NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

Estimation and prediction in spatial models with block composite likelihoods using parallel computing

by

Eidsvik, J., Shaby, B.A., Reich, B.J., Wheeler, M. and Niemi, J.

PREPRINT
STATISTICS NO. 7/2011

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL http://www.math.ntnu.no/preprint/statistics/2011/S7-2011.pdf
Jo Eidsvik has homepage: http://www.math.ntnu.no/∼joeid
E-mail: joeid@stat.ntnu.no
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway.

Estimation and prediction in spatial models with block composite likelihoods using parallel computing

Jo Eidsvik[1], Benjamin A. Shaby[2], Brian J. Reich[3], Matthew Wheeler[4] and Jarad Niemi[4]

1) Department of Mathematical Sciences, NTNU, Norway.

2) Department of Statistical Science, Duke University, US.

3) Department of Statistics, North Carolina State University, US.

4) Department of Statistics and Applied Probability, University of California at Santa Barbara, US.

Corresponding author: Jo Eidsvik (joeid@math.ntnu.no), Department of Mathematical Sciences, NTNU, 7491 Trondheim, NORWAY.

Abstract

A block composite likelihood model is developed for estimation and prediction in large spatial datasets. The composite likelihood is constructed from the joint densities of pairs of adjacent spatial blocks. This allows large datasets to be split into many smaller datasets, each of which can be evaluated separately, and combined through a simple summation. Estimates for unknown parameters as well as optimal spatial predictions under the block composite model are obtained. Asymptotic variances for both parameter estimates and predictions are computed using Godambe sandwich matrices. In addition to the considerable increases in computational efficiency it achieves, the composite structure also obviates the need to load entire datasets into memory at once, completely avoiding memory limitations imposed by massive datasets. Moreover, computing time can be drastically reduced even further by distributing the operations using parallel computing. A simulation study shows that composite likelihood estimates and predictions, as well as their corresponding asymptotic confidence intervals, are competitive with those based on the full likelihood. The procedure is demonstrated on one dataset from the mining industry and one dataset of satellite retrievals. The real-data examples show that the block composite results tend to outperform two competitors; the predictive process model and fixed rank Kriging.

Keywords: composite likelihood, spatial statistics, parallel computing, GPU, Gaussian process

1 Introduction

In recent years there has been a tremendous increase in the magnitude and pervasiveness of massive geocoded scientific datasets. The growth in size is to a large extent driven by new technologies such as GPS and remote sensing, as well as by the ever-increasing storage capacity of digital databses. The explosion of interest in climate research has brought these types of datasets into the spotlight. These developments have triggered demand for more sophisticated statistical modeling and methodology for such data. The computations required for inference and prediction in spatial Gaussian process models, the central construct in spatial statistics, are challenging for large datasets because they require manipulations of large covariance matrices. In particular, evaluation of the likelihood function necessitates performing inverse and determinant operations, both of which are computationally intractable for large matrices.

Several approaches have been proposed to mitigate this computational burden. For instance, Fuentes (2007) approximates the likelihood using the spectral representation of the spatialprocess. Furrer et al. (2006), Kaufman et al. (2008), and Shaby and Ruppert (2011) use covariance tapering, where the full covariance function is multiplied by a compactly-supported correlation function, yielding a sparse covariance matrix, which enables specialised algorithms to be leveraged. Another strategy is to represent the spatial process in a lower-dimensional subspace using low-rank models (e.g. Stein, 2008; Cressie and Johannesson, 2008; Banerjee et al., 2008). Gaussian Markov random fields are also useful for fast computation of Gaussian processes (Lindgren et al., 2011). Finally, there is a large literature on high-dimensional Gaussian distributions in machine learning (Rasmussen and Williams, 2006) and numerical linear algebra (Higham, 2008).

In this paper we implement a unified framework for parameter estimation and prediction based on the composite likelihood (Lindsay, 1988; Varin, 2008). The composite likelihood (CL) is a product of several joint likelihoods of subsets of the data. One important special case is the pairwise likelihood, which is the product of all bivariate marginal likelihoods. Here, we use a form of the CL function defined as the product of joint density functions of pairs of spatial *blocks*. The motivation behind the spatial blocking strategy is that it captures much of the spatial dependence, while still providing the divide and conquer aspect of the CL, which reduces computational complexity and facilitates fast parallel computing.

In the parameter estimation context, the asymptotic properties of the CL are well-understood. Maximum CL estimates are consistent and asymptotically normal under similar conditions as corresponding maximum likelihood estimates (MLEs). The asymptotic covariance for maximum CL estimates is given by a sandwich matrix (Godambe, 1960; Godambe and Heyde, 1987) rather than the usual Fisher information matrix for MLEs.

In addition to parameter estimation for Gaussian random field models, we show how to use the CL for the crucial complementary problem of spatial prediction, which has not previously been considered. We demonstrate how to construct predictions at unobserved sites that are optimal under the block CL, the composite analogue to Kriging. This prediction approach allows fast computing and follows the same approach as the methods we use for parameter estimation. We derive asymptotic prediction variances under the CL model, which have the familiar sandwich form.

The earliest use of the CL for random fields seems to be Curriero and Lele (1999), who used the pairwise form of the CL to estimate covariance parameters, and establish consistency. Several attempts have been made to utilise spatial blocking. Among them is Caragea and Smith (2007), who do not use the CL likelihood in the way we do here. Stein et al. (2004) use a restricted likelihood version of the telescoping conditional probability approximation of Vecchia (1988), which achieves fast computations by reducing the conditioning set to a small subset of the data. This is similar in spirit to the CLs, but it is not obvious how to either use it for spatial prediction or implement it in parallel. Also similar is the notion of pseudolikelihood (Besag, 1974), which uses the product of all full conditional distributions for parameter estimation.

The block CL model reduces the computational burden to $O(n)$, where the hidden constant would depend on the block sizes. Moreover, the usual memory restrictions for large datasets are avoided since the blocks of data can be loaded into memory separately. Finally, the CL approach allows parallel computing. We implement the estimation procedure on a Graphical Processing Unit (GPU) and achieve a many-fold increase in computational efficiency. This speed-up comes on top of the efficiency gains resulting from the structure of the CL function. Graphics cards have evolved into massively-parallel computational engines, which can be appropriated for scientific applications. Statisticians are beginning to exploit the technology (Suchard and Rambaut, 2009; Suchard et al., 2010; Lee et al., 2010), though there is a great deal of room for further development. To date we are not

aware of any examples of exploiting GPUs in the context of spatial statistics or composite likelihoods.

The paper is organised as follows. Section 2 defines the block CL function. Methods for estimation and prediction are presented in Section 3, and computational methods and parallelisation issues are described in Section 4. Section 5 provides simulation studies. Section 6 presents two data examples. The computational details related to the CL function and to parallel implementation are in the Appendix.

