An Inversion-Free Estimating Equation Approach for Gaussian Process Models*

Mihai Anitescu† Jie Chen† Michael Stein‡

February 7, 2014

Abstract

Gaussian processes are a common analysis tool in statistics and uncertainty quantification. The covariance function of the process is generally unknown and often assumed to fall into some parameteric class. One of the scalability bottlenecks for the large-scale usage of these processes is the computation of the maximum likelihood estimates of the parameters of the covariance matrix. In a classical approach this requires a Cholesky factorization of the dense covariance matrix for each optimization iteration. Recent approaches with stochastic approximations of the score equations [1, 30, 29] require solving linear systems only with the covariance matrix, which is a significant improvement but continues to be a nontrivial expense. In this work, we present an estimating equation approach for the maximum likelihood estimation of parameters. The distinguishing feature of this approach is that no linear system needs to be solved with the covariance matrix. As a result, this approach requires only a small fraction of the computational effort of maximum likelihood calculations; for certain commonly used covariance models and data configurations, this approach results in fast and scalable calculations. We prove that when the covariance matrix has a bounded condition number, our approach has the same convergence rate as does maximum likelihood in that the Godambe information matrix of the resulting estimator is at least as large as a fixed fraction of the Fisher information matrix. Moreover, our approach presents additional advantages compared with the previous ones [1, 30, 29], namely, the preservation of an optimization structure and the guarantee of finding global optima for covariance models that are linear in the parameters. We demonstrate the effectiveness of the proposed approach on two synthetic examples, one of which involes up to 1 million data points.

*Preprint ANL/MCS-P5078-0214, Argonne National Laboratory.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. Emails: (anitescu,jiechen)@mcs.anl.gov

‡Department of Statistics, University of Chicago, Chicago, IL 60637. Email: stein@galton.uchicago.edu.

1 Introduction

Gaussian processes have been widely used throughout the statistical and machine learning communities for modeling of natural processes, for regression and classification problems, for uncertainty quantification, and for interpolation in parameter space for computer model output. Their widespread use is in part because they provide conceptual and computational advantages for processes that are a function of a continuous index, and in part because they can be used as a building block for modeling non-Gaussian processes [14, 25].

1.1 Likelihood Function and Its Computational Challenges

In most applications, the covariance structure of such a Gaussian process is at least partially unknown and must be estimated from the available data. Assuming we are willing to specify the covariance structure up to some parameter $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$, the generic problem we are faced with is computing the loglikelihood for $\boldsymbol{y} \sim N(\mathbf{0}, K(\boldsymbol{\theta}))$ for some random vector $\boldsymbol{y} \in \mathbb{R}^n$ and K an $n \times n$ positive definite matrix parameterized by the unknown $\boldsymbol{\theta}$. Since unknown mean parameters generally pose far fewer difficulties than do covariance parameters, we will assume that the mean is known to be $\mathbf{0}$ throughout this work.

The loglikelihood is then, up to an additive constant, given by

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T K(\boldsymbol{\theta})^{-1}\boldsymbol{y} - \frac{1}{2}\log\det\{K(\boldsymbol{\theta})\}. \tag{1}$$

If K has no exploitable structure, the standard direct way of calculating $\mathcal{L}(\boldsymbol{\theta})$ is to compute the Cholesky decomposition of $K(\boldsymbol{\theta})$, which then allows $\boldsymbol{y}^T K(\boldsymbol{\theta})^{-1}\boldsymbol{y}$ and $\log\det\{K(\boldsymbol{\theta})\}$ to be computed quickly. However, the Cholesky decomposition generally requires $O(n^2)$ storage and $O(n^3)$ operations, both of which can be prohibitive for sufficiently large n.

If our goal is just to find the maximum likelihood estimate (MLE), we can avoid computing the log determinant by considering the score equations, which are obtained by setting the gradient of the loglikelihood equal to zero. Specifically, defining $K_i(\boldsymbol{\theta}) = \frac{\partial}{\partial\theta_i}K(\boldsymbol{\theta})$, we can express the score equations for $\boldsymbol{\theta}$ by (suppressing the dependence of K on $\boldsymbol{\theta}$)

$$\frac{1}{2}\boldsymbol{y}^T K^{-1}K_i K^{-1}\boldsymbol{y} - \frac{1}{2}\mathrm{tr}(K^{-1}K_i) = 0 \tag{2}$$

for $i = 1, \ldots, p$.

The left-hand side of (2) requires linear solves with K (arising from expressions of the form $K^{-1}\boldsymbol{x}$ for a vector $\boldsymbol{x}$). For large-scale problems, these can be done by using iterative methods; in turn, efficiency considerations require multiplying arbitrary vectors by K rapidly. Exact matrix-vector multiplication generally requires $O(n^2)$ operations, but if the process is stationary and the data form a regular grid (or a subset of the grid), then it can be done in $O(n \log n)$ operations by using circulant embedding followed by fast Fourier transform [6]; for observations on an irregular grid, fast multipole approximations can be used [1, 9]. If an iterative method is used for linear solves, the number of iterations required

is related to the condition number of K (the ratio of the largest to smallest singular value), thus making preconditioning [10] essential; see [29] for some circumstances under which one can prove that preconditioning works well. Computing the first term in (2) requires only one solve in K; however, the trace term requires n solves (one for each column of K_i) for each i, a prohibitive requirement when n is large.

Recently, in [1], we analyzed and demonstrated a stochastic approximation of the trace term based on the Hutchinson trace estimator [21]. To define it, let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ be iid random vectors in $\mathbb{R}^n$ with iid symmetric Bernoulli components, that is, taking values 1 and -1 with equal probability. Define a set of estimating equations for $\boldsymbol{\theta}$ by

$$g_i(\boldsymbol{\theta}, N) = \frac{1}{2}\boldsymbol{y}^T K^{-1} K_i K^{-1} \boldsymbol{y} - \frac{1}{2N} \sum_{j=1}^{N} \boldsymbol{u}_j^T K^{-1} K_i \boldsymbol{u}_j = 0 \tag{3}$$

for $i = 1, \ldots, p$. In [30], we showed that if $K(\boldsymbol{\theta})$ is well-conditioned, then one can estimate $\boldsymbol{\theta}$ with nearly the same statistical efficiency as the exact maximum likelihood estimates with fairly small values of N ($N \approx 50$ often appears to be adequate). In [29] we showed that optimal preconditioning is achievable for certain classes of Matérn kernels, in the sense that the preconditioned matrix has a bounded condition number independent of the number of observations. Subsequently, we extended this approach to observations on an irregular grid [4].

The approach (3) ensures that the number of vectors $\boldsymbol{x}$ for which one needs to compute $K^{-1}\boldsymbol{x}$ is small and that the number of matrix-vector multiplications needed to solve these linear equations with an iterative method is also small. Nevertheless, if K is dense, then each matrix-vector multiplication is generally an $O(n^2)$ operation. We have demonstrated faster calculations, $O(n \log n)$, for the cases of regular grid and irregular grid in [6, 7] and [9, 8], respectively, by using the aforementioned fast Fourier transforms and fast multipole approximations. The demonstrated calculations scale to matrices of size up to $n = 10^9$ on $O(10^3)$ CPU cores with good parallel efficiency.

The combination of these results [1, 30, 29, 6, 7, 9, 8] has allowed us to scalably perform the overall Gaussian process analysis for up to $O(10^9)$ data points. We have developed a software package `ScalaGAUSS` (available at `http://press3.mcs.anl.gov/scala-gauss/`) that implements this development in two versions, one written in Matlab and one in C++.

1.2 Comparison with Other Approaches

Several other approaches have been developed to accommodate the computational bottlenecks of carrying out the maximum likelihood calculations for the loglikelihood function (1).

Along the vein of our previous work, [1, 30, 29, 6, 7, 9, 8], one can approximate $\log(\det(K)) = \mathrm{tr}(\log(K))$ by using a power series approximation to $\log K$ followed by a stochastic approximation of the trace [35]. The examples presented by [35] show that the approach does not generally provide a good approximation to the loglikelihood, however.

Several other approaches have been proposed for approximating maximum likelihood estimates for large-scale Gaussian processes. However, as opposed to the algorithms based on (3), the approximations generally cannot come arbitrarily close to the exact maximum likelihood estimates. Spectral approximations of the likelihood can be fast and accurate for gridded data [34, 18, 13], but they have difficulty at increasing dimensions; and the performance of such methods is less attractive for ungridded data [16].

Low-rank approximations, in which the covariance matrix is approximated by a low-rank matrix plus a diagonal matrix, can be efficient computationally [12, 15], but they work poorly if the diagonal component of the covariance matrix does not dominate the higher frequency variations in the observations [27].

Covariance tapering replaces the covariance matrix of interest by a sparse covariance matrix with similar local behavior [17]. However, the tapered covariance matrix must be very sparse for computational efficiency, which in turn affects the method accuracy [28].

