mathcal{S}) & \\ \Sigma(\mathcal{S},\mathcal{I}^c) & \Sigma(\mathcal{S},\mathcal{S}) & \mathsf{X}_{\mathcal{S}\mathcal{R}} \\ \hline & \mathsf{X}_{\mathcal{R}\mathcal{S}} & \mathsf{X}_{\mathcal{R}\mathcal{R}} \end{array}\right] = \mathsf{U}_{\mathcal{I}} \left[\begin{array}{cc|c} \Sigma(\mathcal{I}^c,\mathcal{I}^c) & \Sigma(\mathcal{I}^c,\mathcal{S}) & \\ \Sigma(\mathcal{S},\mathcal{I}^c) & \mathsf{X}_{\mathcal{S}\mathcal{S}} & \\ \hline & & \mathsf{X}_{\mathcal{R}\mathcal{R}} \end{array}\right] \mathsf{U}_{\mathcal{I}}^T,$$

where $\mathsf{L}_{\mathcal{I}}$ and $\mathsf{U}_{\mathcal{I}}$ are block unit-triangular matrices that are fast to apply or invert and the X subblocks are linear combinations of the Σ subblocks.

2.2. The recursive skeletonization factorization. Defining the collection of DOF sets corresponding to subdomains at level $\ell = L$ as $\mathscr{L}_L \equiv \{\mathcal{I}_{L;1},\mathcal{I}_{L;2},\ldots,\mathcal{I}_{L;4^L}\}$, we use the skeletonization process of subsection 2.1 to compress the corresponding blocks of Σ for each $\mathcal{I} \in \mathscr{L}_L$, yielding $\Sigma \approx \left(\prod_{\mathcal{I}\in\mathscr{L}_L} \mathsf{L}_{\mathcal{I}}\mathsf{U}_{\mathcal{I}}\right) \tilde{\Sigma}_L \left(\prod_{\mathcal{I}\in\mathscr{L}_L} \mathsf{U}_{\mathcal{I}}^T\mathsf{L}_{\mathcal{I}}^T\right)$, where the order taken in the product over $\mathscr{L}_L$ does not matter due to the structure of the L and U matrices. Using $\mathcal{R}_{L;i}$ and $\mathcal{S}_{L;i}$ to denote the redundant DOFs and skeleton DOFs associated with $\mathcal{I}_{L;i}$ for each $i = 1,\ldots,4^L$ and defining $\mathcal{R}_L \equiv \cup_{i=1}^{4^L}\mathcal{R}_{L;i}$, the blocks of the matrix $\tilde{\Sigma}_L$ have the following structure for each $i = 1,\ldots,4^L$:

- The modified block $\tilde{\Sigma}_L(\mathcal{R}_{L;i},\mathcal{R}_{L;i})$ has been decoupled from the rest of $\tilde{\Sigma}_L$.
- The block $\tilde{\Sigma}_L(\mathcal{S}_{L;i},\mathcal{S}_{L;i})$ has been modified.
- The blocks $\tilde{\Sigma}_L(\mathcal{S}_{L;i},\mathcal{I}_{L;i}^c \setminus \mathcal{R}_L) = \Sigma(\mathcal{S}_{L;i},\mathcal{I}_{L;i}^c \setminus \mathcal{R}_L)$ and $\tilde{\Sigma}_L(\mathcal{I}_{L;i}^c \setminus \mathcal{R}_L,\mathcal{S}_{L;i}) = \Sigma(\mathcal{I}_{L;i}^c \setminus \mathcal{R}_L,\mathcal{S}_{L;i})$ remain unmodified from what they were in Σ.

In other words, we have identified and decoupled all redundant DOFs at the leaf level while leaving unchanged the blocks of Σ corresponding to kernel evaluations between skeleton DOFs in distinct leaf-level subdomains.

The recursive skeletonization factorization of Σ is given by repeating this process at each higher level of the quadtree as visualized in Figure 2. For example, at level $\ell = L-1$ of the quadtree, each subdomain contains skeleton DOFs corresponding to its

four distinct child subdomains in the tree. However, the redundant DOFs of its child subdomains no longer need to be considered as they have already been decoupled. We thus define $\tilde{\mathcal{I}}_{L-1;i} \equiv \mathcal{I}_{L-1;i} \setminus \mathcal{R}_L$ for each $i = 1, \ldots, 4^{(L-1)}$ and write the collection of DOFs remaining at this level as $\mathscr{L}_{L-1} \equiv \{\tilde{\mathcal{I}}_{L-1;1}, \tilde{\mathcal{I}}_{L-1;2}, \ldots, \tilde{\mathcal{I}}_{L-1;4^{(L-1)}}\}$. Due to the hierarchical block low-rank structure of Σ, off-diagonal blocks at this level are again compressible through skeletonization, yielding

$$\tilde{\Sigma}_L \approx \mathsf{P}_L \left(\prod_{\mathcal{I} \in \mathscr{L}_{L-1}} \mathsf{L}_{\mathcal{I}} \mathsf{U}_{\mathcal{I}} \right) \tilde{\Sigma}_{L-1} \left(\prod_{\mathcal{I} \in \mathscr{L}_{L-1}} \mathsf{U}_{\mathcal{I}}^T \mathsf{L}_{\mathcal{I}}^T \right) \mathsf{P}_L^T,$$

where P_L is a global permutation matrix regrouping the DOF sets in $\mathscr{L}_{L-1}$ to be contiguous. Proceeding in this fashion level by level and defining the ordered product $\prod_{\ell=1}^{L} \mathsf{A}_\ell \equiv \mathsf{A}_L \mathsf{A}_{L-1} \ldots \mathsf{A}_1$, we obtain the full recursive skeletonization factorization F of Σ

$$\begin{aligned} \Sigma &\approx \left[\prod_{\ell=1}^{L} \left(\prod_{\mathcal{I} \in \mathscr{L}_\ell} \mathsf{L}_{\mathcal{I}} \mathsf{U}_{\mathcal{I}} \right) \mathsf{P}_\ell \right] \tilde{\Sigma}_1 \left[\prod_{\ell=1}^{L} \left(\prod_{\mathcal{I} \in \mathscr{L}_\ell} \mathsf{L}_{\mathcal{I}} \mathsf{U}_{\mathcal{I}} \right) \mathsf{P}_\ell \right]^T \\ &= \left[\prod_{\ell=1}^{L} \left(\prod_{\mathcal{I} \in \mathscr{L}_\ell} \mathsf{L}_{\mathcal{I}} \mathsf{U}_{\mathcal{I}} \right) \mathsf{P}_\ell \right] \mathsf{C}\mathsf{C}^T \left[\prod_{\ell=1}^{L} \left(\prod_{\mathcal{I} \in \mathscr{L}_\ell} \mathsf{L}_{\mathcal{I}} \mathsf{U}_{\mathcal{I}} \right) \mathsf{P}_\ell \right]^T \equiv \mathsf{F}, \end{aligned} \tag{6}$$

where $\tilde{\Sigma}_1$ is block diagonal with diagonal blocks corresponding to the sets of redundant DOFs at each level and $\tilde{\Sigma}_1 = \mathsf{C}\mathsf{C}^T$ is the Cholesky decomposition of $\tilde{\Sigma}_1$.

Remark 3. Because the factorization $\mathsf{F} \approx \Sigma$ is approximate, using an extremely inaccurate tolerance ϵ in the IDs of Definition 2 admits the possibility that $\tilde{\Sigma}_1$ may be slightly indefinite due to approximation error. In practice, this is not an issue for any tolerance precise enough to be used for computing $\ell(\theta)$ for optimization purposes. When factoring the derivative matrices Σ_i for $i = 1, \ldots, p$ (which may themselves be indefinite), the Cholesky decomposition may be replaced with, e.g., an LDL^T factorization.

2.3. Computational complexity. The computational cost of the recursive skeletonization factorization is in theory dominated by the cost of computing IDs $\Sigma(\mathcal{I}^c, \mathcal{R}) \approx \Sigma(\mathcal{I}^c, \mathcal{S})\mathsf{T}_{\mathcal{I}}$ of $\Sigma(\mathcal{I}^c, \mathcal{I})$ in subsection 2.1 for each $\mathcal{I}$. This is because the typical algorithm to compute an ID is based on a rank-revealing QR factorization, such that the $m \times |\mathcal{I}|$ ID of Definition 2 has complexity $O(m|\mathcal{I}|^2)$ [8]. This dependence on m is prohibitively expensive during the initial levels of the factorization, since $m = |\mathcal{I}^c| = O(n)$ and there are $O(n)$ such IDs to compute.

The original application of skeletonization was to boundary integral equations arising from elliptic partial differential equations, in which case the so-called "proxy trick" described by Martinsson and Rokhlin [31] can be applied to accelerate the computation of an ID through the use of integral identities. These integral identities do not strictly apply in the case where $K(\cdot, \cdot; \theta)$ is a general covariance function, but we find that a variant of this proxy trick works well to obtain similar acceleration.

2.3.1. Modified proxy trick. Suppose that the index set $\mathcal{I}$ corresponds to points inside the subdomain $\mathrm{B} \subset \Omega$ in Figure 3, where we use B as a stand-in for an arbitrary subdomain $\Omega_{\ell;i}$ in our quadtree. The purpose of computing an ID of $\Sigma(\mathcal{I}^c, \mathcal{I})$ is to find a small set of skeleton DOFs $\mathcal{S} \subset \mathcal{I}$ such that the range of $\Sigma(\mathcal{I}^c, \mathcal{S})$

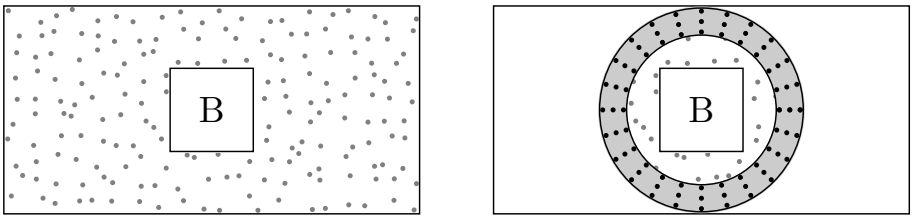

FIG. 3. *Because of the underlying kernel, computing an ID of the submatrix* $\Sigma(\mathcal{I}^c, \mathcal{I})$ *can be accelerated, where* $\mathcal{I} \subset [n]$ *indexes observations inside the subdomain B (left). Rather than considering all of* $\mathcal{I}^c$*, the (modified)* proxy trick *involves neglecting rows of* $\Sigma(\mathcal{I}^c, \mathcal{I})$ *corresponding to points* $\mathcal{F} \subset \mathcal{I}^c$ *far from B. Instead, the ID is computed by considering only points* $\mathcal{N} \subset \mathcal{I}^c$ *near B combined with a small number of so-called "proxy points" discretizing an annulus around B (right).*

approximately captures the range of $\Sigma(\mathcal{I}^c, \mathcal{I})$. The key to computational acceleration using the proxy trick is to accomplish this without using all rows of $\Sigma(\mathcal{I}^c, \mathcal{I})$ in the computation.