2 Block composite likelihood for spatial data

2.1 Geostatistical model

Geostatistical settings typically assume, at locations $\boldsymbol{s} \in D \subseteq \Re^d$, $d = 2$ or 3, a Gaussian response variable $Y(\boldsymbol{s})$ along with a $p \times 1$ vector of spatially-referenced explanatory variables $\boldsymbol{x}(\boldsymbol{s})$ which are associated through a spatial regression model

$$Y(\boldsymbol{s}) = \boldsymbol{x}'(\boldsymbol{s})\boldsymbol{\beta} + w(\boldsymbol{s}) + \epsilon(s), \tag{1}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$ is the regression parameter, and $\epsilon(s) \sim N(0, \tau^2)$ is independent error. The spatial residual $w(\boldsymbol{s})$ provides structural dependence, capturing the effect of unobserved covariates with spatial pattern. The covariance structure of the Gaussian process $w(\boldsymbol{s})$ is typically characterised by a small number of parameters. We denote the collection of all covariance parameters $\boldsymbol{\theta}$, which includes the nugget effect τ^2. Some common models for $\mathrm{Cov}(w(\boldsymbol{s}), w(\boldsymbol{s}')) = C(\boldsymbol{s}', \boldsymbol{s})$ are the exponential, Matérn, and Cauchy covariance models (see Banerjee et al., 2004, e.g.).

We assume data are available at n locations $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$, and denote the collection of data $\boldsymbol{Y} = (Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n))'$. Then $\boldsymbol{Y} \sim N(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{C} + \tau^2 \boldsymbol{I}_n$, with $C(i,j) = \mathrm{Cov}(w(\boldsymbol{s}_i), w(\boldsymbol{s}_j))$. Moreover, row i of matrix $\boldsymbol{X}$ contains the explanatory variables $\boldsymbol{x}'(\boldsymbol{s}_i)$. Ignoring a scalar that does not depend on $\boldsymbol{\beta}$ or $\boldsymbol{\theta}$, the log likelihood is

$$\ell(\boldsymbol{Y}; \boldsymbol{\beta}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}). \tag{2}$$

Noting that $\boldsymbol{\Sigma}$ is $n \times n$, the difficulty with the usual maximum likelihood methods is apparent; evaluating the log-likelihood requires computing $|\boldsymbol{\Sigma}|$ and a quadratic form that includes $\boldsymbol{\Sigma}^{-1}$, both of which are computationally intractable for large n.

2.2 Composite likelihood

In contrast to the full joint likelihood (2), the CL function (Lindsay, 1988) is constructed as the product of marginal likelihoods of subsets of the data, proceeding as though these subsets were independent. If the CL model ignores the less important interactions in data $\boldsymbol{Y}$, it can provide a good approximation to the full likelihood. One version of a composite log likelihood for the geostatistical model (1) is the pairwise likelihood

$$\ell_{PCL}(\boldsymbol{Y}; \boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{i=1}^{n-1} \sum_{i'>i} \ell(Y(\boldsymbol{s}_i), Y(\boldsymbol{s}_{i'}); \boldsymbol{\beta}, \boldsymbol{\theta}), \tag{3}$$

where $\ell(Y(\boldsymbol{s}_i), Y(\boldsymbol{s}_{i'}); \boldsymbol{\beta}, \boldsymbol{\theta})$ is the log of the bivariate normal density of $Y(\boldsymbol{s}_i)$ and $Y(\boldsymbol{s}_{i'})$ (Curriero and Lele, 1999).

Instead, we next present a block CL, where we partition the region D into M blocks $D_1, \ldots, D_M$, with $\cup_k D_k = D$, $D_k \cap D_l = \emptyset$, for all pairs of blocks k, l. Denote the response in block $k = 1, \ldots, M$ as $\boldsymbol{Y}_k = \{Y(\boldsymbol{s}_i); \boldsymbol{s}_i \in D_k\}$. The number of sites in block k is n_k, $\sum_k n_k = n$. Let $\boldsymbol{Y}_{kl} = (\boldsymbol{Y}_k', \boldsymbol{Y}_l')'$ be the collection of data in block k and l. We define the block composite log likelihood as

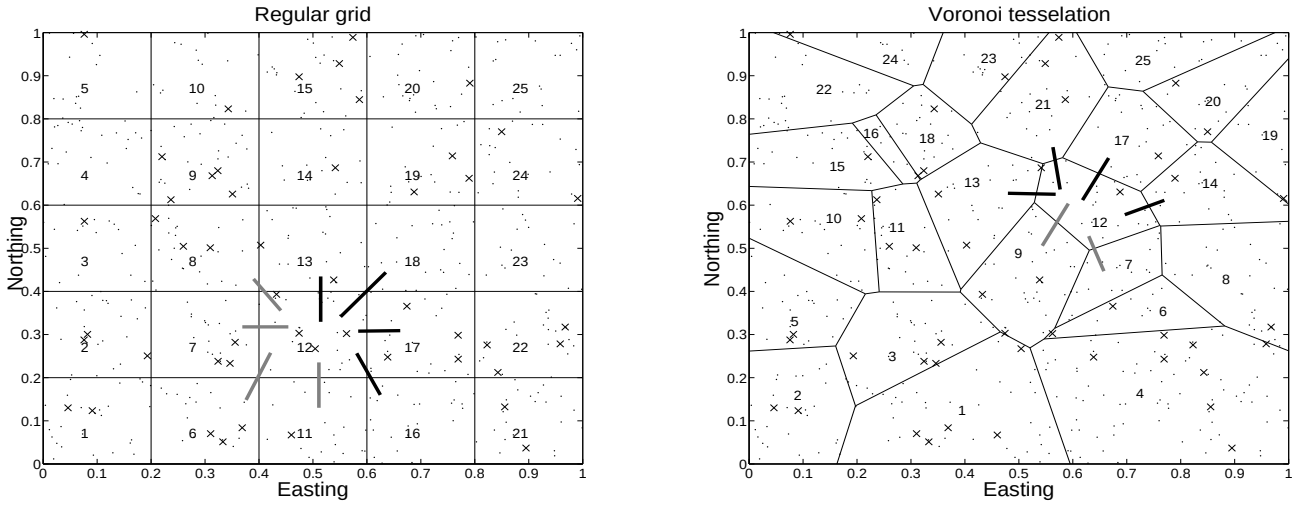
$$\begin{aligned}
\ell_{CL}(\boldsymbol{Y}; \boldsymbol{\beta}, \boldsymbol{\theta}) &= \sum_{k=1}^{M-1} \sum_{l>k} \ell(\boldsymbol{Y}_{kl}; \boldsymbol{\beta}, \boldsymbol{\theta}) \\
&= \sum_{k=1}^{M-1} \sum_{l>k} \Big[-\frac{1}{2}\log|\boldsymbol{\Sigma}_{kl}| - \frac{1}{2}(\boldsymbol{Y}_{kl} - \boldsymbol{X}_{kl}\boldsymbol{\beta})'\boldsymbol{\Sigma}_{kl}^{-1}(\boldsymbol{Y}_{kl} - \boldsymbol{X}_{kl}\boldsymbol{\beta}) \Big].
\end{aligned} \tag{4}$$