Composite likelihood methods [33, 31, 3] based on conditional Gaussian decompositions and approximations can in principle approach the maximum likelihood estimator. To increase the accuracy, however, one must condition on large data subsets, an action that again requires Cholesky factorizations of large covariance matrices.

1.3 This Work

While the approach based on sample average approximation (3) has allowed us to solve scalably large parameter estimation problems for Gaussian process, it still presents several difficulties. The approach is based on nonlinear equations for which it is hard to guarantee that we can find a solution, as opposed to finding at least local minima of optimization problems. This difficulty is aggravated by the fact that likelihood surfaces can be quite complex (see an example in [30]). In addition, solving (3) requires several linear system solves per each nonlinear iteration. Thus, it is worth inquiring whether good alternatives exist that may require less calculation per step.

In this work, we investigate an approach based on new unbiased estimating equations. The approach solves an optimization problem, the computation of which does not require linear solves with the covariance matrix.

The paper is organized as follows. In §2 we introduce the new estimating equations that like (3) involve computing the trace of a matrix but unlike (3) do not involve the inverse covariance matrix. We show that if the condition number of K is bounded, then the statistical efficiency of this approach is within a fixed factor of the standard one. In §3 we discuss computational and global convergence aspects of the new approach in comparison with existing methods. In §4 we present numerical results that demonstrate the robustness and accuracy of the approach on two examples, including one involving more than 1 million data points.

2 Estimating Equations

Using the notation introduced in the preceding section, we have

$$\mathbb{E}\left[\boldsymbol{y}^T K_i \boldsymbol{y}\right] = \mathbb{E}\left[\mathrm{tr}\left(\boldsymbol{y}^T K_i \boldsymbol{y}\right)\right] = \mathbb{E}\left[\mathrm{tr}\left(K_i \boldsymbol{y}\boldsymbol{y}^T\right)\right] = \mathrm{tr}\left(K_i \mathbb{E}\left[\boldsymbol{y}\boldsymbol{y}^T\right]\right) = \mathrm{tr}\left(K_i K\right). \tag{4}$$

Thus, we define a system of estimating equations

$$\boldsymbol{g}(\boldsymbol{\theta}) = \mathbf{0}, \tag{5}$$

where each component is defined as

$$g_i(\boldsymbol{\theta}) := \boldsymbol{y}^T K_i(\boldsymbol{\theta}) \boldsymbol{y} - \mathrm{tr}\left(K_i(\boldsymbol{\theta}) K(\boldsymbol{\theta})\right), \quad i = 1, 2, \ldots, p. \tag{6}$$

Based on (4), we immediately see that the estimating equations (5) are unbiased, that is, $\mathbb{E}_{\boldsymbol{\theta}}(\boldsymbol{g}(\boldsymbol{\theta})) = \mathbf{0}$ for all possible $\boldsymbol{\theta}$. In addition, we note that (5) gives the first-order optimality condition of the optimization problem

$$\max_{\boldsymbol{\theta}} \left(\boldsymbol{y}^T K(\boldsymbol{\theta}) \boldsymbol{y} - \frac{1}{2} \mathrm{tr}\left(K^2(\boldsymbol{\theta})\right) \right). \tag{7}$$

In what follows, we derive the Godambe information matrix of the new estimating equations and show the relation with the Fisher information matrix. To simplify notation we, suppress in the rest of the paper the dependence of $K(\boldsymbol{\theta})$, $\mathbb{E}_{\boldsymbol{\theta}}$, and $\mathrm{cov}_{\boldsymbol{\theta}}$ on $\boldsymbol{\theta}$.

Lemma 1. *The estimating function $\boldsymbol{g}$ admits that*

$$\Lambda_{ij} := \mathbb{E}\left(\frac{\partial g_i(\boldsymbol{\theta})}{\partial \theta_j}\right) = -\mathrm{tr}\left(K_i K_j\right) \tag{8}$$

$$\Gamma_{ij} := \mathrm{cov}(g_i(\boldsymbol{\theta}), g_j(\boldsymbol{\theta})) = 2\mathrm{tr}\left(K_i K K_j K\right) \tag{9}$$

for all $i, j = 1, 2, \ldots, p$.

Proof. For (8), we have

$$\frac{\partial g_i}{\partial \theta_j} = \boldsymbol{y}^T K_{ij} \boldsymbol{y} - \mathrm{tr}\left(K_{ij} K\right) - \mathrm{tr}\left(K_{ij}\right),$$

where $K_{ij} = \frac{\partial}{\partial \theta_j} K_i = \frac{\partial}{\partial \theta_i} K_j$. Since

$$\mathbb{E}\left(\boldsymbol{y}^T K_{ij} \boldsymbol{y}\right) = \mathbb{E}\left\{\mathrm{tr}\left(\boldsymbol{y}^T K_{ij} \boldsymbol{y}\right)\right\} = \mathbb{E}\left\{\mathrm{tr}\left(K_{ij} \boldsymbol{y}\boldsymbol{y}^T\right)\right\} = \mathrm{tr}\left(K_{ij} K\right),$$

(8) immediately follows.

For (9), we have $\mathbb{E}(g_i) = 0$, and (4) implies that

$$\begin{aligned}\Gamma_{ij} &= \operatorname{cov}(g_i, g_j) = \mathbb{E}\left(g_i, g_j\right) \\ &= \mathbb{E}\left(\boldsymbol{y}^T K_i \boldsymbol{y}\boldsymbol{y}^T K_j \boldsymbol{y}\right) + \operatorname{tr}\left(K_i K\right)\operatorname{tr}\left(K_j K\right) \\ &\quad - \mathbb{E}\left\{\operatorname{tr}\left(\boldsymbol{y}^T K_j \boldsymbol{y}\right)\operatorname{tr}\left(K_i K\right)\right\} - \mathbb{E}\left\{\operatorname{tr}\left(\boldsymbol{y}^T K_i \boldsymbol{y}\right)\operatorname{tr}\left(K_j K\right)\right\} \\ &= \mathbb{E}\left(\boldsymbol{y}^T K_i \boldsymbol{y}\boldsymbol{y}^T K_j \boldsymbol{y}\right) - \operatorname{tr}\left(K_i K\right)\operatorname{tr}\left(K_j K\right).\end{aligned}$$

Consider now a vector $\boldsymbol{u}$ with components u_i, $i = 1, 2, \ldots, p$; we have

$$\sum_{i,j=1}^{p} u_i u_j \Gamma_{ij} = \mathbb{E}\left(\boldsymbol{y}^T A \boldsymbol{y}\boldsymbol{y}^T A \boldsymbol{y}\right) - \operatorname{tr}\left(AK\right)\operatorname{tr}\left(AK\right), \tag{10}$$

where $A = \sum_{i=1}^{n} u_i K_i$. If we define $\boldsymbol{y} = K^{0.5}\boldsymbol{e}$, then $\boldsymbol{e}$ is a random vector from $\mathcal{N}(\boldsymbol{0}, I_n)$. Thus, the first term in (10) becomes

$$L = \mathbb{E}\left(\boldsymbol{y}^T A \boldsymbol{y}\boldsymbol{y}^T A \boldsymbol{y}\right) = \mathbb{E}\left(\boldsymbol{e}^T \tilde{A} \boldsymbol{e}\boldsymbol{e}^T \tilde{A} \boldsymbol{e}\right), \quad \tilde{A} = K^{0.5} A K^{0.5}.$$

Since A is symmetric, so is $\tilde{A}$, which has the eigenvalue decomposition $\tilde{A} = QDQ^T$. In turn, this results in

$$L = \mathbb{E}\left(\boldsymbol{y}^T A \boldsymbol{y}\boldsymbol{y}^T A \boldsymbol{y}\right) = \mathbb{E}\left(\boldsymbol{e}^T QDQ^T \boldsymbol{e}\boldsymbol{e}^T QDQ^T \boldsymbol{e}\right) = \mathbb{E}\left(\tilde{\boldsymbol{e}}^T D \tilde{\boldsymbol{e}}\tilde{\boldsymbol{e}}^T D \tilde{\boldsymbol{e}}\right),$$

where $\tilde{\boldsymbol{e}} = Q^T \boldsymbol{e}$ is a random vector from $\mathcal{N}(\boldsymbol{0}, I_n)$. We rewrite the last term to obtain

$$L = \mathbb{E}\left(\sum_{i}^{n} d_i \tilde{\boldsymbol{e}}_i^2\right)^2 = \mathbb{E}\left(\sum_{i,j}^{n} d_i d_j \tilde{\boldsymbol{e}}_i^2 \tilde{\boldsymbol{e}}_j^2\right) = \mathbb{E}\left(\sum_{i,j=1, i\neq j}^{n} d_i d_j \tilde{\boldsymbol{e}}_i^2 \tilde{\boldsymbol{e}}_j^2\right) + \mathbb{E}\left(\sum_{i=1}^{n} d_i^2 \tilde{\boldsymbol{e}}_i^4\right).$$