As detailed by Ho and Ying [25, section 3.3], we can partition the DOFs $\mathcal{I}^c$ into those that are near to B and those that are far from B, denoted $\mathcal{N}$ and $\mathcal{F}$, respectively. For example, we may take $\mathcal{N}$ to be all points $x_i \in \mathcal{I}^c$ such that the distance between x_i and the center of B is less than some radius. The proxy trick proceeds by finding a surrogate representation $\mathsf{M}(\Gamma, \mathcal{I})$ for $\Sigma(\mathcal{F}, \mathcal{I})$ in the ID computation, such that $\mathsf{M}(\Gamma, \mathcal{I})$ has many fewer rows than $\Sigma(\mathcal{F}, \mathcal{I})$ and

$$\begin{bmatrix} \Sigma(\mathcal{N}, \mathcal{R}) \\ \mathsf{M}(\Gamma, \mathcal{R}) \end{bmatrix} \approx \begin{bmatrix} \Sigma(\mathcal{N}, \mathcal{S}) \\ \mathsf{M}(\Gamma, \mathcal{S}) \end{bmatrix} \mathsf{T}_{\mathcal{I}} \implies \begin{bmatrix} \Sigma(\mathcal{N}, \mathcal{R}) \\ \Sigma(\mathcal{F}, \mathcal{R}) \end{bmatrix} \approx \begin{bmatrix} \Sigma(\mathcal{N}, \mathcal{S}) \\ \Sigma(\mathcal{F}, \mathcal{S}) \end{bmatrix} \mathsf{T}_{\mathcal{I}} \tag{7}$$

such that we may compute the left ID in (7) and get the right ID for "free."

In the modified proxy trick, we let Γ be a set of n_{prox} points discretizing the gray annulus in the right of Figure 3. Crucially, this differs from the original proxy trick due to the fact that we discretize a two-dimensional region (the annulus), whereas if our kernel satisfied some form of a Green's identity we could instead discretize a quasi-one-dimensional curve (a circle) around B as in the original proxy trick. Defining the matrix $\mathsf{M}(\Gamma, \mathcal{I})$ to have entries $K(y, x_i; \theta)$ with rows indexed by $y \in \Gamma$ and columns indexed by $x_i \in \mathcal{I}$, we observe that (7) holds without significant loss in accuracy even with n_{prox} relatively small. This brings the complexity of computing the right ID down to $O(|\mathcal{I}|^3 + n_{\text{prox}}|\mathcal{I}|^2)$, which is beneficial when n_{prox} is small compared to $|\mathcal{I}^c|$. In practice, we take n_{prox} to be constant with respect to the total number of points n.

2.3.2. Complexity sketch using the modified proxy trick. Using the modified proxy trick, the cost of the recursive skeletonization factorization is essentially determined by the number of DOFs interior to each skeletonized subdomain, i.e., $|\mathcal{S}_{\ell;i}|$ for each $\ell = 1, \ldots, L$ and $i = 1, \ldots, 4^\ell$. As seen in Figure 2, the skeleton DOFs tend to line the boundaries of their corresponding subdomains. This is statistically intuitive: due to the fact that our kernels of interest $K(\cdot, \cdot; \theta)$ decay smoothly, the subset of DOFs that best represent the covariance structure of a subdomain with the rest of the domain is the subset closest to the rest of the domain.

Assuming a uniform distribution of points $\{x_i\}_{i=1}^n$ and perfect quadtree, the average number of skeleton DOFs per subdomain at level ℓ is on the order of the sidelength of a subdomain in the quadtree at level ℓ. In other words, the number of skeleton

TABLE 1
Complexity of the two-dimensional recursive skeletonization factorization.

Operation	Complexity
Construct $\mathsf{F} \approx \Sigma$	$O\left(n^{3/2}\right)$
Apply F or F^{-1} to a vector	$O(n \log n)$
Apply $\mathsf{F}^{1/2}$ or $\mathsf{F}^{-1/2}$ to a vector	$O(n \log n)$
Compute $\log \lvert\mathsf{F}\rvert$ from F	$O(n \log n)$

DOFs per box grows by roughly a factor of two each time we step up a level in the tree and thus $s_\ell \equiv \frac{1}{4^\ell} \sum_{i=1}^{4^\ell} |\mathcal{S}_{\ell;i}| = O(2^{-\ell})$. The assumptions that lead to this rank growth bound are described in more detail by Ho and Greengard [24, section 4]; we do not discuss them here.

THEOREM 3 (see [31, 25, 24]). *Assuming that the size of the skeleton sets behaves like $s_\ell = O(2^{-\ell})$ for $\ell = 1, \ldots, L$ and $L \sim \log n$, the computational complexity of the recursive skeletonization factorization $\mathsf{F} \approx \Sigma$ (with constants depending on the tolerance ϵ in Definition 2) is $T_{factor} = O(n^{3/2})$ and $T_{apply} = T_{solve} = O(n \log n)$, where T_{factor} is the complexity of the factorization and T_{apply} and T_{solve} are the complexities of applying F or F^{-1} to a vector. The storage complexity is $O(n \log n)$.*

From (6) we see that the application of the factorization F to a vector $x \in \mathbb{R}^n$ simply requires application of the block unit-triangular matrices $\mathsf{L}_\mathcal{I}$ and $\mathsf{U}_\mathcal{I}$ corresponding to each subdomain at each level as well as the block-diagonal Cholesky factor C of $\tilde{\Sigma}_1$. Further, the inverse of F can be applied by noting that

$$\mathsf{F}^{-1} = \left[\prod_{\ell=L}^{1} \left(\prod_{\mathcal{I} \in \mathcal{L}_\ell} \mathsf{P}_\ell^T \mathsf{U}_\mathcal{I}^{-1} \mathsf{L}_\mathcal{I}^{-1} \right) \right]^T \mathsf{C}^{-T} \mathsf{C}^{-1} \left[\prod_{\ell=L}^{1} \left(\prod_{\mathcal{I} \in \mathcal{L}_\ell} \mathsf{P}_\ell^T \mathsf{U}_\mathcal{I}^{-1} \mathsf{L}_\mathcal{I}^{-1} \right) \right].$$

Additionally, a generalized square root $\mathsf{F}^{1/2}$ such that $\mathsf{F} = \mathsf{F}^{1/2}(\mathsf{F}^{1/2})^T$ can be applied (as can its transpose or inverse) by taking

$$\mathsf{F}^{1/2} = \left[\prod_{\ell=1}^{L} \left(\prod_{\mathcal{I} \in \mathcal{L}_\ell} \mathsf{L}_\mathcal{I} \mathsf{U}_\mathcal{I} \right) \mathsf{P}_\ell \right] \mathsf{C}.$$

Finally, the log-determinant of Σ can be approximated by $\log |\Sigma| \approx \log |\mathsf{F}| = 2 \log |\mathsf{C}|$. Table 1 summarizes the computational complexities for these operations, which essentially follow from Theorem 3.

By constructing the recursive skeletonization factorizations F of Σ and F_i of Σ_i for $i = 1, \ldots, p$, we see that after the $O(n^{3/2})$ initial factorization cost each term in the evaluation of the log-likelihood or its gradient can be computed with cost $O(n \log n)$ except for the product traces $\mathrm{Tr}(\Sigma^{-1}\Sigma_i)$ for $i = 1, \ldots, p$. Further, through the approximate generalized square root $\mathsf{F}^{1/2}$ of Σ we can quickly sample from the distribution $N(0, \Sigma)$.

Remark 4. While the recursive skeletonization factorization described here exploits the most well-justified rank assumptions on the covariance kernel $K(\cdot, \cdot; \theta)$, each recursive skeletonization factorization in our framework can be replaced by the closely-related *hierarchical interpolative factorization* [25] or *strong recursive skeletonization factorization* [33], which are observed in practice to exhibit better scaling properties and also admit simple log-determinant computation.

3. Computing the trace terms. There are a number of methods for estimating the term $\mathrm{Tr}(\Sigma^{-1}\Sigma_i)$ appearing in each gradient component g_i for $i = 1, \ldots, p$. Employing the recursive skeletonization factorizations F of Σ and F_i of Σ_i, the product $\Sigma^{-1}\Sigma_i \approx \mathsf{F}^{-1}\mathsf{F}_i$ (or a symmetrized form with the same trace) can be applied to a vector with complexity $O(n \log n)$. Using G to denote this black-box linear operator, the classical statistical approach is the estimator of Hutchinson [26]. Drawing random vectors $u_i \in \{-1, 1\}^n$ for $i = 1, \ldots, q$ such that the components of u_i are independent and take value ± 1 with equal probability, the Hutchinson trace estimator is

$$\mathrm{Tr}(\mathsf{G}) \approx \frac{1}{q} \sum_{i=1}^{q} u_i^T \mathsf{G} u_i, \tag{8}$$

which is unbiased with variance decaying as $1/q$ and has cost $O(qn \log n)$. For low-accuracy estimates of the trace, the Hutchinson estimator is simple and computationally efficient, but for higher accuracy it proves computationally infeasible to use the Hutchinson approach because of the slow rate of convergence in q; see section 5.

When Σ and Σ_i have hierarchical rank structure, it is reasonable to also look for hierarchical rank structure in the product $\Sigma^{-1}\Sigma_i$, as matrix inversion and multiplication preserve such rank structure (albeit with different ranks) in many cases [18]. In our framework, we use the matrix peeling algorithm of Lin, Lu, and Ying [29] for constructing an explicit $\mathcal{H}$-matrix representation of a fast black-box operator G. At a high level, the method proceeds by applying the operator G to random vectors drawn with a specific sparsity structure to construct an approximate representation of the off-diagonal blocks at each level. We recursively perform low-rank compression level by level, following the same quadtree hierarchy as in the recursive skeletonization factorization, albeit in a top-down traversal rather than bottom-up. Finally, at the bottom level of the tree, the diagonal blocks corresponding to leaf-level subdomains can be extracted and their traces computed. While the full algorithm is applicable to both the strongly admissible and weakly admissible settings, the version of the algorithm we detail here is efficient for the simple weakly admissible case. We point the reader to Lin, Lu, and Ying [29] for more details related to the modifications required for strong admissibility.

The use of a randomized method for computing low-rank representations of matrices, which we review below, is integral to the peeling algorithm.

3.1. Randomized low-rank approximations. To begin, suppose that matrix $\mathsf{A} \in \mathbb{R}^{m_1 \times m_2}$ has (numerical) rank r and that we wish to construct an explicit rank-r approximation $\mathsf{A} \approx \mathsf{U}_1 \mathsf{M} \mathsf{U}_2^T$ with $\mathsf{U}_1 \in \mathbb{R}^{m_1 \times r}$, $\mathsf{U}_2 \in \mathbb{R}^{m_2 \times r}$, and $\mathsf{M} \in \mathbb{R}^{r \times r}$. In the context of approximating the trace terms, for example, A will be an off-diagonal block of $\Sigma^{-1}\Sigma_i$ or perhaps of a related symmetrized form with the same trace. Here we provide an overview of an algorithm that accomplishes this goal.