Here, $\boldsymbol{X}_{kl} = (\boldsymbol{X}_k', \boldsymbol{X}_l')'$ is the collection of all covariates in block k and l, and $\boldsymbol{\Sigma}_{kl}$ is the $(n_k + n_l) \times (n_k + n_l)$ covariance matrix

$$\boldsymbol{\Sigma}_{kl} = \begin{bmatrix} \boldsymbol{\Sigma}_{kl}(1) & \boldsymbol{\Sigma}_{kl}(1,2) \\ \boldsymbol{\Sigma}_{kl}(2,1) & \boldsymbol{\Sigma}_{kl}(2) \end{bmatrix}. \tag{5}$$

This covariance matrix is thus partitioned into four blocks, where $\boldsymbol{\Sigma}_{kl}(1)$ is the $n_k \times n_k$ covariance matrix of $\boldsymbol{Y}_k$, $\boldsymbol{\Sigma}_{kl}(2)$ is the $n_l \times n_l$ covariance matrix of $\boldsymbol{Y}_l$, and $\boldsymbol{\Sigma}_{kl}(1,2) = \boldsymbol{\Sigma}_{kl}'(2,1)$ is the $n_k \times n_l$ cross-covariance between $\boldsymbol{Y}_k$ and $\boldsymbol{Y}_l$. If $M = 1$ or 2, the block CL (4) is equal to the full likelihood (2); if $M = n$, we get the pairwise likelihood (3). The block CL (4) is a natural compromise for spatial models, as the number of blocks M represents a trade-off between computational and statistical efficiency.

The CL expression is simplified by omitting distant pairs of blocks from expression (4), assuming negligible dependence between blocks if they are not neighbours. Let N_k denote the neighbours of block k. Figure 1(a) shows an illustration of a regular block design with $M = 5 \cdot 5 = 25$ blocks on a $2D$ domain, while Figure 1(b) shows a Voronoi / Delaunay design. Blocks are neighbours if they share a common border. The Voronoi / Delaunay design allows the block sizes to adapt in domains with non-uniform data densities, and the number of neighbouring blocks vary. In any event, the neighbour structure is easy to represent as a graph. The edges of block 12 are shown in Figure 1. In Figure 1(b) block 7 has a border with 14, and this prevents block 8 from being a neighbour of 12. Many alternative blocking designs are possible.



(a) (b)

Figure 1: Observation sites illustrated by '.' and predictions sites by 'x'. A block CL splits the spatial domain into regular (a) or irregular (b) blocks. Each block communicates pairwise with each of its neighbours. For the regular grid (a), an interior block has eight neighbours. For a random or adaptive grid (b), the number of neighbours varies. In both displays the block indexed 12 has four neighbours with higher indices (black edges).

This neighbour set N_k can be split into a forward part $N_k^{\rightarrow} = \{l > k\} \cap \{l \in N_k\}$ and a backward part $N_k^{\leftarrow} = \{l < k\} \cap \{l \in N_k\}$. These two are displayed using black and gray edge lines in Figure 1. By only considering these neighbouring blocks, the second sum in (4) is only evaluated over $l \in N_k^{\rightarrow}$, so that

$$\begin{aligned} \ell_{CL}(\boldsymbol{Y}; \boldsymbol{\beta}, \boldsymbol{\theta}) &= \sum_{k=1}^{M-1} \sum_{l \in N_k^{\rightarrow}} \Big[-\frac{1}{2} \log |\boldsymbol{\Sigma}_{kl}| - \frac{1}{2} (\boldsymbol{Y}_{kl} - \boldsymbol{X}_{kl}\boldsymbol{\beta})' \boldsymbol{\Sigma}_{kl}^{-1} (\boldsymbol{Y}_{kl} - \boldsymbol{X}_{kl}\boldsymbol{\beta}) \Big] \\ &= \sum_{j} \Big[-\frac{1}{2} \log |\boldsymbol{\Sigma}_j| - \frac{1}{2} (\boldsymbol{Y}_j - \boldsymbol{X}_j\boldsymbol{\beta})' \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{Y}_j - \boldsymbol{X}_j\boldsymbol{\beta}) \Big]. \end{aligned} \tag{6}$$

Here, the shorthand notation with index j is used to represent the set of edges $(k,l)|l \in N_k^{\rightarrow}$ in the graph of blocks and block neighbours. For instance, in Figure 1(a), the set of $j = (k,l)$'s is $(1,2)$, $(1,6)$, $(1,7)$, $(2,3)$,

$(2,6)$, ..., $(24,25)$. The edge notation induces the corresponding shorthand for the block-pair variables $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}_{kl}$, $\boldsymbol{Y}_j = \boldsymbol{Y}_{kl}$, and $\boldsymbol{X}_j = \boldsymbol{X}_{kl}$ defined in (4) and (5). The neighbour structure implies a working assumption of conditional pair dependence, proceeding as though block k was conditionally independent of blocks outside its neighbourhood, given the pairwise block dependencies to k within the neighbourhood.

For data on a regular grid, the relative distances between sites in block-pairs are the same, giving identical covariance matrices $\boldsymbol{\Sigma}_j$ for all equal-sized block pair neighbours $j = (k,l)$, under stationarity and isotropy assumptions on the random field. In this case the required determinants and inverses can be computed only once per block CL evaluation.

3 Inference and prediction using block composite likelihood

3.1 Properties of the Maximum Composite Likelihood Estimator

The maximum CL estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are given by

$$(\hat{\boldsymbol{\beta}}_{CL}, \hat{\boldsymbol{\theta}}_{CL}) = \text{argmax}_{\boldsymbol{\beta},\boldsymbol{\theta}} \left[\ell_{CL}(\boldsymbol{Y}; \boldsymbol{\beta}, \boldsymbol{\theta})\right].$$

In general, the maximum CL estimators are known to be consistent and asymptotically normal under the same conditions as MLEs (Lindsay, 1988). In the case of spatial Gaussian processes, conditions such as those in Mardia and Marshall (1984) yield the desired asymptotic properties for the resultant maximum CL estimators (Curriero and Lele, 1999, e.g).