From the properties of the normal distribution, we have that $\mathbb{E}(\tilde{\boldsymbol{e}}_i^2 \tilde{\boldsymbol{e}}_j^2) = \mathbb{E}(\tilde{\boldsymbol{e}}_i^2)\mathbb{E}(\tilde{\boldsymbol{e}}_j^2) = 1$ when $i \neq j$ and $\mathbb{E}(\tilde{\boldsymbol{e}}_i^4) = 3$. Then,

$$L = \sum_{i,j=1, i\neq j}^{n} d_i d_j + 3\sum_{i=1}^{n} d_i^2 = \left(\sum_{i=1}^{n} d_i\right)^2 + 2\sum_{i=1}^{n} d_i^2.$$

By the fact that the d_i's are the eigenvalues of $\tilde{A} = K^{0.5} A K^{0.5}$ we obtain that $\sum_{i=1}^{n} d_i = \operatorname{tr}(\tilde{A}) = \operatorname{tr}(KA)$ and that $\sum_{i=1}^{n} d_i^2 = \operatorname{tr}(\tilde{A}^2) = \operatorname{tr}(AKAK)$. Thus,

$$L = \left(\operatorname{tr}(KA)\right)^2 + 2\operatorname{tr}(KAKA).$$

Substituting L in (10), we obtain

$$\sum_{i=1}^{n} u_i u_j \Gamma_{ij} = 2\mathrm{tr}(KAKA) = 2\sum_{i,j=1}^{n} u_i u_j \mathrm{tr}\left(KK_iKK_j\right).$$

Since this equality holds for all $\boldsymbol{u}$, we conclude (9). □

Now define the matrices

$$\Gamma = \{\Gamma_{ij}\}_{i,j=1}^{p}, \quad \Lambda = \{\Lambda_{ij}\}_{i,j=1}^{p}. \tag{11}$$

Their quadratic forms can be bounded by each other. For this purpose, we first need the following result.

Lemma 2. *Let D be a positive diagonal matrix and M a positive semidefinite matrix. Then* $\mathrm{tr}(DM) \leq \max_i\{d_i\} \cdot \mathrm{tr}(M)$.

Proof. We have that $\mathrm{tr}(DM) = \sum_{i=1}^{n} d_i m_{ii} \leq \max_i\{d_i\}(\sum_{i=1}^{n} m_{ii})$, the last inequality following from the fact that $m_{ii} \geq 0$. We immediate conclude the lemma. □

Lemma 3. *The matrices Γ and $-\Lambda$ are positive semidefinite and satisfy*

$$\Gamma \preceq 2\sigma_M^2(K)(-\Lambda), \tag{12}$$

where $\sigma_M(K)$ denotes the largest eigenvalue of the covariance matrix K and $\preceq$ denotes the positive semidefinite ordering. If Γ is positive definite, then so is $-\Lambda$, and we thus have

$$\Gamma^{-1} \succeq \frac{1}{2\sigma_M^2(K)}(-\Lambda)^{-1}. \tag{13}$$

Proof. Take now a vector $\boldsymbol{u} \in \mathbb{R}^p$ whose entries are $u_1, u_2, \ldots, u_p$. We have

$$\sum_{i,j=1}^{p} u_i u_j \Lambda_{ij} = -\mathrm{tr}(AA)$$

where $A = \sum_{i=1}^{p} u_i K_i$ is a symmetric matrix, which makes A^2 a symmetric positive semidefinite matrix whose trace is nonnegative. We thus obtain $-\Lambda \succeq 0$.

Similarly,

$$\boldsymbol{u}^T\Gamma\boldsymbol{u} = 2\mathrm{tr}(KAKA) = 2\mathrm{tr}(K^{0.5}AKAK^{0.5}) = 2\mathrm{tr}[(K^{0.5}AK^{0.5})^2] \geq 0,$$

which proves that $\Gamma \succeq 0$.

Define the eigenvalue decomposition of $K = QDQ^T$, with $D \succeq 0$ and $\max_i\{d_i\} = \sigma_M(K)$. We have

$$\mathrm{tr}(KAKA) = \mathrm{tr}(QDQ^TAKA) = \mathrm{tr}(DQ^TAKAQ) \leq \sigma_M(K)\mathrm{tr}(Q^TAKAQ), \tag{14}$$

which follows from Lemma 2. One step further,

$$\mathrm{tr}(Q^TAKAQ) = \mathrm{tr}(AKA) = \mathrm{tr}(KA^2) \leq \sigma_M(K)\mathrm{tr}(A^2).$$

Hence, we obtain

$$\boldsymbol{u}^T\Gamma\boldsymbol{u} = 2\mathrm{tr}(KAKA) \leq 2\sigma_M^2(K)\mathrm{tr}(A^2) = -2\sigma_M^2(K)\boldsymbol{u}^T\Lambda\boldsymbol{u},$$

which proves (12).

For the second part, if Γ is positive definite, then it is invertible. Thus, so must be Λ. Applying positive definite ordering [20, Corollary 7.7.4] to (12) yields (13). □

We now consider the Fisher information matrix $\mathcal{I}$ attached to the likelihood function (1), with $\mathcal{I}_{ij} = \frac{1}{2}\mathrm{tr}\left(K^{-1}K_iK^{-1}K_j\right)$. We have the following result.

Lemma 4. *Let σ_m denote the smallest eigenvalue of K. Then,*

$$\mathcal{I} \preceq \frac{1}{2\sigma_m^2(K)}(-\Lambda).$$

Proof. We take an abitrary vector $\boldsymbol{u} \in \mathbb{R}^p$ and form

$$\boldsymbol{u}^T\mathcal{I}\boldsymbol{u} = \sum_{i,j=1}^{p} u_iu_j\frac{1}{2}\mathrm{tr}\left(K^{-1}K_iK^{-1}K_j\right) = \frac{1}{2}\mathrm{tr}\left(K^{-1}AK^{-1}A\right),$$

where A is defined in the proof of the preceding lemma. Using the eigenvalue decomposition $K = QDQ^T$, we obtain

$$\boldsymbol{u}^T\mathcal{I}\boldsymbol{u} = \frac{1}{2}\mathrm{tr}\left(QD^{-1}Q^TAK^{-1}A\right) = \frac{1}{2}\mathrm{tr}\left(D^{-1}Q^TAK^{-1}AQ\right) \leq \frac{1}{2\sigma_m(K)}\mathrm{tr}\left(AK^{-1}A\right),$$

where the last inequality follows from Lemma 2. Applying this lemma again we have

$$\boldsymbol{u}^T\mathcal{I}\boldsymbol{u} \leq \frac{1}{2\sigma_m(K)}\mathrm{tr}\left(AK^{-1}A\right) = \frac{1}{2\sigma_m(K)}\mathrm{tr}\left(K^{-1}A^2\right) \leq \frac{1}{2\sigma_m^2(K)}\mathrm{tr}\left(A^2\right).$$

By using a result in the proof of the preceding lemma, $\mathrm{tr}\left(A^2\right) = \boldsymbol{u}^T(-\Lambda)\boldsymbol{u}$, we obtain

$$\boldsymbol{u}^T\mathcal{I}\boldsymbol{u} \leq \frac{1}{2\sigma_m^2(K)}\boldsymbol{u}^T(-\Lambda)\boldsymbol{u}.$$

This inequality concludes the proof. □

We are now in a position to make a claim about the statistical efficiency of the estimating equations (5). For such equations, the statistical efficiency is governed by the Godambe

information matrix [32]

$$\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta})) = \mathbb{E}(\nabla_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta}))[\mathrm{cov}(\boldsymbol{g}(\boldsymbol{\theta}))]^{-1}\mathbb{E}(\nabla_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})). \tag{15}$$

Under certain assumptions, its inverse approaches the covariance matrix of the estimate produced by (5). As estimating equations, under some regularity conditions, the score equations (2) are optimal in that their Godambe information matrix is the Fischer information matrix $\mathcal{I}$ and that $\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta})) \preceq \mathcal{I}$ for any unbiased estimating equations $\boldsymbol{g}(\boldsymbol{\theta}) = \mathbf{0}$ [2]. Thus, comparing $\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta}))$ with $\mathcal{I}$ gives a measure of the statistical efficiency of (5). The following result lower bounds $\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta}))$.