We begin by constructing approximations to the column space and row space of the matrix A. Let c be a small integer and suppose $\mathsf{W}_1 \in \mathbb{R}^{m_2 \times (r+c)}$ and $\mathsf{W}_2 \in \mathbb{R}^{m_1 \times (r+c)}$ are appropriately chosen random matrices, the distribution of which we will discuss later. Following Halko, Martinsson, and Tropp [22], let U_1 be a well-conditioned basis for the column space of $\mathsf{A}\mathsf{W}_1$ and U_2 be a well-conditioned basis for the column space of $\mathsf{A}^T\mathsf{W}_2$ constructed via, e.g., column-pivoted QR factorizations. Using the Moore–Penrose pseudoinverse, we obtain a low-rank approximation according to the approach summarized by Lin, Lu, and Ying [29, subsection 1.2] via

$$\mathsf{A} \approx \mathsf{U}_1 \left[(\mathsf{W}_2^T \mathsf{U}_1)^\dagger (\mathsf{W}_2^T \mathsf{A} \mathsf{W}_1) (\mathsf{U}_2^T \mathsf{W}_1)^\dagger \right] \mathsf{U}_2^T = \mathsf{U}_1 \mathsf{M} \mathsf{U}_2^T. \tag{9}$$

Perhaps surprisingly, with an appropriate choice of W_1 and W_2 it is the case that with high probability this approximation is *near optimal*, in the sense that

$$\|\mathsf{A} - \mathsf{U}_1\mathsf{M}\mathsf{U}_2^T\|_2 \leq \alpha(m_1, m_2, c)\|\mathsf{A} - \mathsf{A}_{r,\text{best}}\|_2,$$

where $\mathsf{A}_{r,\text{best}}$ is the best rank-r approximation of A and $\alpha(m_1, m_2, c)$ is a small factor dependent on c and the size of A. Further, the approximation process can be monitored and controlled adaptively to ensure a target desired accuracy [22].

It remains to discuss the choice of distribution for W_1 and W_2. The most common and straightforward choice is for both to have independently and identically distributed $N(0,1)$ entries, which guarantees the strongest analytical error bounds and highest success probability. Under this choice, one can show that the algorithm as stated takes $O(T_{\text{apply}}r + nr^2 + r^3)$, where T_{apply} is the complexity of applying A to a vector. This is sufficiently fast for our purposes, though we note that it is possible to accelerate this using other distributions [35, 22, 42].

3.2. Matrix peeling for weakly admissible matrices. For simplicity, we assume a perfect quadtree as in Remark 2. Further, we will assume that the numerical ranks of the off-diagonal blocks to a specified tolerance ϵ_{peel} are known a priori at each level, such that off-diagonal blocks of G at level ℓ have numerical rank at most r_ℓ. In practice, an adaptive procedure is used to find the ranks. Finally, we assume that G is symmetric, since if the trace of nonsymmetric G is required we can instead always consider a symmetrized form with the same trace such as $\frac{1}{2}(\mathsf{G} + \mathsf{G}^T)$.

3.2.1. First level of peeling algorithm. To begin, at level $\ell = 1$ the domain is partitioned into four subdomains $\Omega_{1;i}$ with corresponding index sets $\mathcal{I}_{1;i}$, $i = 1, \ldots, 4$ as in Figure 1. We follow the style of Lin, Lu, and Ying [29] and write the off-diagonal blocks at this level as $\mathsf{G}_{1;ij} \equiv \mathsf{G}(\mathcal{I}_{1;i}, \mathcal{I}_{1;j})$ to make our notation less cumbersome.

To construct randomized low-rank approximations of $\mathsf{G}_{1;ij}$ for $i \neq j$, we need to find the action of these off-diagonal blocks on random matrices as described in subsection 3.1. Define the block-sparse matrices

$$\mathsf{W}_1^{(1)} \equiv \begin{bmatrix} \mathsf{W}_{1;1} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \mathsf{W}_2^{(1)} \equiv \begin{bmatrix} 0 \\ \mathsf{W}_{1;2} \\ 0 \\ 0 \end{bmatrix}, \ \mathsf{W}_3^{(1)} \equiv \begin{bmatrix} 0 \\ 0 \\ \mathsf{W}_{1;3} \\ 0 \end{bmatrix}, \ \mathsf{W}_4^{(1)} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathsf{W}_{1;4} \end{bmatrix},$$

where $\mathsf{W}_{1;j}$ is a random matrix of dimension $|\mathcal{I}_{1;j}| \times (r_1 + c)$ for $j = 1, \ldots, 4$. Applying G to $\mathsf{W}_1^{(1)}$ gives the action of $\mathsf{G}_{1;1j}$ on the random matrix $\mathsf{W}_{1;1}$ for $j = 1, \ldots, 4$, since

$$\begin{bmatrix} \mathsf{G}_{1;11} & \mathsf{G}_{1;12} & \mathsf{G}_{1;13} & \mathsf{G}_{1;14} \\ \mathsf{G}_{1;21} & \mathsf{G}_{1;22} & \mathsf{G}_{1;23} & \mathsf{G}_{1;24} \\ \mathsf{G}_{1;31} & \mathsf{G}_{1;32} & \mathsf{G}_{1;33} & \mathsf{G}_{1;34} \\ \mathsf{G}_{1;41} & \mathsf{G}_{1;42} & \mathsf{G}_{1;43} & \mathsf{G}_{1;44} \end{bmatrix} \begin{bmatrix} \mathsf{W}_{1;1} \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \mathsf{G}_{1;11}\mathsf{W}_{1;1} \\ \mathsf{G}_{1;21}\mathsf{W}_{1;1} \\ \mathsf{G}_{1;31}\mathsf{W}_{1;1} \\ \mathsf{G}_{1;41}\mathsf{W}_{1;1} \end{bmatrix}.$$

The top block of the right-hand side vector above is unused as it involves a diagonal block of G. However, the remaining blocks are exactly the matrices $\mathsf{G}_{1;i1}\mathsf{W}_{1;1}$ for $i \neq 1$ as required by the randomized low-rank approximation of subsection 3.1. Applying G to each $\mathsf{W}_j^{(1)}$ for $j = 1, \ldots, 4$, for each block $\mathsf{G}_{1;ij}$ with $i \neq j$ we obtain a random

sampling of its column space $\mathsf{G}_{1;ij}\mathsf{W}_{1;j}$. Note that by symmetry of G we also obtain a random sampling of the row space of each block since $\mathsf{G}_{1;ij}\mathsf{W}_{1;j} = \mathsf{G}_{1;ji}^T\mathsf{W}_{1;j}$.

Using (9) to construct rank-r_1 approximations of each of these blocks, we write the approximation of $\mathsf{G}_{1;ij}$ as $\mathsf{G}_{1;ij} \approx \widehat{\mathsf{G}}_{1;ij} \equiv \mathsf{U}_{1;ij}\mathsf{M}_{1;ij}\mathsf{U}_{1;ji}^T$, where the approximation is accurate to the specified tolerance ϵ_{peel} with high probability.

Defining the matrix $\mathsf{G}^{(1)} \in \mathbb{R}^{n\times n}$ with blocks given by

$$\mathsf{G}^{(1)}\left(\mathcal{I}_{1;i}, \mathcal{I}_{1;j}\right) = \begin{cases} \widehat{\mathsf{G}}_{1;ij}, & i \neq j, \\ 0, & \text{else}, \end{cases}$$

we obtain

$$\mathsf{G} - \mathsf{G}^{(1)} \equiv \mathsf{G} - \begin{bmatrix} & \widehat{\mathsf{G}}_{1;12} & \widehat{\mathsf{G}}_{1;13} & \widehat{\mathsf{G}}_{1;14} \\ \widehat{\mathsf{G}}_{1;21} & & \widehat{\mathsf{G}}_{1;23} & \widehat{\mathsf{G}}_{1;24} \\ \widehat{\mathsf{G}}_{1;31} & \widehat{\mathsf{G}}_{1;32} & & \widehat{\mathsf{G}}_{1;34} \\ \widehat{\mathsf{G}}_{1;41} & \widehat{\mathsf{G}}_{1;42} & \widehat{\mathsf{G}}_{1;43} & \end{bmatrix} \approx \begin{bmatrix} \mathsf{G}_{1;11} & & & \\ & \mathsf{G}_{1;22} & & \\ & & \mathsf{G}_{1;33} & \\ & & & \mathsf{G}_{1;44} \end{bmatrix}.$$

In other words, we have approximated the off-diagonal blocks at this level to a specified accuracy and used the result to obtain a fast operator $\mathsf{G} - \mathsf{G}^{(1)}$ that is block diagonal with diagonal blocks the same as those of G.

Remark 5. We note that the matrix $\mathsf{G}^{(1)}$ is not explicitly assembled as a dense matrix inside the peeling algorithm. Instead, we store the nonzero blocks in low-rank form so that $\mathsf{G}^{(1)}$ may be efficiently applied to vectors.

3.2.2. Second level of peeling algorithm. In the next step of the peeling algorithm, we recurse on the diagonal subblocks $\mathsf{G}_{1;ii}$ for $i = 1, \ldots, 4$. Partitioning each subdomain $\Omega_{1;i}$ at level $\ell = 1$ into four child subdomains at level $\ell = 2$ using the quadtree structure and renumbering blocks accordingly, we write the diagonal blocks $\mathsf{G}_{1;ii}$ for $i = 1, \ldots, 4$ as

$$\mathsf{G}_{1;11} = \begin{bmatrix} \mathsf{G}_{2;11} & \mathsf{G}_{2;12} & \mathsf{G}_{2;13} & \mathsf{G}_{2;14} \\ \mathsf{G}_{2;21} & \mathsf{G}_{2;22} & \mathsf{G}_{2;23} & \mathsf{G}_{2;24} \\ \mathsf{G}_{2;31} & \mathsf{G}_{2;32} & \mathsf{G}_{2;33} & \mathsf{G}_{2;34} \\ \mathsf{G}_{2;41} & \mathsf{G}_{2;42} & \mathsf{G}_{2;43} & \mathsf{G}_{2;44} \end{bmatrix}, \; \mathsf{G}_{1;22} = \begin{bmatrix} \mathsf{G}_{2;55} & \mathsf{G}_{2;56} & \mathsf{G}_{2;57} & \mathsf{G}_{2;58} \\ \mathsf{G}_{2;65} & \mathsf{G}_{2;66} & \mathsf{G}_{2;67} & \mathsf{G}_{2;68} \\ \mathsf{G}_{2;75} & \mathsf{G}_{2;76} & \mathsf{G}_{2;77} & \mathsf{G}_{2;78} \\ \mathsf{G}_{2;85} & \mathsf{G}_{2;86} & \mathsf{G}_{2;87} & \mathsf{G}_{2;88} \end{bmatrix},$$

and so on for $\mathsf{G}_{1;33}$ and $\mathsf{G}_{1;44}$.

For each $j = 1, \ldots, 16$ we define the random matrix $\mathsf{W}_{2;j} \in \mathbb{R}^{|\mathcal{I}_{2;j}|\times(r_2+c)}$, which is appropriately sized to give a random sample of the column space of $\mathsf{G}_{2;ij}$ for each $i = 1, \ldots, 16$, $i \neq j$. We can minimize the number of times we apply the operator $\mathsf{G} - \mathsf{G}^{(1)}$ as follows due to its block diagonal structure. For each $k = 1, \ldots, 4$, we define $\mathsf{W}_k^{(2)} \in \mathbb{R}^{n\times(r_2+c)}$ to have rows divided into 16 blocks according to

$$\mathsf{W}_k^{(2)}(\mathcal{I}_{2;j}, :) = \begin{cases} \mathsf{W}_{2;j} & j \in \{k, k+4, k+8, k+12\}, \\ 0 & \text{else}. \end{cases}$$

In other words, block k of $\mathsf{W}_k^{(2)}$ is nonzero, as is every fourth block after k.