A useful place to begin analytical exposition is with the vector-valued block composite score function, defined as $\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\beta},\boldsymbol{\theta})}{\partial \theta_r}$, $r = 1, \ldots, R$, where R is the dimension of the parameter $\boldsymbol{\theta}$. For notational simplicity, we assume here that $\boldsymbol{Y}$ is a mean-zero Gaussian random field. Differentiating (6) with respect to θ_r, the score (Appendix A) can be expressed as

$$\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{\partial \theta_r} = \sum_j \Big[-\frac{1}{2}\text{trace}\left(\boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{\partial \theta_r}\right) + \frac{1}{2}\boldsymbol{Y}'_j \boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{\partial \theta_r} \boldsymbol{Q}_j \boldsymbol{Y}_j \Big],$$

where $\boldsymbol{Q} = \boldsymbol{\Sigma}_j^{-1}$. Taking expectations and using $\text{E}(\boldsymbol{Y'BY}) = \text{trace}(\boldsymbol{B\Sigma})$ we see that

$$E\left(\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_r}\right) = \sum_j \Big[-\frac{1}{2}\text{trace}\left(\boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{\partial \theta_r}\right) + \frac{1}{2}\text{trace}\left(\boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{\partial \theta_r}\right) \Big] = 0,$$

revealing that the block composite score is an unbiased estimating function for $\boldsymbol{\theta}$ for any blocking scheme.

As is typical of asymptotically-normal estimators resulting from unbiased estimating functions, the asymptotic covariance of $\hat{\boldsymbol{\theta}}_{CL}$ has a sandwich form (Godambe, 1960), $\hat{\boldsymbol{\theta}}_{CL} \sim N(\boldsymbol{\theta}, \boldsymbol{G}^{-1})$, where

$$\begin{aligned} \boldsymbol{G} = \boldsymbol{G}(\boldsymbol{\theta}) &= \boldsymbol{H}(\boldsymbol{\theta})\boldsymbol{J}^{-1}(\boldsymbol{\theta})\boldsymbol{H}(\boldsymbol{\theta}), \\ \boldsymbol{H}(\boldsymbol{\theta}) &= -E\left(\frac{\partial^2 \ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}\right), \quad \boldsymbol{J}(\boldsymbol{\theta}) = \text{Var}\left(\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right). \end{aligned}$$

We note that in the case of the full likelihood function, $\boldsymbol{H}(\boldsymbol{\theta})\boldsymbol{J}^{-1}(\boldsymbol{\theta}) = \boldsymbol{I}$, so $\boldsymbol{G}(\boldsymbol{\theta})$ is just the Fisher information. For the block CL, analytical expressions are available for both $\boldsymbol{H}(\boldsymbol{\theta})$ and $\boldsymbol{J}(\boldsymbol{\theta})$. The negative expected Hessian (Appendix A) has elements

$$H_{rs}(\boldsymbol{\theta}) = \sum_j \frac{1}{2}\text{trace}\left(\boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{d\theta_s} \boldsymbol{Q}_j \frac{\partial \boldsymbol{\Sigma}_j}{d\theta_r}\right), \quad r,s = 1, \ldots, R. \tag{7}$$

The expression for $\boldsymbol{J}(\boldsymbol{\theta})$ is more complicated (Appendix A). In practice we evaluate $\boldsymbol{H}(\boldsymbol{\theta})$ and $\boldsymbol{J}(\boldsymbol{\theta})$ at $\hat{\boldsymbol{\theta}}_{CL}$.

To re-introduce covariates into the model, we simply substitute $(\boldsymbol{Y}_j - \boldsymbol{X}_j\hat{\boldsymbol{\beta}}_{CL})$ for $\boldsymbol{Y}_j$ into the above expressions. Analogously, the maximum block CL estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are obtained by maximizing (6). For fixed $\boldsymbol{\theta}$,

the regression estimate $\hat{\boldsymbol{\beta}}_{CL}$ is analytically available by writing out the quadratic form in (6) in terms of $\boldsymbol{\beta}$. This gives

$$\hat{\boldsymbol{\beta}}_{CL} = \boldsymbol{A}^{-1}\boldsymbol{b}, \quad \boldsymbol{A} = \sum_j \boldsymbol{X}_j'\boldsymbol{Q}_j\boldsymbol{X}_j, \quad \boldsymbol{b} = \sum_j \boldsymbol{X}_j'\boldsymbol{Q}_j\boldsymbol{Y}_j. \tag{8}$$

The covariance matrix of the limiting normal distribution of $\hat{\boldsymbol{\beta}}_{CL}$ also has a sandwich form, which is computed from the expected Hessian and the variance of the score. The negative expected Hessian is simply $\boldsymbol{H}(\boldsymbol{\beta}) = \boldsymbol{A}$ as defined in (8). The variance of the score is

$$\begin{aligned}\boldsymbol{J} = \mathrm{Var}\left(\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{\partial \boldsymbol{\beta}}\right) &= \sum_j \sum_{j'} \mathrm{Cov}(\boldsymbol{X}_j'\boldsymbol{Q}_j\boldsymbol{Y}_j, \boldsymbol{X}_{j'}'\boldsymbol{Q}_{j'}\boldsymbol{Y}_{j'}) \\ &= \sum_j \sum_{j'} \boldsymbol{X}_j'\boldsymbol{Q}_j \mathrm{Cov}(\boldsymbol{Y}_j, \boldsymbol{Y}_{j'})\boldsymbol{Q}_{j'}\boldsymbol{X}_{j'}. \end{aligned} \tag{9}$$

In practice, we only sum over terms with edges $j = (k, l)$ and $j' = (k', l')$ that have common nodes among the blocks (k, l, k', l'). This means that $\boldsymbol{Y}_j$ and $\boldsymbol{Y}_{j'}$ that do not have common elements are excluded from (9).

3.2 Prediction using block composite likelihood

Suppose that we want to predict the value of $Y(\boldsymbol{s}_0)$ at an unobserved site $\boldsymbol{s}_0$. The best linear unbiased prediction (BLUP) of $Y(\boldsymbol{s}_0)$ given the data $\boldsymbol{Y}$ is

$$\hat{Y}(\boldsymbol{s}_0) = \boldsymbol{x}'(\boldsymbol{s}_0)\boldsymbol{\beta} + \boldsymbol{\Sigma}_{0,1:n}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}), \tag{10}$$

where $\boldsymbol{\Sigma}_{0,1:n}$ is the cross-covariance between $\boldsymbol{s}_0$ and all observation sites $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$. The prediction (10) is the well-known Kriging equation, and it can be thought of as the conditional mean of the Gaussian process at $\boldsymbol{s}_0$ given the observations. In this sense, (10) is the optimal prediction based on the Gaussian likelihood. In this section, we describe the optimal prediction based on the block CL. The composite prediction is fast to compute and easily parallelised. Throughout this section, we will assume the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are known, but in practice we use plug-in values $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{CL}$ and $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_{CL}$.