Theorem 1. *The Godambe information matrix* (15) *for the unbiased estimating equations* (5) *satisfies*

$$\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta})) \succeq \frac{1}{\mathrm{cond}(K)^2}\mathcal{I},$$

where $\mathrm{cond}(K) = \sigma_M(K)/\sigma_m(K)$ *is the condition number of* K*.*

Proof. From Lemma 1 we have

$$\mathcal{E}(\boldsymbol{g}(\boldsymbol{\theta})) = \Lambda\Gamma^{-1}\Lambda.$$

Using Lemmas 3 and 4, we obtain

$$\Lambda\Gamma^{-1}\Lambda \succeq \frac{1}{2\sigma_M^2(K)}\Lambda(-\Lambda)^{-1}\Lambda = \frac{1}{2\sigma_M^2(K)}(-\Lambda) \succeq \frac{2\sigma_m^2(K)}{2\sigma_M^2(K)}\mathcal{I},$$

which proves the claim. □

3 Discussion

The value of solving the optimization problem (7) for estimating the Gaussian process parameters $\boldsymbol{\theta}$ is that the calculation of the objective function and its gradient (6) *does not require solving linear systems with the covariance matrix* K, as opposed to maximizing the likelihood (1), solving the score equations (2), and our previous work (3). Moreover, not only are the resulting estimating equations unbiased, but also, by Theorem 1, when the condition number of the covariance matrix is uniformly bounded as the number of data points increases, the Godambe information matrix is bounded below by a fixed factor of the Fisher information matrix. Therefore, the statistical accuracy of the estimating equations (5) is of the same order as the optimal one of the maximum likelihood estimator. Specifically, if we use our estimator (5) for M independent samples from the Gaussian process, then the ratio of the norm of the asymptotic variance to the one of the maximum likelihood variance as $M \to \infty$ is the same as the ratio between the inverses of the Godambe information matrix and the Fisher information matrix [32]. Thus, by Theorem 1 this ratio is bounded independent of n. Asymptotic statements for one sample but increasing n are more difficult to make even for the maximum likelihood estimator alone [26], but the relationship between

the Godambe information matrix and the Fischer information matrix is still a good measure of the relative statistical efficiency of the two methods [30].

For the estimating equation approach (5), even a condition number slowly increasing with n may result in a statistically and computationally efficient method (see §4.2, in particular, Table 4). When we can efficiently precondition K, resulting in a small condition number of the covariance matrix of the transformed data, the approach based on (5) has substantial appeal. We have demonstrated an efficient preconditioning for the case of the Matérn process class using Laplacian filtering [29, 4], which indicates that the method proposed in this work offers potential to be both accurate and efficient. We now discuss some numerical issues.

3.1 Computational Considerations

There are several classes of problems for which (7), and implicitly the estimating equations (5), can be solved efficiently, at least in cost per gradient computation of (7).

- If K is sparse for any $\boldsymbol{\theta}$, then the gradient of (7) can be computed in $O(n)$. We present one such example in §4.1. In this case, computing the likelihood (2) and its gradient directly may be competitive only when the Cholesky factorization can be carried out in time comparable to $O(n)$, which, while is true for the example in §4.1, is hard to guarantee in general. We also note that this setting may appear not only for Gaussian process models but also for certain large-scale generalized linear models when mixed effects are used [22].

- If the data is on a regular grid and the kernel is stationary, that is, the kernel function $k(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = k(\boldsymbol{x} - \boldsymbol{y}; \boldsymbol{\theta})$, then matrix-vector multiplications can be carried out in $O(n \log n)$ by means of circulant embedding and fast Fourier transform. Moreover, as we show in the appendix, the trace of the product of two Toeplitz matrices, as in KK_i, can be computed in $O(n)$ time. Thus, both the function evaluation of (7) and the gradient evaluation (6) can be computed in $O(n \log n)$ time. We present such an example in §4.2.

- If the data is not on a regular grid but the kernel is smooth away from the origin, then computing matrix-vector products can be done in $O(n \log n)$ time by using treecode methods, as we have developed and demonstrated for the Matérn kernel [1, 9].

For the last two cases, maximum likelihood estimation would require the Cholesky factorization of a dense matrix, which is prohibitive on large data sets. Moreover, for the approach proposed here, no linear system with K needs to be solved, which puts the method in this paper at an advantage over our previous work [1, 30, 29, 6, 9].

Since the number of parameters p is generally much smaller than n, the fact that the objective and the gradient of the optimization problem (7) can be efficiently computed suggests quasi-Newton approaches for solving (7).

3.2 Global Convergence

We can determine a solution to the estimating equation (5) by solving the associated optimization problem (7). Since optimization algorithms [23] can be guaranteed under fairly mild conditions to be globally convergent in the sense of convergence to a stationary point and since the latter satisfies the unbiased estimating equations (5), the optimization approach may be advantageous over the nonlinear equation approach.

An additional advantage of optimization is that in a nonlinear equation approach, the parameter $\boldsymbol{\theta}$ is no longer intrinsically constrained on the domain where K is positive definite (which is done by the fact that the log-det term becomes singular in (1) and acts as a penalty for K leaving the $K \succ 0$ region). While we have not encountered this problem in our approach, it can certainly in principle become a difficulty. Explicit constraints on $\boldsymbol{\theta}$ are not difficult to impose, particularly in an optimization setup; but the KKT optimality conditions would involve complementarity constraints [23] and would no longer be exactly (5). For example, in the one-dimensional case, when fitting an exponential kernel $k(x, y; \boldsymbol{\theta}) = \theta_1 \exp(-|x - y|/\theta_2)$, one would require $\theta_1, \theta_2 \geq 0$.

Nevertheless, important examples exist for which the problem can be stated so as not to pose significant difficulties. First, one can use reparameterization to remove constraints, such as replacing a parameter $\theta \geq 0$ by a free parameter $\tilde{\theta}$ with $\tilde{\theta}^2 = \theta$.

Second, there is a class of covariance models where one has that K is linear in the parameter $\boldsymbol{\theta}$, as occurring, for example, when taking linear combinations of fixed covariance models [25, §4.2.4] with to-be-determined coefficients. We also note that this situation appears for generalized linear models that include stochastic or mixed effects and cases when the parameters are the squared variances [22]. The resulting optimization problem (7) is then a concave maximization, and we are guaranteed to get a globally optimal solution by solving one linear system of equations, of size $p \times p$ only, with respect to $\boldsymbol{\theta}$. This by itself points to a feature of our approach that maximum likelihood does not exhibit: for a fairly large class of models (where K is linear in $\boldsymbol{\theta}$) we can obtain a global solution by using a local algorithm. In effect, any kernel can in principle be approximated to belong to such a class, at the cost of a large number of "basis" covariance matrices with which the linear combination is formed. As we demonstrate in §4.1, the likelihood function will not be convex for this model class; while we cannot prove that the likelihood function has other stationary points, it seems difficult to guarantee that maximum likelihood can find the global optimum, since solving (5) would do in this case. Of course, if the linear model is not a good model, then solving either problem could result in estimates for which the covariance function is even not positive definite; however, in this case, the uniqueness of the solution to (5) will point out the need for refining the problematic model.

4 Numerical Experiments

We first investigate the benefit of using (5) in a simple setting, where the covariance matrix K depends linearly on its parameters. In this case, the estimating equations (5) are linear

and can thus be solved with one $p \times p$ small linear system. For likelihood maximizations, on the other hand, one would have to solve the likelihood problem iteratively, a considerably more complicated endeavor. In this case our method shares some features with the MINQUE method as applied to random effects models [24, 19]. Indeed, both methods are based on setting quadratic forms of the data to their expected value, and both result in estimating equations that are linear in their parameters, although the quadratic forms used here and their theoretical guarantees are different. We next perform large-scale experiments to showcase the superiority of the optimization formulation (7) in a nonlinear setting. These experiments also demonstrate a better scalability in numerical calculations compared with our previous approach (3) that requires the solution of linear systems of size $n \times n$.

4.1 An Example with Linear Parametric Dependence of the Covariance

Consider the covariance matrix $K = \theta_1 I_n + \theta_2 L_n$, where I_n is the identity matrix and L_n is the 2D Laplacian matrix, both of size $n \times n$. We simulated data from this distribution with $\boldsymbol{\theta} = [3; 2]$ and we used (5) to estimate $\boldsymbol{\theta}$ and compare it with the true value $[3; 2]$. We note that for $\boldsymbol{\theta} \in [\epsilon, T] \times [0, T]$, where $\epsilon, T > 0$ are fixed parameters, the condition number of K is uniformly bounded, independent of the dimension n. We are thus in the regime posited by Theorem 1, and we therefore expect that the accuracy of the estimators produced by solving (5) is quite good.

An interesting issue is in comparing the complexity of (5) with the likelihood approach. For $n = 100$ we sampled from the distribution attached to the covariance matrix $K([1; 2])$ and we computed the likelihood at $[1; \theta_2]$ for varying values of θ_2. We display the loglikelihood and its second derivative in Figure 1. The loglikelihood has a minimum very close to 2, as expected. On the other hand, we see that the second derivative is negative for a large range and eventually approaches singularity. This situation would likely create nontrivial challenges for optimization-based formulations that employ Newton-type algorithms. Moreover, there is no obvious simplifying structure here: no convexity or low-degree polynomial structure. For comparison, the estimating equations (5) are linear; and from Theorem 1 we expect them to have comparable statistical power with maximum likelihood.