DEFINITION 4 (Quadtree siblings). *In the context of the quadtree decomposition of Ω, we say that $\Omega_{\ell;i}$ and $\Omega_{\ell;j}$ are* siblings *if $i \neq j$ and both $\Omega_{\ell;i} \subset \Omega_{\ell-1;k}$ and $\Omega_{\ell;j} \subset \Omega_{\ell-1;k}$ for some $1 \leq k \leq 4^{(\ell-1)}$.*

Let $\mathsf{B}_k \equiv (\mathsf{G} - \mathsf{G}^{(1)})\mathsf{W}_k^{(2)}$ and suppose that $\mathsf{W}_k^{(2)}(\mathcal{I}_{2;j},:)$ is nonzero. For each i such that $\Omega_{2;i}$ and $\Omega_{2;j}$ are siblings, we have $\mathsf{B}_k(\mathcal{I}_{2;i},:) \approx \mathsf{G}_{2;ij}\mathsf{W}_{2;j}$. For example, in $\mathsf{W}_1^{(2)}$ the nonzero blocks are $\mathsf{W}_1^{(2)}(\mathcal{I}_{2;j},:) = \mathsf{W}_{2;j}$ for $j \in \{1,5,9,13\}$, so

$$\mathsf{B}_1(\mathcal{I}_{2;i},:) \approx \begin{cases} \mathsf{G}_{2;i1}\mathsf{W}_{2;1}, & i = 1,\ldots,4, \\ \mathsf{G}_{2;i5}\mathsf{W}_{2;5}, & i = 5,\ldots,8, \\ \mathsf{G}_{2;i9}\mathsf{W}_{2;9}, & i = 9,\ldots,12, \\ \mathsf{G}_{2;i13}\mathsf{W}_{2;13}, & i = 13,\ldots,16. \end{cases}$$

Therefore, applying $\mathsf{G} - \mathsf{G}^{(1)}$ to $\mathsf{W}_k^{(2)}$ for $k = 1,\ldots,4$ gives a random sample of the column space and row space of $\mathsf{G}_{2;ij}$ for each i and j such that $\Omega_{2;i}$ and $\Omega_{2;j}$ are siblings. For all such i and j we use the randomized low-rank approximation algorithm as before to construct

$$\widehat{\mathsf{G}}_{2;ij} = \mathsf{U}_{2;ij}\mathsf{M}_{2;ij}\mathsf{U}_{2;ji}^T.$$

Defining $\mathsf{G}^{(2)} \in \mathbb{R}^{n\times n}$ with blocks

$$\mathsf{G}^{(2)}(\mathcal{I}_{2;i},\mathcal{I}_{2;j}) = \begin{cases} \widehat{\mathsf{G}}_{2;ij} & \text{if } \Omega_{2;i} \text{ and } \Omega_{2;j} \text{ are siblings,} \\ 0 & \text{else,} \end{cases}$$

we have that $\mathsf{G} - \mathsf{G}^{(1)} - \mathsf{G}^{(2)}$ is approximately block diagonal with diagonal blocks $\mathsf{G}_{2;ii}$ for $i = 1,\ldots,16$.

3.2.3. Subsequent levels of peeling algorithm. In general, at level $\ell > 2$ we see that $\mathsf{G} - \sum_{m=1}^{\ell-1}\mathsf{G}^{(m)}$ is approximately block diagonal with $4^{(\ell-1)}$ diagonal blocks. For each $k = 1,\ldots,4$ we define $\mathsf{W}_k^{(\ell)} \in \mathbb{R}^{n\times(r_\ell+c)}$ to have rows divided into 4^ℓ blocks according to

$$\mathsf{W}_k^{(\ell)}(\mathcal{I}_{\ell;j},:) = \begin{cases} \mathsf{W}_{\ell;j}, & j \equiv k \pmod 4, \\ 0, & \text{else,} \end{cases}$$

where each $\mathsf{W}_{\ell;j}$ is a random matrix of size $\mathbb{R}^{|\mathcal{I}_{\ell;j}|\times r_\ell}$. Using the same logic as in subsection 3.2.2, we apply $\mathsf{G} - \sum_{m=1}^{\ell-1}\mathsf{G}^{(m)}$ to $\mathsf{W}_k^{(\ell)}$ for each $k = 1,\ldots,4$ and use the results to construct low-rank approximations

$$\widehat{\mathsf{G}}_{\ell;ij} = \mathsf{U}_{\ell;ij}\mathsf{M}_{\ell;ij}\mathsf{U}_{\ell;ji}^T$$

for each i and j such that $\Omega_{\ell;i}$ and $\Omega_{\ell;j}$ are siblings. We define

$$\mathsf{G}^{(\ell)}(\mathcal{I}_{\ell;i},\mathcal{I}_{\ell;j}) = \begin{cases} \widehat{\mathsf{G}}_{\ell;ij} & \text{if } \Omega_{\ell;i} \text{ and } \Omega_{\ell;j} \text{ are siblings,} \\ 0 & \text{else} \end{cases}$$

such that $\mathsf{G} - \sum_{m=1}^{\ell}\mathsf{G}^{(m)}$ is approximately block diagonal with 4^ℓ diagonal blocks.

3.2.4. Extracting the trace. At the bottom level of the quadtree, each diagonal block of $\mathsf{G} - \sum_{m=1}^{L}\mathsf{G}^{(m)}$ is of a constant size independent of n as discussed in subsection 2.1. Define $n_{L;i} \equiv |\mathcal{I}_{L;i}|$ and $n_L \equiv \max_i n_{L;i}$ such that n_L is the maximum number of observations in a leaf-level subdomain. We construct a block matrix $\mathsf{E} \in \mathbb{R}^{n\times n_L}$ such that

$$\mathsf{E}(\mathcal{I}_{L;i},[n_{L;i}]) = \mathsf{I} \in \mathbb{R}^{|\mathcal{I}_{L;i}|\times|\mathcal{I}_{L;i}|}$$

for each i, where I is an appropriately sized identity matrix. Letting

$$\mathsf{H} \equiv \left(\mathsf{G} - \sum_{m=1}^{L} \mathsf{G}^{(m)} \right) \mathsf{E},$$

we find that $\mathsf{H}\left(\mathcal{I}_{L;i}, [n_{L;i}]\right) \approx \mathsf{G}_{L;ii}$ for each $i = 1, \ldots, 4^L$. We can then approximate the trace of G using the relation

$$\mathrm{Tr}(\mathsf{G}) = \sum_{i=1}^{4^L} \mathrm{Tr}(\mathsf{G}_{L;ii}) \approx \sum_{i=1}^{4^L} \mathrm{Tr}(\mathsf{H}\left(\mathcal{I}_{L;i}, [n_{L;i}]\right)).$$

Remark 6. When using the peeling algorithm to construct an approximate trace of an operator G with numerically low-rank off-diagonal blocks, it is important to note that we do not have direct control of the relative error of the trace approximation. This is because a matrix with diagonal entries with large absolute value but mixed signs can have a small trace due to cancellation. In practice, however, our numerical results in section 5 show excellent agreement between the approximate trace and true trace.

3.3. Computational complexity. For each level of the weak-admissibility-based peeling algorithm described in subsection 3.2 there are two key steps: applying the operator $\mathsf{G} - \sum_{m=1}^{\ell-1} \mathsf{G}^{(m)}$ and forming the low-rank factorizations $\widehat{\mathsf{G}}_{\ell;ij}$ for each i and j such that $\Omega_{\ell;i}$ and $\Omega_{\ell;j}$ are siblings. Analyzing the cost of these steps leads to the following complexity result.

THEOREM 5. *Let the cost of applying $\mathsf{G} \in \mathbb{R}^{n \times n}$ to a vector be T_{apply} and assume that the observations are uniformly distributed in Ω such that $|\mathcal{I}_{\ell;i}| = O(4^{-\ell} n)$ for each $1 \leq \ell \leq L$ and $1 \leq i \leq 4^\ell$. Assuming that the ranks of the off-diagonal blocks $\mathsf{G}_{\ell;ij}$ are bounded by r_ℓ for each i and j such that $\Omega_{\ell;i}$ and $\Omega_{\ell;j}$ are siblings, and define*

$$s_1 \equiv \sum_{\ell=1}^{L} r_\ell, \quad \text{and} \quad s_2 \equiv \sum_{\ell=1}^{L} r_\ell^2.$$

Then the complexity of the weak-admissibility-based peeling algorithm is

$$T_{peel} = O(T_{apply} s_1 + n s_2 \log n). \tag{10}$$

The storage complexity is $O(n s_1)$.

Proof. We adapt the proof of Lin, Lu, and Ying [29] to the weak admissibility case. At the first level, applying G to each $\mathsf{W}_k^{(1)}$ costs $O(T_{\text{apply}} r_1)$ and each randomized factorization costs $O(n r_1^2)$, leading to an overall cost for level 1 of $O(T_{\text{apply}} r_1 + n r_1^2)$.

At level $\ell > 1$, we break the cost of applying $\mathsf{G} - \sum_{m=1}^{\ell-1} \mathsf{G}^{(m)}$ into two pieces. The cost of applying G to each $\mathsf{W}_k^{(\ell)}$ is $O(T_{\text{apply}} r_\ell)$. The matrix $\sum_{m=1}^{\ell-1} \mathsf{G}^{(m)}$ is a heavily structured matrix with blocks in low-rank form. Applying this to each $\mathsf{W}_k^{(\ell)}$ costs $O\left(\sum_{m=1}^{\ell-1} n r_m r_\ell\right)$, which is $O(n s_2)$. We additionally must construct each randomized factorization at this level. Each one costs $O(4^{-\ell} n r_\ell^2)$ and there are $O(4^\ell)$ off-diagonal blocks to compress at this level, so the overall cost for level ℓ is $O(T_{\text{apply}} r_\ell + n s_2 + n r_\ell^2)$.

Summing the cost of each level from $\ell = 1, \ldots, L$, we obtain (10). Note that at level $\ell = L$, we must additionally extract the diagonal blocks, but by the assumption

TABLE 2

The runtime and storage complexity of the peeling algorithm depend on the asymptotic rank r_ℓ of off-diagonal blocks of G at level ℓ. The tabulated complexities are based on the assmption that the recursive skeletonization factorization is used to apply G as a fast operator.

r_ℓ	Time	Storage
$O(\log n_\ell)$	$\tilde{O}(n)$	$\tilde{O}(n)$
$O(\sqrt{n_\ell})$	$\tilde{O}\left(n^2\right)$	$\tilde{O}(n^{3/2})$

these blocks are of constant size so this does not increase the asymptotic cost. The storage complexity comes from noting that at level ℓ we must store the $O(4^\ell)$ matrices of rank r_ℓ, where each has outer dimension that is $O(4^{-\ell}n)$. ∎

When the underlying matrix has the rank of all off-diagonal blocks bounded by $r_\ell = O(1)$ for all ℓ, then the computational complexity of weak peeling is $\tilde{O}(T_{\text{apply}}+n)$, where we use the so-called "soft-O" notation from theoretical computer science to suppress factors that are polylogarithmic in n. In this case, peeling $\mathsf{G} = \mathsf{\Sigma}$ itself using its recursive skeletonization factorization results in $\tilde{O}(n)$ complexity for both time and memory.