The computational difficulty encountered with the Kriging prediction (10) is matrix factorisation of a large $n \times n$ matrix. The same challenge occurs for the prediction variance,

$$\mathrm{Var}(\hat{Y}(\boldsymbol{s}_0)) = \Sigma_0 - \boldsymbol{\Sigma}_{0,1:n}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{0,1:n},$$

where Σ_0 is the marginal variance $\mathrm{Var}(Y(\boldsymbol{s}_0))$ of the field at the prediction site.

Consider the task of making predictions at $n_{k0} \geq 1$ unobserved sites, all situated within block k. The first step is to augment the data vector such that $\boldsymbol{Y}_k^a = (\boldsymbol{Y}_{k0}', \boldsymbol{Y}_k')'$. By including $\boldsymbol{Y}_{k0}$ as unobserved data in the CL and setting the derivative of ℓ_{CL} in (6) equal to 0, we obtain the composite predictions $\hat{\boldsymbol{Y}}_{k0}$.

The contribution of the unobserved data $\boldsymbol{Y}_{k0}$ to the CL is given by block terms (k, l), $l \in N_k$, looking both forward and backward in the graph of block-edges. We organise these pairs such that block k is always at the top in every (k, l) block-pair, so that the $(n_{k0} + n_k + n_l) \times (n_{k0} + n_k + n_l)$ precision matrix for $(\boldsymbol{Y}_k'^a, \boldsymbol{Y}_l')'$ is

$$\boldsymbol{Q}_{0kl} = \begin{bmatrix} \boldsymbol{Q}_{0kl}(0) & \boldsymbol{Q}_{0kl}(0,1) & \boldsymbol{Q}_{0kl}(0,2) \\ \boldsymbol{Q}_{0kl}(1,0) & \boldsymbol{Q}_{0kl}(1) & \boldsymbol{Q}_{0kl}(1,2) \\ \boldsymbol{Q}_{0kl}(2,0) & \boldsymbol{Q}_{0kl}(2,1) & \boldsymbol{Q}_{0kl}(2) \end{bmatrix}.$$

The block CL at the unobserved locations is thus

$$\begin{aligned}\ell_{CL}(\boldsymbol{Y}_{k0}) = \sum_{l \neq k} \Big[\text{const} &- \frac{1}{2}(\boldsymbol{Y}_{k0} - \boldsymbol{X}_{k0}\boldsymbol{\beta})'\boldsymbol{Q}_{0kl}(0)(\boldsymbol{Y}_{k0} - \boldsymbol{X}_{k0}\boldsymbol{\beta}) \\ &- (\boldsymbol{Y}_{k0} - \boldsymbol{X}_{k0}\boldsymbol{\beta})'\boldsymbol{Q}_{0kl}(0,1)(\boldsymbol{Y}_k - \boldsymbol{X}_k\boldsymbol{\beta}) \\ &- (\boldsymbol{Y}_{k0} - \boldsymbol{X}_{k0}\boldsymbol{\beta})'\boldsymbol{Q}_{0kl}(0,2)(\boldsymbol{Y}_l - \boldsymbol{X}_l\boldsymbol{\beta}) \Big], \end{aligned} \tag{11}$$

now regarded as a function of $\boldsymbol{Y}_{k0}$, and where the $n_{k0} \times p$ matrix $\boldsymbol{X}_{k0}$ collects the explanatory variables at prediction sites in block k. Because the effect of nearby observations dominates the prediction, in practice we save computational effort by only summing over $l \in N_k$ in (11).

The first and second derivatives of $\ell_{CL}(\boldsymbol{Y}_{k0})$ are easily obtained by differentiating the quadratic form in (11). The first derivative is

$$\frac{d\ell_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{k0}} = -\sum_{l \in N_k} \Big[\boldsymbol{Q}_{0kl}(0)(\boldsymbol{Y}_{k0} - \boldsymbol{X}_{k0}\boldsymbol{\beta}) + \boldsymbol{Q}_{0kl}(0,1)(\boldsymbol{Y}_{k} - \boldsymbol{X}_{k}\boldsymbol{\beta}) + \boldsymbol{Q}_{0kl}(0,2)(\boldsymbol{Y}_{l} - \boldsymbol{X}_{l}\boldsymbol{\beta})\Big]. \tag{12}$$

Setting the derivative (12) equal to 0 gives the block composite prediction $\hat{Y}_{k0} = \boldsymbol{X}_{k0}\boldsymbol{\beta} + \boldsymbol{A}_0^{-1}\boldsymbol{b}_0$, where

$$\begin{aligned} \boldsymbol{A}_0 &= \sum_{l \in N_k} \boldsymbol{Q}_{0kl}(0), \\ \boldsymbol{b}_0 &= -\sum_{l \in N_k} \Big[\boldsymbol{Q}_{0kl}(0,1)(\boldsymbol{Y}_{k} - \boldsymbol{X}_{k}\boldsymbol{\beta}) + \boldsymbol{Q}_{0kl}(0,2)(\boldsymbol{Y}_{l} - \boldsymbol{X}_{l}\boldsymbol{\beta})\Big]. \end{aligned} \tag{13}$$

Since the mean of $\boldsymbol{Y}_k$ is $\boldsymbol{X}_k\boldsymbol{\beta}$, for any k, it is easily seen that (12) is an unbiased estimating function for $\boldsymbol{Y}_{k0}$ by checking that $E\left(\frac{d\ell_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{k0}}\right) = 0$. Here, the expectation is taken over $\boldsymbol{Y}$, including the random $\boldsymbol{Y}_{k0}$.

The asymptotic variance of the block composite prediction is described by a Godambe sandwich;

$$\begin{aligned} \boldsymbol{G}_0(\boldsymbol{Y}_{k0}) &= \boldsymbol{H}_0(\boldsymbol{Y}_{k0})\boldsymbol{J}_0^{-1}(\boldsymbol{Y}_{k0})\boldsymbol{H}_0(\boldsymbol{Y}_{k0}), \\ \boldsymbol{H}_0(\boldsymbol{Y}_{k0}) &= -E\left(\frac{d^2\ell_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{k0}^2}\right),\ \boldsymbol{J}_0(\boldsymbol{Y}_{k0}) = \mathrm{Var}\left(\frac{d\ell_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{k0}}\right). \end{aligned} \tag{14}$$