Table 1: Estimators from exact trace computation estimator (5) for the model $K(\boldsymbol{\theta}) = \theta_1 I_n + \theta_2 L_n$. Truth: $\theta_1 = 3$ and $\theta_2 = 2$. Standard deviations are computed from $M = 100$ repeated experiments.

n	100	1000	10000
$\hat{\theta}_1 \pm \sigma_1$	3.046±0.783	2.964±0.278	3.004±0.076
$\hat{\theta}_2 \pm \sigma_2$	2.024±0.543	2.010±0.191	1.991±0.049

We generated $M = 100$ random realizations for $n = 100$, $n = 1000$, and $n = 10000$ for $\boldsymbol{\theta} = [3; 2]$, and we used (5) to compute estimate $\hat{\boldsymbol{\theta}}$. The mean and standard deviation across the M experiments are displayed in Table 1. We note that the estimates converge to their

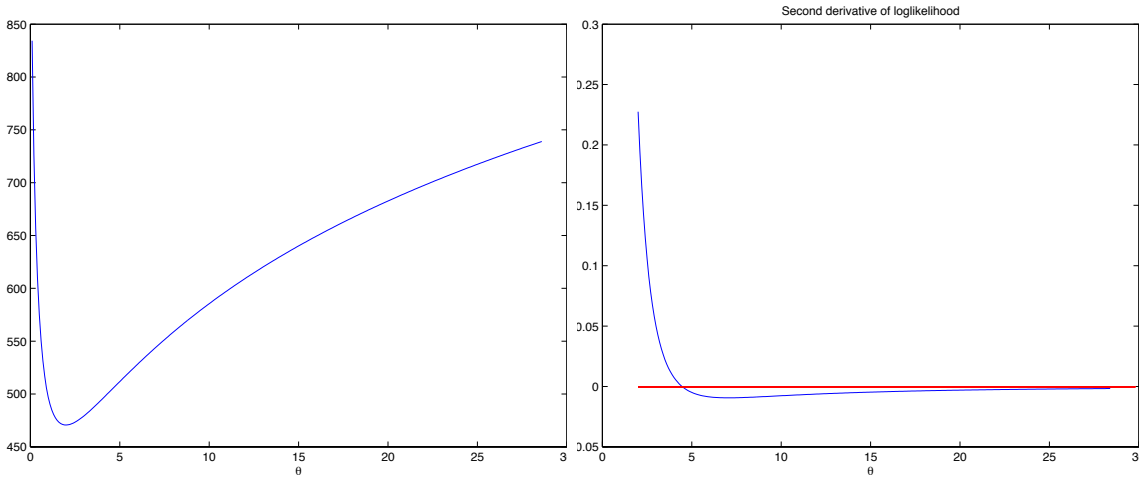

Figure 1: Loglikelihood and its second-order divided differences for a simulated data set with $n = 100$.

true values and the variance for both cases behaves close to $n^{-0.5}$. Moreover, the number of operations to set up the equations is $O(n)$, since I_n and L_n are sparse, (5) is linear in $\boldsymbol{\theta}$, and they need be set up only once. Note that it would be hard to guarantee a priori that the maximum likelihood estimates be obtained by using $O(n)$ operations, as is the case for the method developed in this paper on this particular example, because the number of nonlinear iterations for solving maximum likelihood problem would be hard to bound.

4.2 Large-Scale Experiments with Power-law Kernel

We simulate data with the d-dimensional power-law kernel ($d = 2$)

$$G(\boldsymbol{x}; \boldsymbol{\theta}) = \begin{cases} \Gamma(-\alpha/2) r^{\alpha}, & \text{if } \alpha/2 \notin \mathbb{N} \\ (-1)^{1+\alpha/2} r^{\alpha} \log r, & \text{if } \alpha/2 \in \mathbb{N}, \end{cases} \tag{16}$$

where $\boldsymbol{x} = [x_1; \ldots; x_d] \in \mathbb{R}^d$ denotes coordinates, $\boldsymbol{\ell} = [\ell_1; \ldots; \ell_d] \in \mathbb{R}^d$ denotes scales, and r is the elliptical radius

$$r = \sqrt{\frac{x_1^2}{\ell_1^2} + \cdots + \frac{x_d^2}{\ell_d^2}}. \tag{17}$$

Hence, $\boldsymbol{\theta} = [\ell_1; \ldots; \ell_d; \alpha]$ consists of all the fitting parameters. The function G is conditionally positive definite; therefore, only the covariances of authorized linear combinations of G are defined [11, Sec. 4.3]. Thus, we perform filtering to ensure positive definiteness. We note that this requires an a priori upper bound on α. Stein et al. [29] show that if $\alpha + d \in 4\mathbb{N}$, then performing $\tau = (\alpha + d)/4$ times filtering on the process will yield a covariance matrix with bounded condition number independent of the matrix size. In what follows, the matrix

K represents the covariance matrix of the filtered process.

We use the method of [5] to simulate a random realization of the filtered process. The method computes $\boldsymbol{y} = K^{\frac{1}{2}}\boldsymbol{z}$ for a random vector $\boldsymbol{z} \sim N(\boldsymbol{0}, I)$ in a matrix-free style (i.e., without any factorization of the matrix); hence, it is practical to generate samples with large n. Then, we estimate the parameters and compare the results against the true parameters that are used to generate $\boldsymbol{y}$. To evaluate the trace terms in (5) and (7), we restrict the simulation on a regular grid so that the involved matrices are multilevel Toeplitz. We use the technique presented in the appendix to evaluate the trace of the product of two multilevel Toeplitz matrices in $O(n)$ time. Furthermore, we compute the matrix-vector products in (5) and (7) by using an $O(n \log n)$ time algorithm with circulant embedding and fast Fourier transform [1]. For computing the confidence intervals, the Godambe information matrix $\mathcal{E}$ and the Fisher information matrix $\mathcal{I}$ cannot be computed exactly by using less than $O(n^3)$ time. We thus use the Hutchison estimator with $N = 50$ random vectors [1] to approximate the trace terms therein and use the conjugate gradient linear solver for solving linear systems with respect to K [6]. Specifically, Λ in (8), Γ in (9), and the Fisher information matrix $\mathcal{I}$ are approximated by their unbiased estimators

$$\widehat{\Lambda}_{ij} = \sum_{k=1}^{N} 2\boldsymbol{u}_k^T K_i K_j \boldsymbol{u}_k \tag{18}$$

$$\widehat{\Gamma}_{ij} = \sum_{k=1}^{N} 2\boldsymbol{u}_k^T K_i K K_j K \boldsymbol{u}_k \tag{19}$$

$$\widehat{\mathcal{I}}_{ij} = \sum_{k=1}^{N} \frac{1}{2}\boldsymbol{u}_k^T K^{-1} K_i K^{-1} K_j \boldsymbol{u}_k, \tag{20}$$

where each $\boldsymbol{u}_k$ is a random vector with independent entries of ± 1 with equal probability. The cost of computing the matrices Λ and Γ is then $O(Nn \log n)$, and the cost of computing $\mathcal{I}$ is $O(mNn \log n)$, where m is the number of conjugate gradient iterations.

An $n_1 \times n_2$ regular grid (so that $n = n_1 n_2$) is laid on a rectangular region of size 100×100. The true parameters are set to be $\boldsymbol{\theta} = [7; 13; 1]$. For estimation, we use the equations (5), solved as a nonlinear equation. The initial guess is the same as the truth, and the nonlinear solver is the Matlab `fsolve` function with the default algorithm `trust-region-dogleg`. Here we note that $\theta_3 = \alpha > 4$ may result in covariance matrices that are not positive definite. However, our numerical algorithm remained significantly away from that region, so this did not pose a problem for our setup. Table 2 summarizes the fitting results $\hat{\theta}_1$, $\hat{\theta}_2$, and $\hat{\theta}_3$, with the respective standard deviations $\sigma_1 = \sqrt{(\mathcal{E}^{-1})_{11}}$, $\sigma_2 = \sqrt{(\mathcal{E}^{-1})_{22}}$, and $\sigma_3 = \sqrt{(\mathcal{E}^{-1})_{33}}$, where $\mathcal{E}$ is evaluated at the estimate $\hat{\boldsymbol{\theta}}$. For comparison, we also list $\sigma_1' = \sqrt{(\mathcal{I}^{-1})_{11}}$, $\sigma_2' = \sqrt{(\mathcal{I}^{-1})_{22}}$, and $\sigma_3' = \sqrt{(\mathcal{I}^{-1})_{33}}$, where $\mathcal{I}$ is also evaluated at $\hat{\boldsymbol{\theta}}$. As anticipated by our theoretical results, the latter estimates of the standard deviations are not much smaller than the ones based on $\mathcal{E}$.