Many real matrices of interest, however, do not exhibit off-diagonal blocks with ranks independent of n. For example, our experiments with the Matérn kernel of (3) show that a constant number of off-diagonal blocks at each level of the hierarchy exhibit ranks bounded only as $r_\ell = O(2^{-\ell}\sqrt{n})$. This coincides with the argument for rank growth in the recursive skeletonization factorization in subsection 2.3. Thus, this simplified peeling algorithm in the case of the Matérn kernel has asymptotic time complexity $\tilde{O}(n^2)$ and storage complexity $\tilde{O}(n^{3/2})$, where we pick up at most a polylogarithmic factor in the ranks since we are looking at $\mathsf{G} = \mathsf{\Sigma}^{-1}\mathsf{\Sigma}_i$ and not $\mathsf{\Sigma}$ itself. In theory, using the simple peeling algorithm described here is asymptotically no better than extracting the trace by applying G to the coordinate vectors e_i for $i = 1, \ldots, n$. This necessitates the standard form of peeling for large problems.

Remark 7. In practice, the standard form of the peeling algorithm [29] that uses the full generality of strong admissibility can be employed to remedy such rank growth by explicitly avoiding compression of off-diagonal blocks that are not sufficiently low rank. Using the modifications described in that paper, the complexity of peeling follows the same bound as Theorem 5 but with the rank bound r_ℓ referring to a bound on the ranks of only those blocks that are compressed in the strongly admissible hierarchical format. We find in subsection 5.1 that using peeling based on strong admissibility is more efficient when n is large, as expected. However, the implicit constants in the asymptotic runtime lead to weak admissibility being more efficient for moderately sized problems.

We summarize our complexity results in Table 2. Note that these results were derived on the assumption that $n_\ell = O(4^{-\ell}n)$, i.e., a quasi-uniform distribution of observations and a perfect quadtree decomposition of space. In practice, observations that are distributed in a different fashion can actually exhibit better behavior, particularly if the observations are concentrated around a quasi-one-dimensional curve [25].

4. Summary of MLE framework. In Algorithm 1 we summarize our complete framework for computing the log-likelihood $\ell(\theta)$ and gradient $g(\theta)$ given θ, which can be used inside of any first-order optimization routine for Gaussian process maximum

likelihood estimation. As mentioned previously, the approach is flexible and does not rely on the specific hierarchical factorization used (e.g., the recursive skeletonization factorization, the hierarchical interpolative factorization, the strong recursive skeletonization factorization) or the form of peeling used (i.e., the peeling based on weak admissibility described in section 3 or the form by Lin, Lu, and Ying [29] based on strong admissibility). Rather, the exact components of the framework should be decided on a case-by-case basis depending on the rank properties of the kernel family.

Algorithm 1 Computing the Gaussian process log-likelihood and gradient.

Given: observation vector $z \in \mathbb{R}^n$, observation locations $\{x_i\}_{i=1}^n \subset \mathbb{R}^2$, peel tolerance ϵ_{peel}, factorization tolerance $\epsilon_{\text{fact}} < \epsilon_{\text{peel}}$, parameter vector $\theta \in \mathbb{R}^p$, and covariance kernel $K(\cdot, \cdot; \theta)$

1: `// Factor Σ with hierarchical factorization`
2: $\mathsf{F} \leftarrow$ Recursive skeletonization factorization of Σ with tolerance ϵ_{fact}
3: `// Use fast hierarchical solve and log-determinant`
4: $\hat{\ell}(\theta) \leftarrow -\frac{1}{2} z^T \mathsf{F}^{-1} z - \frac{1}{2} \log |\mathsf{F}| - \frac{1}{2} \log 2\pi \approx \ell(\theta)$
5: **for** $i = 1, \ldots, p$ **do**
6: `// Factor Σ_i with hierarchical factorization`
7: $\mathsf{F}_i \leftarrow$ Recursive skeletonization factorization of Σ_i with tolerance ϵ_{fact})
8: `// Compute trace of Σ^{-1}Σ_i with peeling algorithm`
9: $t_i \leftarrow$ Trace of operator $\frac{1}{2}(\mathsf{F}^{-1}\mathsf{F}_i + \mathsf{F}_i\mathsf{F}^{-1})$ via peeling algorithm with tolerance ϵ_{peel}
10: `// Use fast hierarchical apply and solve`
11: $\hat{g}_i \leftarrow \frac{1}{2} z^T \mathsf{F}^{-1}\mathsf{F}_i\mathsf{F}^{-1} z - \frac{1}{2} t_i \approx g_i$
12: **end for**

Output: $\hat{\ell}(\theta)$ and $\hat{g}(\theta)$

Remark 8. After estimation of the parameter vector θ, there remains the question of how to sample from the Gaussian process conditioned on the observed data z. Assuming $[z', z]^T$ is jointly distributed according to the original Gaussian process, this conditional distribution is given by

$$z'|z \sim N(\Sigma_{12}\Sigma_{22}^{-1} z,\, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^T),$$

where Σ_{11} is the covariance matrix of z', Σ_{22} is the covariance matrix of z, and so on. Using the identity

$$\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^T = \begin{bmatrix} \mathsf{I} & -\Sigma_{12}\Sigma_{22}^{-1} \end{bmatrix} \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix} \begin{bmatrix} \mathsf{I} \\ -\Sigma_{22}^{-1}\Sigma_{12}^T \end{bmatrix}$$

and letting Σ denote the two-by-two block matrix in a slight abuse of notation, we can apply a square root of $\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^T$ with skeletonization factorizations $\mathsf{F}_{22} \approx \Sigma_{22}$ and $\mathsf{F} \approx \Sigma$ by using $\mathsf{F}^{1/2}$ to apply a square root of Σ, F to apply Σ_{12} through appropriate padding, and F_{22}^{-1} to apply Σ_{22}^{-1}. This gives a fast method for sampling from the conditional distribution or computing the conditional mean.

5. Numerical results. To demonstrate the effectiveness of our approach to Gaussian process MLE, we first test the accuracy and runtime of the peeling-based technique for approximating the trace and then test our full method on two examples using synthetic datasets and one example using a dataset of measurements of ocean surface temperatures. In all of our examples, we take the number of proxy points to be $n_{\text{prox}} = 256$, and use a quadtree decomposition of space with a maximum of $n_{\text{occ}} = 64$ points per leaf subdomain.

In our tests we use the FLAM library (https://github.com/klho/FLAM/) for the recursive skeletonization factorization and a custom implementation of matrix peeling as described in subsection 3.2. This additional code is available at https://github.com/asdamle/GPMLE/. All numerical results shown were run in MATLAB R2015a on a quad-socket Intel Xeon E5-4640 processor clocked at 2.4 GHz using up to 1.5 TB of RAM.

5.1. Runtime scaling of the peeling algorithm. To begin, we investigate the numerical performance of the peeling algorithm on synthetic examples. We take the observation locations $\{x_i\}_{i=1}^n$ to be a $\sqrt{n}\times\sqrt{n}$ grid of points uniformly discretizing the square $[0,100]^2 \subset \mathbb{R}^2$. We let $\theta = [\theta_1, \theta_2]$ parameterize the correlation length scale of the process in each coordinate direction, defining the scaled distance

$$\|x-y\|_\theta^2 = \frac{(x_1-y_1)^2}{\theta_1^2} + \frac{(x_2-y_2)^2}{\theta_2^2},$$

where here x_i and y_i are used to denote components of vectors x and y. Using this parameterization and incorporating an additive noise term, the two kernels we test are the rational quadratic kernel of (2) with $\alpha = 1/2$,

$$K_{RQ}(x,y;\theta) = \left(1+\|x-y\|_\theta^2\right)^{-1/2} + \sigma_N^2\delta_{xy}, \tag{11}$$

and the Matérn kernel of (3) with parameter $\nu = 3/2$,

$$K_M(x,y;\theta) = (1+\sqrt{3}\|x-y\|_\theta)\exp(-\sqrt{3}\|x-y\|_\theta) + \sigma_N^2\delta_{xy}. \tag{12}$$

Here δ_{xy} is the Kronecker delta which satisfies $\delta_{xy} = 1$ if $x = y$ and $\delta_{xy} = 0$ otherwise.

Remark 9. In both (11) and (12) the additional term $\sigma_N^2\delta_{xy}$ can be interpreted as modeling additive white noise with variance σ_N^2 on top of the base Gaussian process model. In practice, this so-called "nugget effect" is frequently incorporated to account for measurement error or small-scale variation from other sources [32] and, further, is numerically necessary for many choices of parameter θ due to exceedingly poor conditioning of many kernel matrices.

We compute high-accuracy recursive skeletonization factorizations of the matrices Σ and $\Sigma_1 \equiv \frac{\partial}{\partial\theta_1}\Sigma$, which we combine to obtain the fast black-box operator

$$\mathsf{G} = \frac{1}{2}(\Sigma^{-1}\Sigma_1 + \Sigma_1\Sigma^{-1}) \tag{13}$$

for input to the peeling algorithm to compute the trace to specified tolerance $\epsilon_{\text{peel}} = 1\times10^{-6}$. We choose the parameter vector $\theta = [10, 7]$ for these examples as in Figure 4 (left), and set the noise parameter at $\sigma_N^2 = 1\times10^{-4}$.

Beginning with the rational quadratic kernel, in Table 3 we give runtime results for both the simplified peeling algorithm described in subsection 3.2 ("weak peeling") as well as the full strong-admissibility-based peeling algorithm of Lin, Lu, and Ying [29] ("strong peeling"). As can be seen in Figure 5 (left), the runtime of the peeling algorithm with the kernel (11) seems to scale between $O(n)$ and $O(n^{3/2})$ with the number of observations n, regardless of whether weak or strong peeling is used. Further, the relative error in the trace approximation, e_{peel}, is near the specified tolerance ϵ_{peel}, though the tolerance is not a hard upper bound. Note that we omit the relative error for our largest example, as the operator was too large to determine the true trace using the naïve approach.

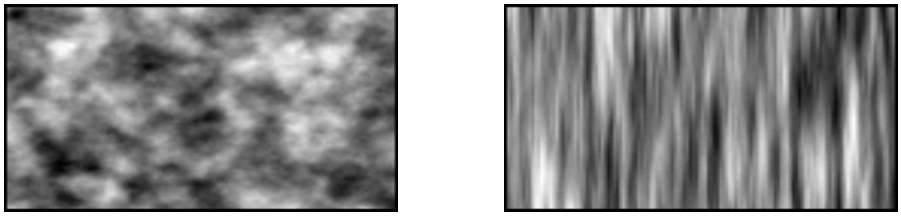

FIG. 4. *Two different realizations on the domain* $[0, 200] \times [0, 100]$ *of the Matérn kernel Gaussian process with covariance seen in* (12) *and noise parameter* $\sigma_N^2 = 0$. *In the left figure the parameter vector is* $\theta = [10, 7]$ *corresponding to a kernel that is relatively close to isotropic. In contrast, in the right figure the parameter vector* $\theta = [3, 30]$ *generates strong anisotropy.*

TABLE 3

Runtime t_{peel} *of the the peeling algorithm with the rational quadratic kernel of* (11). *Note that we omit the relative error* e_{peel} *in the estimated trace for our largest example, as the operator was too large to determine the true trace using the naïve approach.*

n	$t_{\text{peel,weak}}$ (s)	$e_{\text{peel,weak}}$	$t_{\text{peel,strong}}$ (s)	$e_{\text{peel,strong}}$
64^2	9.17×10^0	5.68×10^{-7}	4.85×10^1	1.60×10^{-7}
128^2	9.16×10^1	1.02×10^{-5}	5.64×10^2	2.58×10^{-7}
256^2	6.63×10^2	3.72×10^{-5}	3.04×10^3	3.32×10^{-6}
512^2	2.88×10^3	-	1.73×10^4	-

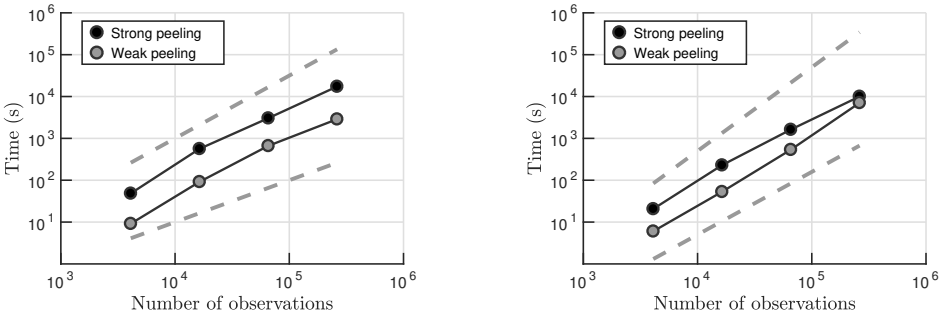

FIG. 5. *On the left the runtime of peeling for the rational quadratic kernel is plotted along with an* $O(n^{3/2})$ *trend line (top) and an* $O(n)$ *trend line (bottom), showing subquadratic scaling for weak peeling in this case. In contrast, on the right the runtime of peeling for the Matérn kernel is plotted along with an* $O(n^2)$ *trend line (top) and an* $O(n^{3/2})$ *trend line (bottom). We see that weak peeling with the Matérn kernel seems to ultimately exhibit quadratic scaling, whereas strong peeling seems to exhibit slightly better than* $O(n^{3/2})$ *scaling. The corresponding data are given in Tables* 3 and 4.