The prediction variances at locations $\boldsymbol{s}_0$ are the diagonal elements of $\boldsymbol{G}_0^{-1}(\boldsymbol{Y}_{k0})$. The Hessian of (11), required for the computation of (14), is the fixed quantity $\frac{d^2\ell_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{k0}^2} = -\boldsymbol{A}_0$ defined in (13). The variance that defines $\boldsymbol{J}_0(\boldsymbol{Y}_{k0})$ is computed over $\boldsymbol{Y}$, including the random $\boldsymbol{Y}_{k0}$. By introducing the $n_{k0} \times (n_{k0} + n_k)$ matrix $\boldsymbol{B}_{k0} = \left[\sum_{l \in N_k} \boldsymbol{Q}_{0kl}(0), \sum_{l \in N_k} \boldsymbol{Q}_{0kl}(0,1)\right]$ from (12) we get

$$\begin{aligned} \boldsymbol{J}_0(\boldsymbol{Y}_{k0}) = \ & \mathrm{Var}\left(\frac{dl_{CL}(\boldsymbol{Y}_{k0})}{d\boldsymbol{Y}_{0,k}}\right) = \boldsymbol{B}_{k0}\mathrm{Var}(\boldsymbol{Y}_k^a)\boldsymbol{B}_{k0}' \\ & + 2\sum_{l \in N_k} \boldsymbol{B}_{0k}\mathrm{Cov}(\boldsymbol{Y}_k^a, \boldsymbol{Y}_l)\boldsymbol{Q}_{0kl}'(0,2) \\ & + \sum_{l \in N_k}\sum_{l' \in N_k} \boldsymbol{Q}_{0kl}(0,2)\mathrm{Cov}(\boldsymbol{Y}_l, \boldsymbol{Y}_{l'})\boldsymbol{Q}_{0kl'}'(0,2). \end{aligned} \tag{15}$$

The derivations of the sampling properties of the composite prediction contained in this section are computed from a somewhat different standpoint than the analogous derivations for parameter estimation in the previous section. In Section 3.1, the parameters were considered fixed and unknown, while in this section we have considered the prediction variable as a random quantity. This distinction resembles that between confidence intervals and prediction intervals in traditional regression analysis.

We note that if we fix the block boundaries and allow the data density to increase to infinity, the block CL prediction converges to the Kriging prediction, and thus enjoys the same infill asymptotic properties as the BLUP.

4 Computation

4.1 Computing the block composite estimator

Optimisation of the block CL proceeds iteratively, alternately solving for $\hat{\boldsymbol{\beta}}_{CL}$ given $\hat{\boldsymbol{\theta}}_{CL}$, and for $\hat{\boldsymbol{\theta}}_{CL}$ given $\hat{\boldsymbol{\beta}}_{CL}$. While each optimisation with respect to the regression parameters $\boldsymbol{\beta}$ can be expressed analytically (see (8)),

the optimisation for $\boldsymbol{\theta}$ must be done numerically. We define a starting value $\boldsymbol{\theta}_0$ and use Fisher-scoring updates according to

$$\boldsymbol{\theta}_{b+1} = \boldsymbol{\theta}_b - E\left[\frac{\partial^2 \ell_{CL}(\boldsymbol{Y};\boldsymbol{\beta},\boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}^2}\right]^{-1} \frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\beta},\boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}}. \tag{16}$$

The score $\frac{\partial \ell_{CL}(\boldsymbol{Y};\boldsymbol{\beta},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ and the expectation of the second derivative of the block CL can be derived analytically (Section 3.1 and Appendix A). Convergence typically occurs in about 5 Fisher-scoring updates. The expressions in (16) require the derivatives of the covariance function. For most covariance models in common use, the derivatives $\frac{\partial \boldsymbol{\Sigma}_j}{\partial \theta_r}$ are available in closed form. For instance, the exponential covariance function $\Sigma_j(i,i') = \tau^2 I(i = i') + \sigma^2 \exp(-\phi h)$, $h = \|\boldsymbol{s}_i - \boldsymbol{s}_{i'}\|$ has derivatives

$$\frac{\partial \Sigma_j(i,i')}{\partial \sigma^2} = \exp(-\phi h),\ \frac{\partial \Sigma_j(i,i')}{\partial \phi} = -h\sigma^2 \exp(-\phi h),\ \frac{\partial \Sigma_j(i,i')}{\partial \tau^2} = I(i = i').$$

Derivatives of the general Matérn covariance function are also available, but are considerably more complicated (especially with respect to the smoothness parameter ν), as they require derivatives of modified Bessel functions $K_\nu(x)$ (Abramowitz and Stegun, 1964).

An efficient algorithm for the Fisher-scoring update is given in Algorithm 1. Note that many of the derivatives and matrix factorisations are the same for the CL, the score, and the Hessian, and hence can be re-used.

Algorithm 1 Computation for a Fisher scoring update for the block composite likelihood

Require: $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_R)'$, initialise $u_r = 0$ and $H_{rs} = 0, r = 1, \ldots, R, s = r, \ldots, R$.
1: **for** $k = 1$ to $M - 1$ **do**
2: **for** $l \in N_k^{\rightarrow}$ **do**
3: Build and factorise $\boldsymbol{\Sigma}_{kl} = \boldsymbol{L}_{kl}\boldsymbol{L}_{kl}^t$. $\boldsymbol{Q}_{kl} = \boldsymbol{\Sigma}_{kl}^{-1} = \boldsymbol{L}_{kl}^{-t}\boldsymbol{L}_{kl}^{-1}$
4: Compute $\boldsymbol{q}_{kl} = \boldsymbol{Q}_{kl}(\boldsymbol{Y}_{kl} - \boldsymbol{X}_{kl}\boldsymbol{\beta})$
5: **for** $r = 1$ to R **do**
6: Compute $\boldsymbol{W}_{klr} = \boldsymbol{Q}_{kl}\frac{d\boldsymbol{\Sigma}_{kl}}{d\theta_r}$
7: $u_r \leftarrow u_r - \frac{1}{2}\text{trace}(\boldsymbol{W}_{klr}) + \frac{1}{2}q'_{kl}\frac{d\boldsymbol{\Sigma}_{kl}}{d\theta_r}q_{kl}$
8: **for** $s = r$ to R **do**
9: $H_{rs} \leftarrow H_{rs} + \frac{1}{2}\text{trace}(\boldsymbol{W}_{klr}\boldsymbol{W}_{kls})$
10: **end for**
11: **end for**
12: **end for**
13: **end for**
14: **return** $\frac{d\ell_{CL}}{d\boldsymbol{\theta}} = (u_1, \ldots, u_R)'$, $-E\left(\frac{d^2\ell_{CL}}{d\boldsymbol{\theta}_r d\theta_s}\right) = \begin{pmatrix} H_{11} & \cdots & H_{1R} \\ \vdots & \ddots & \vdots \\ H_{1R} & \cdots & H_{RR} \end{pmatrix}$, $\boldsymbol{\theta} = \boldsymbol{\theta} + \boldsymbol{H}^{-1}\boldsymbol{u}$.