One sees that the estimated parameters agree well with the truth and that the two sets

of standard deviations are close. Clearly, the third parameter θ_3 (that is, the power α) is the best estimated; the standard deviations decrease by half every time n is increased by a factor of 4, as expected.

Table 2: Results for nonlinear equations formulation (5). Truth: $\theta = [7, 13, 1]$. Initial guess same as the truth.

Grid Size	64×64	128×128	256×256	512×512	1024×1024
$\hat{\theta}_1 \pm \sigma_1$	6.988±1.014	7.862±0.791	7.043±0.387	7.400±0.245	7.119±0.132
$\hat{\theta}_2 \pm \sigma_2$	12.684±1.965	14.165±1.502	13.448±0.786	13.653±0.474	13.281±0.258
$\hat{\theta}_3 \pm \sigma_3$	1.004±0.059	0.970±0.029	0.996±0.014	0.990±0.007	0.996±0.004
$\pm\sigma_1'$	±0.539	±0.404	±0.204	±0.128	±0.070
$\pm\sigma_2'$	±1.065	±0.782	±0.419	±0.251	±0.137
$\pm\sigma_3'$	±0.033	±0.016	±0.008	±0.004	±0.002

When we try to perform the same calculation with a different set of initial guess, even if the guess is close to the truth (e.g., $\boldsymbol{\theta}^{\texttt{guess}} = [10; 11; 1]$), the nonlinear solver does not converge to a desired point. To examine the cause, we show in Figure 2 the contour plots of (7) projected on the ℓ_1-α plane, together with the trajectory of the iterates. The black cross marker is the truth, somewhere close to the maximum of (7). The solver starts near the truth but gradually drifts away instead of approaching the truth. The residuals of the estimating equations (5) do decrease, but the value of the antiderivative (7) does not increase. The nonlinear solver uses a direction that is not an ascent direction for the corresponding optimization problem and consequently diverges away from its solution.

Thus, we directly solve the optimization problem (7) instead by using the Matlab function `fmincon` with the algorithm `interior-point/bfgs`. We pose (7) as a constrained optimization problem subject to $\theta_3 \in [0, 2]$ because of difficulties in manipulating the power-law kernel for larger α. However, since the bounds were not reached during optimization, not enforcing the bounds would have produced the same result. The solver successfully finds the solution even when the initial guess is far from the truth. In Table 3 we show the results for the initial guess $\boldsymbol{\theta}^{\texttt{guess}} = [30; 50; 1.8]$. One sees that all the estimates are close to the truth, just like the case in Table 2. The standard deviations of the parameters are also similar to the those in Table 2.

Table 3: Results for optimization formulation (7). Truth: $\boldsymbol{\theta} = [7; 13; 1]$. Initial guess: $\boldsymbol{\theta}^{\texttt{guess}} = [30; 50; 1.8]$.

Grid Size	64×64	128×128	256×256	512×512	1024×1024
$\hat{\theta}_1 \pm \sigma_1$	6.899±0.935	7.630±0.717	7.068±0.374	7.394±0.236	7.119±0.128
$\hat{\theta}_2 \pm \sigma_2$	12.574±1.887	13.803±1.404	13.453±0.771	13.636±0.464	13.282±0.253
$\hat{\theta}_3 \pm \sigma_3$	1.008±0.058	0.977±0.029	0.995±0.014	0.990±0.007	0.996±0.003

In Figure 3 we compare the computation time of two methods: (i) solving the estimating equations (5) in this paper and (ii) solving the SAA of the score equations proposed in [1]. Not

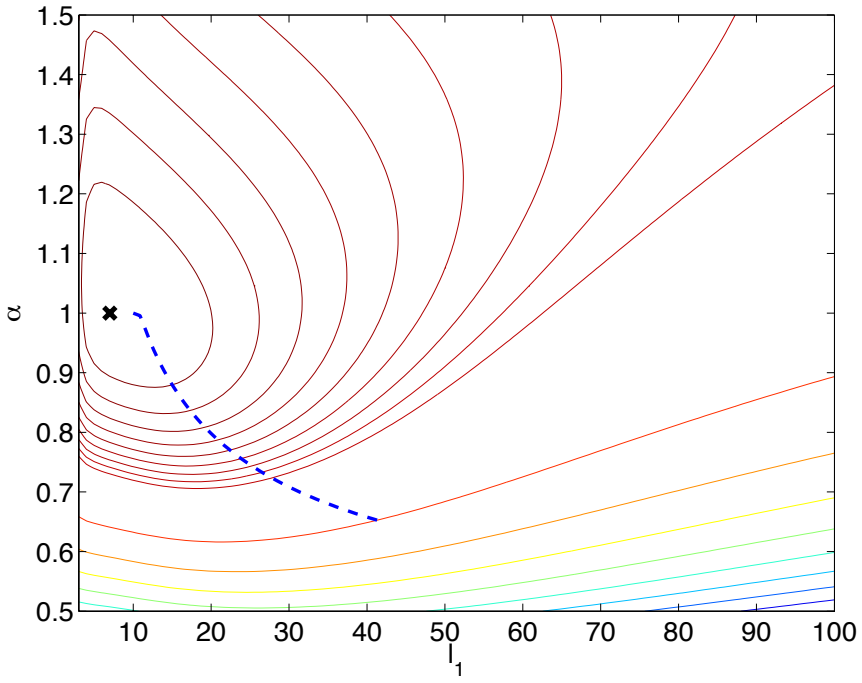

Figure 2: Contour plots of the objective function, projected on the ℓ_1-α plane, and the trajectory of the nonlinear solve.

plotted are the times to compute the confidence intervals; for (i), the time is approximately the same as that of computing the estimates; for (ii), the time is typically 1/10 of that of computing the estimates. The computational advantage of the former method is that it does not require linear solves as does the latter method. For fairness of comparison we use the nonlinear equations formulation for both approaches, since the SAA score equations approach does not have an optimization counterpart to (7). We start the initial guess at $\boldsymbol{\theta}^{\texttt{guess}} = [7; 13; 1]$ and use the Matlab function `fsolve`. The computations are performed on one compute node of a computing cluster. The compute node consists of Intel Sandy Bridge processors with a total of 16 cores, which means that Matlab is run with a maximum of 16 threads. In the figure, the dashed lines serve as references with a slope 1. One sees that the times approximately conform with the theoretical linear complexity; the fluctuations are caused by the dissimilar numbers of linear iterations and nonlinear iterations for different grid sizes. For the largest grid, solving the SAA of the score equations is more than 100 times slower than solving the estimating equations proposed in this paper. We note that for the largest case treated here, a classical approach based on maximum likelihood that requires a factorization of a 4 million by 4 million dense matrix would have been entirely beyond reach. The widths of the confidence intervals of the two schemes are similar.

When the smoothness order α is not in the form of $4\mathbb{Z}_+ - d$, where $\mathbb{Z}_+$ denotes a positive integer, filtering is not able to yield a bounded condition number. However, the bound in Theorem 1 that relates the ratio of $\mathcal{E}$ and $\mathcal{I}$ by the squared condition number is too

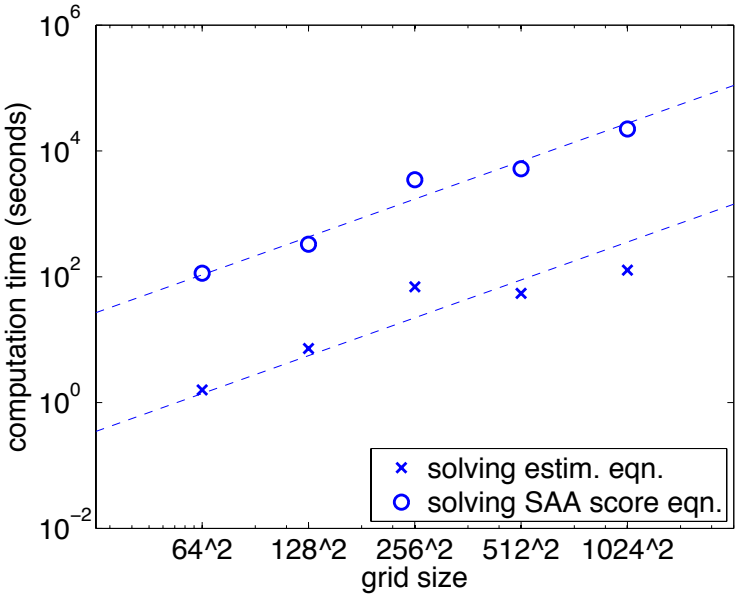

Figure 3: Computation time as grid size increases.

pessimistic sometimes. In Table 4 we list the ratios σ_1/σ_1', σ_2/σ_2', and σ_3/σ_3' for two cases: (i) $\alpha = 3$, filtered once, and (ii) $\alpha = 3$, filtered twice. One sees that in the first case the ratios increase slightly slower than $O(1/\sqrt{n})$ and in the latter case they stay approximately constant. Thus, even when the condition number is not bounded, fitting results originating from solving the estimating equations (5) can be as reliable as those originating from solving the standard score equations. The N/A case in the table demonstrate a computational advantage of the method presented in this paper: neither fitting nor the computation of confidence intervals requires a linear solve whose success is heavily tied to the conditioning of the covariance matrix.