In contrast, the results in Table 4 for the Matérn kernel in (12) show different scaling behavior for weak and strong peeling. In Figure 5 (right), we see that the runtime for weak peeling seems to be close to quadratic in the number of observations, which agrees with our analysis from section 3. Using strong peeling, however, the complexity of peeling scales considerably better, ultimately following the $O(n^{3/2})$ trend line. We see again that the relative trace error is well-controlled by ϵ_{peel} in both cases.

Though the observed scaling behavior of strong peeling is as good as or better than that for weak peeling for both kernels, in practice we see that for problems with up to a quarter of a million observations weak peeling has a smaller time to solution. As such, in the remainder of our examples we show results using only weak peeling.

TABLE 4

Runtime t_{peel} of the the peeling algorithm with the Matérn kernel of (12)*. Note that we omit the relative error e_{peel} in the estimated trace for our largest example, as the operator was too large to determine the true trace using the naïve approach.*

n	$t_{\text{peel,weak}}$ (s)	$e_{\text{peel,weak}}$	$t_{\text{peel,strong}}$ (s)	$e_{\text{peel,strong}}$
64^2	6.03×10^0	5.73×10^{-8}	2.06×10^1	4.78×10^{-10}
128^2	5.30×10^1	2.46×10^{-7}	2.29×10^2	3.36×10^{-10}
256^2	5.37×10^2	4.28×10^{-6}	1.62×10^3	8.14×10^{-10}
512^2	7.07×10^3	-	1.00×10^4	-

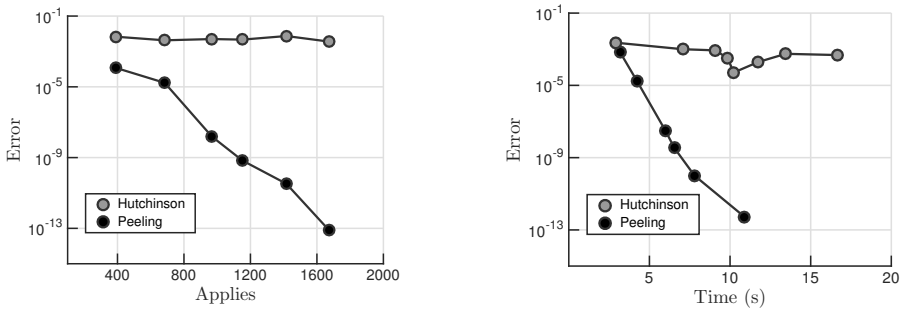

FIG. 6. *Plotting the relative error in the trace approximation versus the number of applications of the black-box operator, we see in the left figure that the Hutchinson estimator exhibits characteristic inverse square root convergence as dictated by the central limit theorem. In contrast, using the peeling algorithm described in subsection* 3.2*, we see that the same number of black-box applies yields a much improved accuracy, though the rate of convergence depends on the spectra of off-diagonal blocks of the operator. In the right figure, we plot the error of each method versus wall-clock time to establish that the same scaling behavior holds when error is viewed as a function of time to solution.*

5.2. Relative efficiency of peeling versus the Hutchinson estimator. As discussed in section 3, a common alternative statistical approach for approximating the trace of a matrix G is the estimator of Hutchinson [26] seen in (8). The aim of this section is to show that for matrices with hierarchical low-rank structure our peeling-based algorithm can be much more efficient when a high-accuracy trace approximation is desired.

As in subsection 5.1, we take our observations to be a regular grid discretizing $[0, 100]^2 \subset \mathbb{R}^2$ using the Matérn kernel of (12) with noise $\sigma_N^2 = 1 \times 10^{-4}$ and parameter vector $\theta = [10, 7]$. We fix the number of observations at $n = 64^2$ and consider how the accuracy of the trace approximation varies with the number of applications of the black-box operator for both weak peeling and the Hutchinson estimator.

Using a high-accuracy recursive skeletonization factorization to construct the black-box operator in (13) as in subsection 5.1, we vary the tolerance ϵ_{peel} in the peeling algorithm and plot in Figure 6 the relative error in the trace approximation as a function of both the number of black-box applies and total peeling runtime. Additionally, for the Hutchinson estimator we use the same factorizations to construct the unsymmetric operator $\mathsf{G}' = \Sigma^{-1}\Sigma_1$. We plot the same quantities for a given instantiation of the estimator for comparison.

For low-accuracy approximations with relative error on the order of 1×10^{-1} to 1×10^{-3}, we see that the Hutchinson estimator is a competitive alternative to the peeling algorithm for finding the trace. When increased accuracy is desired, however, it is clear that in our examples the peeling algorithm is the more attractive option. While the Hutchinson estimator has a simple form and is easy to compute,

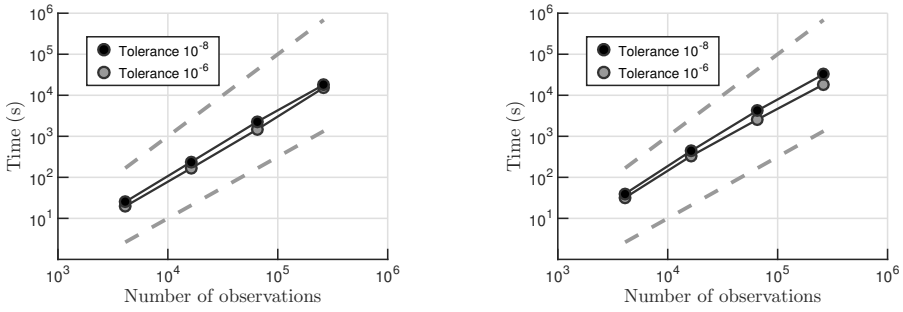

FIG. 7. *On the left we plot the total runtime of evaluating a single objective function and gradient for the uniform grid example of section* 5.3 *as a function of the total number of observations for two different tolerances. The top trend line shows* $O(n^2)$ *scaling and the bottom shows* $O(n^{3/2})$ *scaling. On the right we plot the corresponding results for the scattered data of section* 5.4 *with the same trend lines. We observe that the scaling in all cases looks like* $O(n^{3/2})$.

the relatively slow inverse square root convergence means that M in (8) must be taken to be exceedingly large to drive the variance down to reasonable levels, whereas the peeling algorithm is observed to make more economical use of its black-box matrix-vector products. It is worth noting that, for this choice of n, only 4096 applies are needed to explicitly construct all diagonal entries of the operator via application to the identity, though this is not feasible for larger n.

5.3. Gridded synthetic data example. We now profile a full objective function and gradient evaluation for the MLE problem for θ. As before, we consider the Matérn kernel of (12) with noise $\sigma_N^2 = 1 \times 10^{-4}$.

We set the parameter vector at $\theta = [10, 7]$ and again take the observation locations to be a regular $\sqrt{n} \times \sqrt{n}$ grid discretizing the square $[0, 100]^2$. Evaluating $\ell(\theta)$ and g_i for $i = 1, \ldots, 2$ then requires three skeletonization factorizations and two different trace approximations. We investigate the algorithm's performance for two different peeling tolerances ϵ_{peel}, and in each case take the factorization tolerance to be $\epsilon_{\text{fact}} = \frac{1}{1000}\epsilon_{\text{peel}}$. For varying n between 64^2 and 512^2, we measured the runtime of both the factorization portion and peeling portion of Algorithm 1. We note that, given the factorizations and peeled trace estimates, the remainining pieces of Algorithm 1 are several orders of magnitude less costly in terms of runtime.

In Figure 7 (left), we plot the total runtime for a single objective function and gradient evaluation for the uniform grid of observations (corresponding data in Table 5). We see from the figure that the runtime seems to scale as roughly $O(n^{3/2})$ with the number of observations; a least-squares fit of the data gives $O(n^{1.6})$. As can be seen in the table, the amount of time spent in calculating the recursive skeletonization factorizations is roughly an order of magnitude less than the time spent in the peeling trace approximation and, further, scales slightly better than peeling for this example.

5.4. Scattered synthetic data example. While all examples thus far have used a regular grid of observations, our framework does not rely on this assumption. To complement the examples on gridded observations, we repeat the same experiment from the previous section with real-world observation locations coming from release 2.5 of the International Comprehensive Ocean-Atmosphere Data Set (ICOADS) [46] obtained from the National Center for Atmospheric Research at http://rda.ucar.edu/datasets/ds540.0/. We subselect from ICOADS a set of sea surface temperatures measured at varying locations in the North Atlantic ocean between the years 2008

TABLE 5

Runtime for one objective function and gradient evaluation (i.e., the work for a single iteration) on a uniform grid of observations.