4.2 Computational efficiency and parallel computing

To study computational aspects of the block CL approach, we compare computing times for a variety of sites per block, $n_k = c$, and data sizes $n = cM$. Under the assumption of fixed c and increasing n, the computational complexity of the block CL is $O(n)$. This follows since the for-loop goes over $n|N_k^{\rightarrow}|/c$ steps, and at every step the computation time is $O(c^3)$ for the smaller (fixed size c) matrix factorisation. This kind of linear order in n is usually required for massive datasets. It holds for most knot-based models and basis representations such as the predictive process model (Banerjee et al., 2008) and fixed rank Kriging (Cressie and Johannesson, 2008), when the number of knots or basis functions are fixed at a low rank. In contrast, the Fourier approximations (Fuentes, 2007) are $O(n \log n)$, while Gaussian Markov random fields (Lindgren et al., 2011) are $O(n^{3/2})$ for two-dimensional spatial data.

In addition to the order-n computational complexity, the block CL approach has no limit on data size due to computer memory. Since the CL, score, and Hessian computations are sums over independent calculations for

each pair of blocks, the only in-memory information is that pertinent to the current pair. Therefore computer memory imposes a limit on the size of blocks, but not on the size of data.

The block CL approach is highly amenable to parallelisation. First, the CL expression is a sum over independent calculations for each pair of blocks, and these can be performed in parallel. Second, the main computational cost is due to linear algebra subroutines, e.g. matrix decompositions, which are also highly parallelisable (Galoppo et al., 2005; Krüger and Westermann, 2005; Volkov and Demmel, 2008). We experimented with a parallel sum over block pairs and no parallel matrix decompositions, and vice versa. We found that parallelising matrix decomposition and not sums over block pairs lead to greater improvements because of issues with memory allocation. Parallelising both of these computations simultaneously would likely give an even greater computational improvement, especially for small and moderate block sizes, and is an area of future work.

Parallelisation has historically been dominated by CPU clusters which is just a group of linked computers, called nodes. Typical clusters might have 8 or 16 nodes, while modern supercomputers have thousands. The number of nodes determines the maximum speed-up due to parallelisation. These CPU parallelised systems are capable of high parallelisation, but at a high cost for implementation and maintenance. CPU parallelisation has advanced through the introduction of multithreaded CPUs which can efficiently execute multiple threads in parallel on a single CPU. In contrast to the CPU cluster, these multithreaded systems share host CPU resources, e.g. RAM.

More recently the development of highly parallelised general purpose GPUs provides a low cost, highly parallel solution for many scientific computing applications. These affordable video cards ($2000 for the high-end cards we are using) can be used in almost any desktop computer. These video cards provide speedups on the order of 100 or 200-fold for statistical applications (Li and Petzold, 2010; Suchard et al., 2010; Niemi and Wheeler, 2011).

Therefore, we investigate the use of GPUs to accelerate computation and allow for analysis of large data sets. The two approaches we assessed were a MATLAB toolbox called Jacket and CUDA (Compute Unified Device Architecture) C. Both approaches require CUDA-capable NVIDIA GPUs. MATLAB/Jacket utilise a *gfor* routine which can be applied to run the CL sum in parallel. The current CUDA implementation uses parallel routines for matrix factorisation. Computational details are presented in Appendix B.

5 Simulation study

In this section, we present a synthetic data example to study the sampling properties of parameter estimates and predictions using the block CL. In addition, we examine the computational efficiency gains achieved by using a GPU parallel computing environment.

5.1 Inference and prediction

We generate a spatial design with $n = 2,000$ observation sites on a spatial domain $(0,1) \times (0,1)$. The selected design is of a regular plus random infill type (Diggle and Lophaven, 2006). We generate 700 regular points, then select 100 of these at random and draw 10 random points around each of them. The remaining sites are drawn randomly within the unit square.

We compare properties of parameter estimates and predictions for various block sizes of the CL model. With n as small as $2,000$ we can also compare with full likelihood results. Covariates are $\boldsymbol{x}'(\boldsymbol{s}_i) = (1, s_{i1})$, with true regression parameters $\boldsymbol{\beta} = (-1,1)^t$. We use a Matern covariance model with smoothness parameter 3/2, i.e. $\Sigma(i,i') = \tau^2 I(h=0) + \sigma^2(1+\phi h)\exp(-\phi h)$ for distance $h = ||\boldsymbol{s}_i - \boldsymbol{s}_{i'}||$ between two sites $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i'}$. We use a parameterisation on the real line, with log precisions and log range parameter: $\theta_1 = -\log(\sigma^2)$, $\theta_2 = \log(\phi)$, and $\theta_3 = -\log(\tau^2)$. This parameterisation makes the Fisher-scoring optimisation robust. The scale parameters used to generate the data are $\theta_1 = 0$, $\theta_3 = 2$, while the range is set to a short effective spatial correlation ($\theta_2 = 3$), or to a long effective spatial correlation ($\theta_2 = 2$).

The results of mean square error (MSE), asymptotic relative efficiency, coverage probabilities and computing time are given in Table 1 (short effective spatial correlation) and Table 2 (long effective spatial correlation). We compare results of the full likelihood (L) and the block CL. For the CL, we use regular blocks (Figure 1(a)) of lattice size $9 = 3^2$, $25 = 5^2$, $49 = 7^2$ and $100 = 10^2$. The results are averaged over $1,000$ replicates of $n = 2,000$ data for the same spatial design. The prediction results are averaged over 500 prediction sites (not included in

Table 1: Synthetic data: $n = 2,000$ and a Matern (3/2) covariance function with **short** effective spatial correlation. The asymptotic CL variances, relative to full likelihood, are shown in parantheses. Results are averages over $1,000$ replicates.

	L	CL9	CL25	CL49	CL100
MSE $\hat{\beta}_1$ (Asymp. RE)	0.05 (1)	0.05 (0.81)	0.06 (0.77)	0.06 (0.74)	0.06 (0.79)
MSE $\hat{\beta}_2$ (Asymp. RE)	0.17 (1)	0.21 (0.82)	0.24 (0.77)	0.23 (0.74)	0.23 (0.79)
MSE $\hat{\theta}_1$ (Asymp. RE)	0.014 (1)	0.018 (0.80)	0.019 (0.78)	0.019 (0.80)	0.018 (0.85)
MSE $\hat{\theta}_2$ (Asymp. RE)	0.0036 (1)	0.0043 (0.80)	0.0048 (0.77)	0.0052 (0.76)	0.0054 (0.76)
MSE $\hat{\theta}_3$ (Asymp. RE)	0.0007 (1)	0.0008 (0.86)	0.0008 (0.88)	0.0008 (0.88)	0.0008 (0.87)
Coverage (0.95) $\hat{\beta}_1$	0.96	0.95	0.96	0.96	0.96
Coverage (0.95) $\hat{\beta}_2$	0.93	0.92	0.93	0.92	0.92
Coverage (0.95) $\hat{\theta}_1$	0.93	0.92	0.92	0.91	0.91
Coverage (0.95) $\hat{\theta}_2$	0.94	0.94	0.93	0.91	0.91
Coverage (0.95) $\hat{\theta}_3$	0.95	0.95	0.95	0.95	0.94
MSPE (Mean Asymp. RE)	193 (1)	195 (1)	198 (1)	200 (0.99)	204 (0.97)
Mean coverage (0.95)	0.95	0.95	0.95	0.95	0.95
Computing time (sec), no GPU	76	39	16	12	13

the $n = 2,000$ data) per replicate. The asymptotic relative efficiency is defined by the ratio of the asymptotic variances obtained by the Hessian (likelihood) and the Godambe sandwich (CL).