Table 4: Comparison of $\mathcal{E}$ and $\mathcal{I}$. The N/A cases are caused by the failure of convergence (within 1,000 block conjugate gradient iterations) in the linear solves when estimating the diagonal of $\mathcal{I}$ by SAA for error estimation comparison purposes.

(i) $\alpha = 3$, filtered once.

Grid size	64×64	128×128	256×256	512×512	1024×1024
σ_1/σ_1'	2.497	4.038	8.158	15.187	27.396
σ_2/σ_2'	2.763	4.658	8.625	17.467	31.654
σ_3/σ_3'	6.147	7.758	19.516	40.365	67.807

(ii) $\alpha = 3$, filtered twice.

σ_1/σ_1'	2.956	2.936	2.919	2.879	N/A
σ_2/σ_2'	2.605	2.656	2.700	2.699	N/A
σ_3/σ_3'	2.478	2.536	2.589	2.601	N/A

5 Conclusions

Gaussian processes are important statistical models used in data analysis and uncertainty quantification. For large-scale data sets, estimating the covariance parameters of such models can be demanding because of the cost of computing the likelihood, which in its classical form requires performing the Cholesky factorization of a potentially dense and large covariance matrix.

In this work, we have presented a new approach for estimating the parameters of a Gaussian process, and, in general, of a Gaussian model. The approach solves an optimization problem whose optimality condition is the unbiased estimator of the Gaussian process covariance parameters. The most expensive part of setting up the gradient of this optimization problem is the computation of the trace of a product of matrices. This can be computed in $O(n)$ time, where n is the number of data points, for sparse matrices and for generally positioned data points. We prove that the Godambe information matrix of the resulting estimating equations is bounded below by a factor of the Fisher information matrix, where the factor is the inverse of a small power of the condition number of the covariance matrix. In turn, when the condition number of the covariance matrix is bounded independent of the number of data points, the resulting estimates have a statistical efficiency of the same order with the maximum likelihood independent of data size, at only a small fraction of its computational cost. Such a bounded condition number can be obtained, for example, for certain Matérn processes when preconditioned with Laplacian filtering [29, 4].

Moreover, for models whose covariance matrix is linear in its parameters, the estimating equations have a unique solution that can be obtained by solving a single $p \times p$ linear system, where p is the number of parameters. We note that for likelihood approaches one can rarely be guaranteed to practically obtain the global maximum for a large problem class. Further, the fact that we solve an optimization problem, as opposed to a nonlinear equation as we did for our recent scalable approach [1, 30], confers additional robustness to this method.

We demonstrate the benefit of the new approach on two examples of synthetic data using a sparse covariance matrix with linear dependence on its parameter and using a power-law process. In the latter case, the approach is compared with another stochastic estimation approach we introduced recently that was demonstrated to be scalable [1, 30] but that still required solving several systems of linear equations per nonlinear iteration. On these examples the estimating equation method introduced in this paper is proved accurate in the sense of recovering the parameters that were used to create the data, with an accuracy close to the optimal one as measured by the inverse of the Fisher information matrix, but at a fraction of the cost (1% or even less) for data sets of up to 1 million data points. We note that when estimating the Fisher information matrix itself for the large data sets by a stochastic approach—which requires linear solves with the covariance matrix—the preconditioned conjugate gradient algorithm took more than 1,000 iterations and still had not converged. On the other hand, the Godambe information matrix—which can be used to estimate the variance of the estimator produced by the new approach—does not require linear system solves with the covariance matrix and can be efficiently computed. Moreover, as for the power-law example where the covariance matrix was dense, the likelihood calculation would

have required a dense Cholesky factorization, which would have been very difficult for the 1 million data points because of its $O(n^3)$ complexity. These facts indicate the significant potential of computational efficiency combined with good accuracy that can be brought about by the proposed method.

The method does present several downsides. Likelihood calculations have an importance that goes beyond estimating parameters, such as in model comparison or for use with Bayesian approaches; our approach cannot make up for this lost information. Moreover, as opposed to likelihood approaches, there is no natural barrier to exploring parametric regions where the covariance function is not positive definite, if the set of validity of the parameters has a boundary. The latter may be a problem if the model class explored is not a good one for the data. While several fixes exist for some problem classes, this issue requires further investigation. However, the potential of the method of obtaining estimates of the parameters much faster than existing methods and, at least in some cases, at near-optimal accuracy motivates our further investigation in removing some of these shortcomings.

Acknowledgments

This work was supported by Contract No. DE-AC02-06CH11357 (Mihai Anitescu and Jie Chen) and Award DE-SC0011087 (Michael L. Stein) both from the U.S. Department of Energy. We gratefully acknowledge the use of the Blues cluster in the Laboratory Computing Resource Center at Argonne National Laboratory.

A A Linear-Time Algorithm for Computing the Trace of the Product of Two Toeplitz Matrices

Here, we present a linear-time algorithm for computing the trace of the product of two multilevel Toeplitz matrices with matching dimensions. The algorithm is simple to code in most programming languages. We demonstrate the code in Matlab.

We start from the 1-level case. A matrix A is (1-level) Toeplitz if each of its diagonals is constant, that is, A is in the form

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & \cdots & a_{n-1} \\ a_{-1} & a_0 & a_1 & \ddots & & \vdots \\ a_{-2} & a_{-1} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_1 & a_2 \\ \vdots & & \ddots & a_{-1} & a_0 & a_1 \\ a_{-n+1} & \cdots & \cdots & a_{-2} & a_{-1} & a_0 \end{bmatrix}. \tag{21}$$

The matrix A is represented solely by its first column and the first row and is stored as an

order-1 tensor:

$$\texttt{tA} = [a_{-n+1},\ \cdots,\ a_{-2},\ a_{-1},\ a_0,\ a_1,\ a_2,\ \cdots,\ a_{n-1}].$$

Denote by $C = AB$, which is the product of two Toeplitz matrices A and B. Since $\mathrm{tr}(C)$ is equal to the sum of all the elements of the matrix $A \circ B^T$, where $\circ$ denotes elementwise multiplication, and because $A \circ B^T$ is Toeplitz, we have

$$\mathrm{tr}(C) = \sum_{k=-n+1}^{n-1} |n+k| a_k b_{-k}.$$

Pictorially, we write the data representation of A, the flipping of the data representation of B, and a mask $|n+k|$ by ranging k from $-n+1$ to $n-1$ as in the following:

$$\begin{array}{rccccccccc}
\texttt{tA:} & a_{-n+1} & \cdots & a_{-2} & a_{-1} & a_0 & a_1 & a_2 & \cdots & a_{n-1} \\
\texttt{flip of tB:} & b_{n-1} & \cdots & b_2 & b_1 & b_0 & b_{-1} & b_{-2} & \cdots & b_{-n+1} \\
\texttt{mask:} & 1 & \cdots & n-2 & n-1 & n & n-1 & n-2 & \cdots & 1
\end{array}.$$

Then, we perform an elementwise multiplication of these order-1 data tensors and obtain the trace as the sum of the elements of the resulting tensor.

A multilevel Toeplitz matrix is defined recursively based on the number of levels. Specifically, in (21), A is $(d+1)$-level Toeplitz if each a_i, considered as a submatrix, is d-level Toeplitz. To keep track of the sizes, let the dimensions of each level be n_1, n_2, ..., n_d, ... Then, the data representation of a d-level Toeplitz matrix is an order-d tensor of size $(2n_1-1) \times (2n_2-1) \times \cdots \times (2n_d-1)$. The following shows the layout of a 2-level Toeplitz matrix

$$A = \left[\begin{array}{ccc|c|ccc}
a_{0,0} & \ddots & a_{0,n_2-1} & & a_{n_1-1,0} & \ddots & a_{n_1-1,n_2-1} \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
a_{0,-n_2+1} & \ddots & \ddots & & a_{n_1-1,-n_2+1} & \ddots & \ddots \\
\hline
 & \ddots & & \ddots & & \ddots & \\
\hline
a_{-n_1+1,0} & \ddots & a_{-n_1+1,n_2-1} & & \ddots & \ddots & \ddots \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
a_{-n_1+1,-n_2+1} & \ddots & \ddots & & \ddots & \ddots & \ddots
\end{array}\right]$$

and its data representation:

$$\texttt{tA} = \begin{bmatrix} a_{-n_1+1,-n_2+1} & \cdots & a_{-n_1+1,0} & \cdots & a_{-n_1+1,n_2-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{0,-n_2+1} & \cdots & a_{0,0} & \cdots & a_{0,n_2-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n_1-1,-n_2+1} & \cdots & a_{n_1-1,0} & \cdots & a_{n_1-1,n_2-1} \end{bmatrix}. \tag{22}$$