ϵ_{peel}	n	t_{fact} (s)	$t_{\text{peel,weak}}$ (s)	t_{total} (s)
1×10^{-6}	64^2	7.34×10^0	1.23×10^1	1.96×10^1
	128^2	4.86×10^1	1.20×10^2	1.68×10^2
	256^2	2.73×10^2	1.22×10^3	1.49×10^3
	512^2	1.41×10^3	1.39×10^4	1.53×10^4
1×10^{-8}	64^2	9.70×10^0	1.55×10^1	2.52×10^1
	128^2	7.08×10^1	1.68×10^2	2.39×10^2
	256^2	4.28×10^2	1.76×10^3	2.29×10^3
	512^2	2.64×10^3	1.54×10^4	1.80×10^4

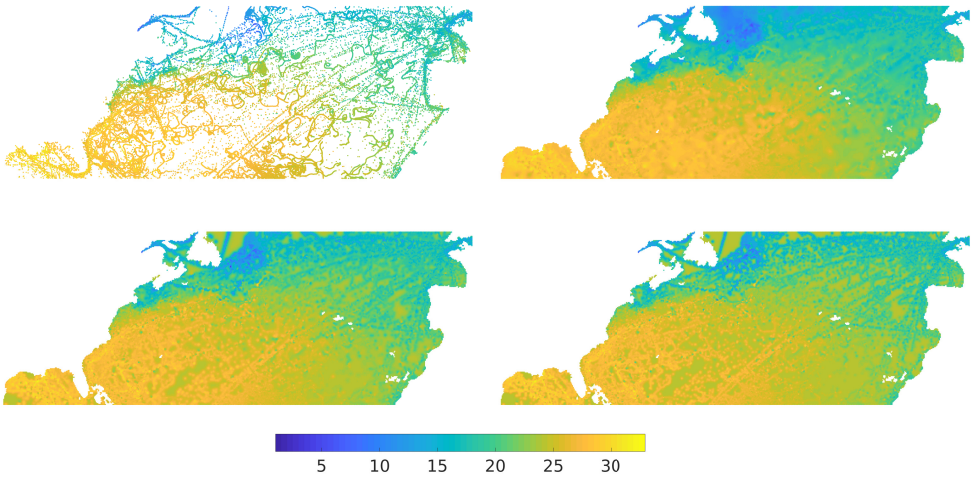

FIG. 8. *In the top-left plot, we show a subselection of $n = 2^{17}$ Atlantic ocean surface temperature measurements from ICOADS projected to a two-dimensional plane through Mercator projection and then scattered on top of a white background for visualization. Fitting the model with constant mean and covariance given by* (14)*, we use the estimated parameters to find the conditional mean temperatures throughout this region of the Atlantic for $\nu = 1/2$ (top right), $\nu = 3/2$ (bottom left), and $\nu = 5/2$ (bottom right). The color bar shows the estimated sea surface temperature in Celsius.*

and 2014. Restricting the data to observations made in the month of July across all years we obtain roughly 300,000 unique observation locations and corresponding sea surface temperature measurements, some of which can be seen in Figure 8.

Because large-scale spatial measurements typically cover a nontrivial range of latitudes and longitudes, the development of valid covariance functions on the entire sphere that respect the proper distance metric has been the subject of much recent work; see, e.g., Gneiting [14] and related work [34, 27, 23]. As the focus of this manuscript is not statistical modeling, we employ a simplified model based on Mercator projection of the observations to two spatial dimensions. Note that the choice of axis scaling in the Mercator projection is arbitrary; in our convention the horizontal axis spans 90 units and the vertical axis spans 70 units.

To perform scaling tests on the cost of an objective function and gradient evaluation according to Algorithm 1, we subselect from our full dataset of unique observation locations by drawing observations uniformly at random without replacement. Figure 7 (right) shows the runtime scaling results as a function of the number of observations, with corresponding data in Table 6. We see that the runtime scaling for the scattered

TABLE 6
Runtime for one objective function and gradient evaluation (i.e., the work for a single iteration) on scattered observations with locations from ICOADS.

ϵ_{peel}	n	t_{fact} (s)	$t_{\text{peel,weak}}$ (s)	t_{total} (s)
1×10^{-6}	2^{12}	7.82×10^{0}	2.41×10^{1}	3.19×10^{1}
	2^{14}	4.25×10^{1}	2.91×10^{2}	3.34×10^{2}
	2^{16}	8.60×10^{1}	2.49×10^{3}	2.57×10^{3}
	2^{18}	4.84×10^{2}	1.79×10^{4}	1.84×10^{4}
1×10^{-8}	2^{12}	1.06×10^{1}	2.92×10^{1}	3.98×10^{1}
	2^{14}	5.96×10^{1}	3.83×10^{2}	4.43×10^{2}
	2^{16}	2.72×10^{2}	3.98×10^{3}	4.26×10^{3}
	2^{18}	1.03×10^{3}	3.15×10^{4}	3.25×10^{4}

observations follows essentially the same scaling behavior as the gridded observations from subsection 5.3, with observed complexity between $O(n^{1.5})$ and $O(n^{1.6})$. Again, the skeletonization factorizations take considerably less time than the trace estimation.

As an illustrative example of the full power of Algorithm 1 in context, we take a subset of $n = 2^{16}$ scattered observations and realize an instance of a Gaussian process at those locations with true parameter vector $\theta^* = [10, 7]$ and noise parameter $\sigma_N^2 = 1 \times 10^{-4}$ to generate the observation vector z. Setting the peel tolerance to $\epsilon_{\text{peel}} = 1 \times 10^{-6}$ and the factorization tolerance to $\epsilon_{\text{fact}} = 1 \times 10^{-9}$, we plugged our approximate log-likelihood and gradient routines into the MATLAB routine `fminunc` for unconstrainted optimization using the quasi-Newton option. Starting from an initial guess of $\theta_0 = [3, 30]$, we found that after 13 iterations (14 calls to Algorithm 1) the first-order optimality as measured by the ℓ_∞-norm of the gradient had been reduced by three orders of magnitude, yielding an estimate of $\hat{\theta} = [10.0487, 7.0496]$ after approximately 4.86×10^4 seconds.

Remark 10. While a large percentage of this runtime was spent in the peeling algorithm, we find it worthwhile to note that in this example the use of our gradient approximation proved essential—using finite difference approximations to the gradient led to stagnation at the first iteration, even with a factorization tolerance $\epsilon_{\text{fact}} = 1 \times 10^{-15}$, i.e., at the limits of machine precision.

Because the number of iterations to convergence depends on many factors (e.g., the choice of optimization algorithm, how well the data can be modeled by a Gaussian process, and many convergence tolerances depending on the chosen algorithm), we do not find it useful to attempt to profile the full minimization algorithm more extensively than this, but direct the reader instead to the single-iteration results.

5.5. Scattered ocean data example. While the factorizations and peeling in Algorithm 1 depend only on the locations of the observations and not their values, the log-likelihood $\ell(\cdot)$ can have a more complicated shape with real observations z than with synthetic data, which may impact the required tolerance parameters ϵ_{fact} and ϵ_{peel} and the difficulty of MLE. Further, there are a number of practical considerations relevant for real data not addressed thus far in our synthetic examples.

As a refinement of (1), suppose now that the data are distributed according to $z \sim N(\mu\mathbf{1}, \sigma^2\Sigma(\theta))$, where $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector, μ and σ^2 represent the constant but unknown mean and variance level, and our parameterized Matérn model

is given by (for several different choices of ν)

$$[\Sigma(\theta)]_{ij} = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}r_{ij}}{\rho}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}r_{ij}}{\rho}\right) + \sigma_N^2\delta_{ij} \tag{14}$$

$$= \begin{cases} \exp\left(-\frac{r_{ij}}{\rho}\right), & \nu = 1/2, \\ \left(1+\frac{\sqrt{3}r_{ij}}{\rho}\right)\exp\left(-\frac{\sqrt{3}r_{ij}}{\rho}\right) + \sigma_N^2\delta_{ij}, & \nu = 3/2, \\ \left(1+\frac{\sqrt{5}r_{ij}}{\rho}+\frac{5r_{ij}^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}r_{ij}}{\rho}\right) + \sigma_N^2\delta_{ij}, & \nu = 5/2, \end{cases}$$

with $r_{ij} = \|x_i - x_j\|$. In this example, the parameter vector is $\theta = [\rho, \sigma_N^2]$, consisting of a single correlation length parameter and the noise level. To optimize the new log-likelihood over μ, σ^2, and θ, we note that optimization over μ and σ^2 results in closed form expressions for these parameters in terms of θ, which may then be substituted back into (4) to obtain the *log profile likelihood* for this model

$$\tilde{\ell}(\theta) \equiv -\frac{1}{2}\log|\Sigma| - \frac{n}{2}\log\left(z^T(\Sigma + \mathbf{1}\mathbf{1}^T)^{-1}z\right) + \frac{n}{2}\left(\log n - 1 - 2\pi\right) \tag{15}$$

with gradient components given by

$$\tilde{g}_i \equiv -\frac{1}{2}\operatorname{Tr}(\Sigma^{-1}\Sigma_i) + \frac{n}{2}\left(\frac{z^T(\Sigma+\mathbf{1}\mathbf{1}^T)^{-1}\Sigma_i(\Sigma+\mathbf{1}\mathbf{1}^T)^{-1}z}{z^T(\Sigma+\mathbf{1}\mathbf{1}^T)^{-1}z}\right), \quad i = 1, \ldots, p.$$

Optimization of this new model fits neatly into the computational framework of Algorithm 1 with trivial modifications. The new model has the advantage of greater plausibility, though it still admits many further improvements.

From the full set of sea surface temperature observations, we subselected $n = 2^{17}$ unique temperature measurements corresponding to observations between July 2013 and August 2013. Taking $\epsilon_{\text{fact}} = 1 \times 10^{-9}$ and $\epsilon_{\text{peel}} = 1 \times 10^{-6}$, we use the MATLAB optimization routine `fmincon` with the SQP option to estimate the correlation length parameter $\theta_1 = \rho$ and noise parameter $\theta_2 = \sigma_N^2$ for the standardized temperature measurements. For $\nu = 3/2$ and $\nu = 5/2$ this was accomplished by numerically maximizing (15) subject to the lower-bound constraint $\sigma_N^2 \geq 1 \times 10^{-5}$, which was necessary to ensure Σ was not numerically rank deficient. For $\nu = 1/2$ the covariance matrix Σ is naturally better conditioned so a looser lower bound $\sigma_N^2 \geq 1 \times 10^{-8}$ was used. The tolerances dictating the minimum step size and minimum change in objective function between successive iterates were both set to 1×10^{-6}.

Due to the nonconvex nature of the problem, we tried several choices of starting parameter for each ν; the results we present are for the best initialization in each case. For choices of initial parameters leading to convergent iterates (e.g., $\theta_0 = [5, 1]$ or $\theta_0 = [1 \times 10^{-1}, 1 \times 10^{-3}]$), the converged solutions all agreed to the specified tolerance and the objective function value at the optimal points agreed to six digits. For some choices of initial parameters, the optimization terminated prematurely due to the relative improvement tolerances used to evaluate convergence (i.e., when the initial parameters are very poor, even a large improvement relative to the initial parameters can be far from the best choice of parameters). We did not observe any evidence of multiple local optima, though the possibility that our reported parameters are globally suboptimal cannot be ruled out.

The results of our numerical optimization for each choice of ν can be seen in Table 7, where in each case optimization terminated due to the step-size tolerance.

TABLE 7

Parameter estimates $\hat{\theta} = [\hat{\rho}, \widehat{\sigma_N^2}]$, the corresponding mean and variance level $\mu(\hat{\theta})$ and $\sigma^2(\hat{\theta})$, and the log-likelihood values of the fitted parameters for the model with kernel (14)*. We note that for $\nu = 1/2$, the estimate of σ_N^2 is at its lower bound.*

ν	$\hat{\rho}$	$\widehat{\sigma_N^2}$	$\mu(\hat{\theta})$	$\sigma^2(\hat{\theta})$	$\ell(\hat{\theta})$
1/2	$2.12 \times 10^{+0}$	1.00×10^{-8}	-5.65×10^{-1}	9.97×10^{-1}	$9.28 \times 10^{+4}$
3/2	2.52×10^{-1}	4.37×10^{-3}	-4.38×10^{-1}	8.71×10^{-1}	$8.42 \times 10^{+4}$
5/2	1.78×10^{-1}	5.67×10^{-3}	-4.30×10^{-1}	8.57×10^{-1}	$7.74 \times 10^{+4}$

At each corresponding $\hat{\theta}$, however, we note that the gradient is small relative to the objective function.