To compute $\mathrm{tr}(AB)$, where A and B are both d-level Toeplitz with matching dimensions, we perform a procedure similar to the 1-level case. It suffices to show a 2-level example. We reuse the data representation of A, `tA`, in (22). We write the flipping of the data representation of B as in the following:

$$\texttt{flip of tB} = \begin{bmatrix} b_{n_1-1,n_2-1} & \cdots & b_{n_1-1,0} & \cdots & b_{n_1-1,-n_2+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{0,n_2-1} & \cdots & b_{0,0} & \cdots & b_{0,-n_2+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{-n_1+1,n_2-1} & \cdots & b_{-n_1+1,0} & \cdots & b_{-n_1+1,-n_2+1} \end{bmatrix}.$$

Note that the flipping is performed along all the dimensions. The mask is defined as

$$\texttt{mask} = \begin{bmatrix} 1 \times 1 & \cdots & 1 \times n_2 & \cdots & 1 \times 1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ n_1 \times 1 & \cdots & n_1 \times n_2 & \cdots & n_1 \times 1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 \times 1 & \cdots & 1 \times n_2 & \cdots & 1 \times 1 \end{bmatrix},$$

which is the outer product of two 1-level masks $[1,\ \ldots,\ n_1-1,\ n_1,\ n_1-1,\ \ldots,\ 1]$ and $[1,\ \ldots,\ n_2-1,\ n_2,\ n_2-1,\ \ldots,\ 1]$. We perform the elementwise multiplication of `tA`, `flip of tB`, and `mask`. Then, $\mathrm{tr}(AB)$ is the sum of all the elements of the resulting tensor.

Figure 4 shows the Matlab code, which consists of only seven lines of calculations, excluding the comments. The code was written in a manner that generalization to cases other than 2-level Toeplitz is straightforward. One sees that the computational cost (both in time and in storage) for the general d-level case is $O(n_1 n_2 \cdots n_d)$, linear in the number of rows of A and B. If the matrices are symmetric in all Toeplitz levels, they can be represented by using data tensors of size $n_1 \times n_2 \times \cdots \times n_d$. Thus, the code can be straightforwardly optimized, resulting in a computational cost reduced by a factor of 2^d.

References

[1] M. Anitescu, J. Chen, and L. Wang, *A matrix-free approach for solving the para-*

```
function tr = tr_toep_toep_mult_2level(tA, tB)
% This code computes the trace of the product of two 2-level Toeplitz
% matrices A and B, which are stored as tA and tB by using a data
% tensor format. Let the dimensions of each level be n1 and n2,
% respectively; that is, A has n1*n2 rows (and columns), and similarly
% for B. Both A and B are stored as an order-2 tensor of size
% (2*n1-1)*(2*n2-1).

% Get the size of tA
sz1 = size(tA);

% Get the two numbers n1 and n2
sz2 = (sz1+1)/2;

% Compute the elementwise product of tA and the flipping of tB
tC = tA .* tB(end:-1:1, end:-1:1);

% Generate mask. The mask is the outer product of two vectors v1 and
% v2, where v1 = 1,2,..,n1,..,2,1 and v2 = 1,2,..,n2,..,2,1
mask1 = repmat(reshape([1:sz2(1) sz2(1)-1:-1:1], sz1(1),1), [1,sz1(2)]);
mask2 = repmat(reshape([1:sz2(2) sz2(2)-1:-1:1], 1,sz1(2)), [sz1(1),1]);
mask = mask1 .* mask2;

% Final result is the sum of all the elements of the elementwise
% product of tC and mask
tr = sum(sum(tC.*mask));
```

Figure 4: Matlab code of computing $\mathrm{tr}(AB)$, where A and B are 2-level Toeplitz.

metric Gaussian process maximum likelihood problem, SIAM Journal on Scientific Computing, 34 (2012), pp. A240–A262.

[2] V. P. Bhapkar, *On a measure of efficiency of an estimating equation*, Sankhyā A, 34 (1972), pp. 467–472.

[3] P. C. Caragea and R. L. Smith, *Asymptotic properties of computationally efficient alternative estimators for a class of multivariate normal models*, Journal of Multivariate Analysis, 98 (2007), pp. 1417–1440.

[4] J. Chen, *On the use of discrete Laplace operator for preconditioning kernel matrices*, SIAM Journal on Scientific Computing, 35 (2013), pp. A577–A602.

[5] J. Chen, M. Anitescu, and Y. Saad, *Computing $f(A)b$ via least squares polynomial approximations*, SIAM J. Sci. Comput., 33 (2011), pp. 195–222.

[6] J. Chen and T. L. H. Li, *Parallelizing the conjugate gradient algorithm for multilevel Toeplitz systems*, Procedia Computer Science, 18 (2013), pp. 571–580.

[7] J. Chen, T. L. H. Li, and M. Anitescu, *A parallel linear solver for multilevel Toeplitz systems with possibly several right-hand sides*, Tech. Rep. ANL/MCS-P5040-1113, Argonne National Laboratory, 2013.

[8] J. Chen, L. Wang, and M. Anitescu, *A parallel tree code for computing matrix-vector products with the Matérn kernel*, Tech. Rep. ANL/MCS-P5015-0913, Argonne National Laboratory, 2013.

[9] ——, *A fast summation tree code for Matérn kernel*, SIAM J. Sci. Comput., (to appear).

[10] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge, UK: Cambridge University Press, 2005.

[11] J. P. Chilès and P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty*, New York: Wiley-Interscience, 2nd ed., 2012.

[12] N. Cressie and G. Johannesson, *Fixed rank kriging for very large spatial data sets*, Journal of the Royal Statistical Society, Series B, 70 (2008), pp. 209–226.

[13] R. Dahlhaus and H. Künsch, *Edge effects and efficient parameter estimation for stationary random fields*, Biometrika, 74 (1987), pp. 877–882.

[14] P. J. Diggle, J. A. Tawn, and R. A. Moyeed, *Model-based geostatistics*, Applied Statistics, 47 (1998), pp. 299–350.

[15] M. Eidsvik, A. O. Finley, S. Banerjee, and H. Rue, *Approximate Bayesian inference for large spatial datasets using predictive process models*, Computational Statistics and Data Analysis, 56 (2012), pp. 1362–1380.

[16] M. Fuentes, *Approximate likelihood for large irregularly spaced spatial data*, Journal of the American Statistical Association, 102 (2007), pp. 321–331.

[17] R. Furrer, M. G. Genton, and D. Nychka, *Covariance tapering for interpolation of large spatial datasets*, Journal of Computational and Graphical Statistics, 15 (2006), pp. 502–523.

[18] X. Guyon, *Parameter estimation for a stationary process on a d-dimensional lattice*, Biometrika, 69 (1982), pp. 95–105.

[19] R. R. Hocking, *Methods and applications of linear models: regression and the analysis of variance*, Wiley Interscience, New York, 1996.

[20] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, 2012.

[21] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Communications in Statistics – Simulation and Computation, 19 (1990), pp. 433–450.

[22] P. McCullagh and J. A. Nelder, *Generalized linear models (monographs on statistics and applied probability 37)*, Chapman Hall, London, (1989).

[23] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.

[24] C. R. Rao, *Estimation of variance and covariance componentsminque theory*, Journal of multivariate analysis, 1 (1971), pp. 257–275.

[25] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*, MIT Press, Cambridge, Massachusets., 2006.

[26] M. Stein, *Interpolation of spatial data: Some theory for Kriging*, Springer-Verlag, Berlin, 1999.

[27] M. L. Stein, *A modeling approach for large spatial datasets*, Journal of the Korean Statistical Society, 37 (2008), pp. 3–10.

[28] ——, *Statistical properties of covariance tapers*, Journal of Computational and Graphical Statistics, (2012), p. DOI: 10.1080/10618600.2012.719844.

[29] 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.

[30] M. L. Stein, J. Chen, and M. Anitescu, *Stochastic approximation of score functions for Gaussian processes*, Annals of Applied Statistics, 7 (2013), pp. 1162–1191.

[31] M. L. Stein, Z. Chi, and L. J. Welty, *Approximating likelihoods for large spatial datasets*, Journal of the Royal Statistical Society, Series B, 66 (2004), pp. 275–296.

[32] C. Varin, N. Reid, and D. Firth, *An overview of composite likelihood methods*, Statistica Sinica, 21 (2011), pp. 5–42.

[33] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, Journal of the Royal Statistical Society, Series B, 50 (1988), pp. 297–312.

[34] P. Whittle, *On stationary processes in the plan*, Biometrika, 41 (1954), pp. 434–449.

[35] Y. Zhang, *Uniformly distributed seeds for randomized trace estimator on $O(N^2)$-operation log-det approximation in gaussian process regression*, in ICNSC '06. Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006, pp. 498–503.

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. DE-AC02-06CH11357 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.