Of the three different models, we find that the fitted model for $\nu = 1/2$ gives the best fit as measured both by comparative likelihood and qualitatively (see Figure 8). Since ν dictates the smoothness of the denoised process, these results imply that the best description of the observed data among our choices is the one with the least assumptions on smoothness. We caution that this does not preclude a much better fit with a more sophisticated model, but this simple example illustrates that our framework is effective for MLE even for real observations.

6. Conclusions. The framework for Gaussian process MLE presented in this paper and summarized in Algorithm 1 provides a straightforward method of leveraging hierarchical matrix representations from scientific computing for fast computations with kernelized covariance matrices arising in spatial statistics. The general linear algebraic approach to approximating off-diagonal blocks of the covariance matrix to a specified error tolerance by adaptively determining their ranks gives a flexible way of attaining high-accuracy approximations with reasonable runtimes. A further merit to this approach is that it does not rely on having gridded observations or a translation-invariant covariance kernel.

While in this paper we have focused on MLE for Gaussian processes, these methods are equally viable for the Bayesian setting. For example, computing maximum a posteriori estimates follows essentially the same approach with the addition of a term depending on the prior. Further, sampling from the posterior distriution of θ in a Bayesian setting can be accomplished using standard Markov chain Monte Carlo methods based on quickly evaluating the likelihood and posterior. This can also be combined with Remark 8 for a fully Bayesian treatment.

Our numerical results in section 5 show that our framework scales favorably when applied to our two test cases (the rational quadratic and Matérn family kernels), leading to runtimes scaling approximately as $O(n^{3/2})$ with n the number of observations. Further, we see that the tolerance parameter ϵ_{peel} controlling the rank of off-diagonal block approximations in the peeling algorithm serves as a good estimate of the order of the error in the ultimate trace approximation as well. In practice, the tolerances ϵ_{peel} and ϵ_{fact} can be dynamically modified during the course of the maximum likelihood process for performance, e.g., one could use relatively low-accuracy approximations during initial iterations of the optimization routine and slowly decrease the tolerance as the optimization progresses.

While the methods and complexity estimates discussed in this paper relate to the case of two spatial dimensions, they trivially extend to one-dimensional (time-series) data or quasi-two-dimensional data, e.g., observations in three-dimensions where the sampling density in one dimension is much smaller than in the other two. While the

same methods apply in principle to truly three-dimensional data, the corresponding computational complexity is bottlenecked by the cost of using peeling to obtain a high-accuracy trace estimate of the matrices $\Sigma^{-1}\Sigma_i$ for $i = 1, \ldots, p$ due to increased rank growth. In fact, even in the two-dimensional case it is clear from Tables 5 and 6 that the most expensive piece of our framework in practice is determining these traces. One solution is to instead use the hierarchical matrix representations inside of an estimator such as that of Stein, Chen, and Anitescu [40], which obviates the need for the trace. For true MLE, however, future work on efficiently computing this trace to high accuracy is necessary. Given a method for efficiently computing this trace for three-dimensional data, we expect that related factorizations based on more sophisticated use of skeletonization should give complexities for computing the log-likelihood and gradient that are as good as or better than those we obtain with recursive skeletonization in the two-dimensional case. For example, the hierarchical interpolative factorization [25] (which uses further levels of compression to mitigate rank growth of off-diagonal blocks) may be used in our framework as an efficient method of applying Σ_i and Σ^{-1} and computing the log-determinant of Σ for three-dimensional problems.

While Gaussian process regression is widely used for data in $\mathbb{R}^d$ with d much larger than three, the methods of this paper are designed with spatial data in mind. In particular, in the high-dimensional setting the geometry of the observations becomes very important for efficiency. If the data can be well-approximated according to an intrinsic low-dimensional embedding that is efficient to identify, there is hope for efficient approximations using hierarchical rank structure (see, for example, Yu, March, and Biros [48]). However, in general we expect that rank-structured factorizations will continue to be most effective for low-dimensional spatial applications.

Acknowledgments. The authors thank Matthias Cremon, Eileen Martin, Sven Schmit, and Austin Benson for useful discussion on Gaussian process regression, the anonymous reviewers for thoughtful comments that improved the presentation of this paper, and Stanford University and the Stanford Research Computing Center for providing computational resources and support that have contributed to these research results.

REFERENCES

[1] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, *Fast direct methods for Gaussian processes*, IEEE Trans. Pattern Anal. Mach. Intell., 38 (2016), pp. 252–265.

[2] M. Anitescu, J. Chen, and L. Wang, *A matrix-free approach for solving the parametric Gaussian process maximum likelihood problem*, SIAM J. Sci. Comput., 34 (2012), pp. A240–A262.

[3] E. Aune, D. P. Simpson, and J. Eidsvik, *Parameter estimation in high dimensional Gaussian distributions*, Statist. Comput., 24 (2014), pp. 247–263, https://doi.org/10.1007/s11222-012-9368-y.

[4] S. Börm and J. Garcke, *Approximating Gaussian processes with $\mathcal{H}^2$-matrices*, in Proceedings of the 18th European Conference on Machine Learning, Springer, Berlin, 2007, pp. 42–53.

[5] J. E. Castrillón-Candás, M. G. Genton, and R. Yokota, *Multi-level restricted maximum likelihood covariance estimation and kriging for large non-gridded spatial datasets*, Spat. Stat., 18A (2015), pp. 105–124.

[6] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for HSS representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 67–81.

[7] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622.

[8] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404.
[9] E. Corona, P.-G. Martinsson, and D. Zorin, *An $O(N)$ direct solver for integral equations on the plane*, Appl. Comput. Harmon. Anal., 38 (2015), pp. 284–317.
[10] N. Cressie and G. Johannesson, *Fixed rank kriging for very large spatial data sets*, J. R. Stat. Soc. Ser. B Stat. Methodol., 70 (2008), pp. 209–226.
[11] J. Eidsvik, B. A. Shaby, B. J. Reich, M. Wheeler, and J. Niemi, *Estimation and prediction in spatial models with block composite likelihoods*, J. Comput. Graph. Statist., 23 (2014), pp. 295–315, https://doi.org/10.1080/10618600.2012.760460.
[12] R. Furrer, M. G. Genton, and D. Nychka, *Covariance tapering for interpolation of large spatial datasets*, J. Comput. Graph. Statist., 15 (2006), pp. 502–523.
[13] A. Gillman, P. M. Young, and P.-G. Martinsson, *A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains*, Front. Math. China, 7 (2012), pp. 217–247.
[14] T. Gneiting, *Strictly and non-strictly positive definite functions on spheres*, Bernoulli, 19 (2013), pp. 1327–1349, https://doi.org/10.3150/12-BEJSP06.
[15] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, *Fast direct solvers for integral equations in complex three-dimensional domains*, Acta Numer., 18 (2009), pp. 243–275.
[16] L. Greengard and V. Rokhlin, *On the numerical solution of two-point boundary value problems*, Comm. Pure Appl. Math., 44 (1991), pp. 419–452.
[17] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I*: Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.
[18] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer Ser. Comput. Math., Springer, Berlin, 2015.
[19] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35.
[20] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part* II*: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.
[21] W. Hackbusch, B. N. Khoromskij, and W. Kriemann, *Hierarchical matrices based on a weak admissibility criterion*, Computing, 73 (2004), pp. 207–243.
[22] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[23] M. Heaton, M. Katzfuss, C. Berrett, and D. W. Nychka, *Constructing valid spatial processes on the sphere using kernel convolutions*, Environmetrics, 25 (2014), pp. 2–15, https://doi.org/10.1002/env.2251.
[24] K. L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34 (2012), pp. A2507–A2532.
[25] K. L. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: Integral equations*, Comm. Pure Appl. Math., 69 (2015), pp. 1415–1451.
[26] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Comm. Statist. Simulation Comput., 19 (1990), pp. 433–450.
[27] M. Jun and M. L. Stein, *Nonstationary covariance models for global data*, Ann. Appl. Stat., 2 (2008), pp. 1271–1289, https://doi.org/10.1214/08-AOAS183.
[28] B. N. Khoromskij, A. Litvinenko, and H. G. Matthies, *Application of hierarchical matrices for computing the Karhunen–Loève expansion*, Computing, 84 (2008), pp. 49–67.
[29] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087.
[30] F. Lindgren, H. Rue, and J. Lindström, *An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach*, J. R. Stat. Soc. Ser. B Methodol., 73 (2011), pp. 423–498, https://doi.org/10.1111/j.1467-9868.2011.00777.x.
[31] P.-G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comput. Phys., 205 (2005), pp. 1–23.
[32] G. Matheron, *Principles of geostatistics*, Econom. Geol., 58 (1963), pp. 1246–1266.
[33] V. Minden, K. L. Ho, A. Damle, and L. Ying, *A recursive skeletonization factorization based on strong admissibility*, Multiscale Model. Simul., 15 (2017), pp. 768–796, https://doi.org/10.1137/16M1095949.
[34] E. Porcu, M. Bevilacqua, and M. G. Genton, *Spatio-temporal covariance and cross-covariance functions of the great circle distance on a sphere*, J. Amer. Statist. Assoc., 111 (2016), pp. 888–898, https://doi.org/10.1080/01621459.2015.1072541.

[35] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212–13217.
[36] H. Sang and J. Z. Huang, *A full-scale approximation of covariance functions for large spatial data sets*, J. Roy. Stat. Soc. Ser. B Stat. Methodol., 74 (2012), pp. 111–132.
[37] P. Starr and V. Rokhlin, *On the numerical solution of two-point boundary value problems* II, Comm. Pure Appl. Math., 47 (1994), pp. 1117–1159.
[38] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*, Springer Ser. Statist., Springer, New York, 1999.
[39] M. L. Stein, J. Chen, and M. Anitescu, *Difference filter preconditioning for large covariance matrices*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 52–72.
[40] M. L. Stein, J. Chen, and M. Anitescu, *Stochastic approximation of score functions for Gaussian processes*, Ann. Appl. Stat., 7 (2013), pp. 1162–1191.
[41] M. L. Stein, Z. Chi, and L. J. Welty, *Approximating likelihoods for large spatial data sets*, J. Roy. Stat. Soc. Ser. B Stat. Methodol., 66 (2004), pp. 275–296.
[42] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal., 3 (2011), pp. 115–126.
[43] J. Vanhatalo, V. Pietiläinen, and A. Vehtari, *Approximate inference for disease mapping with sparse Gaussian processes*, Stat. Med., 29 (2010), pp. 1580–1607.
[44] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, J. Roy. Stat. Soc. Ser. B Stat. Methodol., 50 (1988), pp. pp. 297–312.
[45] P. Whittle, *On stationary processes in the plane*, Biometrika, 41 (1954), pp. pp. 434–449.
[46] S. D. Woodruff, S. J. Worley, S. J. Lubker, Z. Ji, J. E. Freeman, D. I. Berry, P. Brohan, E. C. Kent, R. W. Reynolds, S. R. Smith, and C. Wilkinson, *ICOADS release* 2.5: *Extensions and enhancements to the surface marine meteorological archive*, Int. J. Climatol., 31 (2011), pp. 951–967.
[47] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), p. 953–976.
[48] C. D. Yu, W. B. March, and G. Biros, *An n log n parallel fast direct solver for kernel matrices*, in Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, Piscataway, NJ, 2017, pp. 886–896, https://doi.org/10.1109/IPDPS.2017.10.