The increase in MSE for the block CL models is small over the full likelihood model, in particular for predictions (MSPE). For all models the MSEs are higher for long effective spatial correlation (Table 2), while the MSPE is smaller in this case. The relative MSE increase for the CL is largest for the long correlation case (Table 2).

The asymptotic relative efficiencies (in parantheses) show that standard deviations of the parameter estimates are larger for the CL models. Moreover, these asymptotic relative efficiencies are larger for a long correlation range, except for the spatial correlation parameter θ_2. For the 100 blocks case, with only 20 points per block on average, the asymptotic efficiencies might not be very accurate. There is also some Monte Carlo error over the $1,000$ replicates. For all block CL models, the prediction efficiency is near 1.

The coverage probabilities are computed by, for every replicate, checking if the true value is within 1.96 standard deviations of the estimate. The results are averaged over all $1,000$ replicates. The coverages are about the same for the full likelihood and the block CL models. They tend to be a little smaller than the nominal level in our case with data size $n = 2,000$, especially for the long correlation case (Table 2), and for θ_1 and θ_2. Notably, the prediction coverages are excellent for all models.

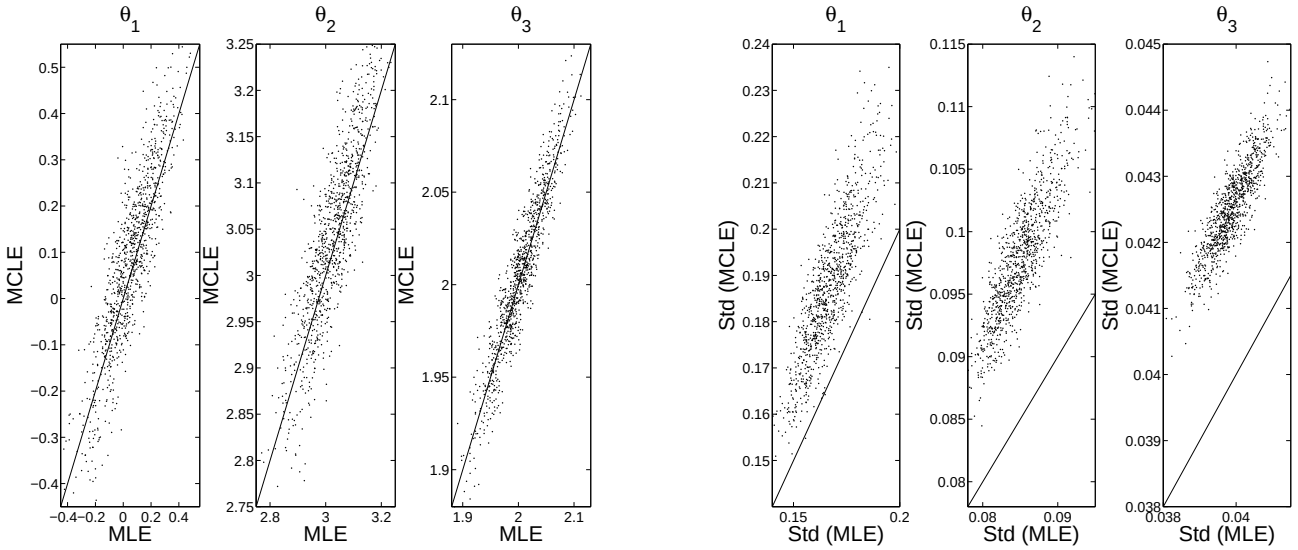
The bottom row shows the computing times required for optimisation, asymptotic variance calculation, and predictions, using no parallel implementation. This computing time is reduced by a factor 7 when using the 49 or 100 block CL models instead of the full likelihood. The computation time does not go down from 49 to 100 blocks because the relative computing time of looping over pairs of blocks becomes as large as the time required for matrix factorisation.

We next study the performance by cross-plotting the CL and likelihood results. In Figure 2(a) we show the parameter estimates for log precision (left), log range (middle) and log nugget precision (right) using maximum likelihood (x-axis) versus that of CL with 100 blocks (y-axis). These plots show results for the situation with a short effective spatial range. The points fall near the straight line with unit slope, showing that the maximum CL estimates are very close to the MLEs. In Figure 2(b) we similarly show the asymptotic standard deviations based on the Hessian of the likelihood and the Godambe sandwich for the 100-block CL model. The points are above the straight line, visualising the slight decrease in efficiency when using the CL with 100 blocks. For θ_1 and θ_2 there is a tendency of greater loss of efficiency for larger values, while θ_3 has a more constant decrease in efficiency, which is related to a simpler sandwich expression for the nugget term.

Similarly, Figure 3(a) shows cross-plots of predictions, which fall tightly along the straight line with unit slope. Figure 3(b) shows the prediction standard errors obtained by Kriging equations (x-axis) or the sandwich

Table 2: Synthetic data: $n = 2,000$ and a Matern (3/2) covariance function with **long** effective spatial correlation. The asymptotic CL variances, relative to full likelihood, are shown in parantheses. Results are averages over 1,000 replicates.

	L	CL9	CL25	CL49	CL100
MSE $\hat{\beta}_1$ (Asymp. RE)	0.14 (1)	0.13 (0.87)	0.14 (0.83)	0.15 (0.84)	0.15 (0.89)
MSE $\hat{\beta}_2$ (Asymp. RE)	0.42 (1)	0.51 (0.86)	0.57 (0.82)	0.59 (0.81)	0.61 (0.85)
MSE $\hat{\theta}_1$ (Asymp. RE)	0.10 (1)	0.15 (0.81)	0.19 (0.80)	0.21 (0.82)	0.23 (0.87)
MSE $\hat{\theta}_2$ (Asymp. RE)	0.020 (1)	0.029 (0.77)	0.040 (0.72)	0.051 (0.70)	0.065 (0.68)
MSE $\hat{\theta}_3$ (Asymp. RE)	0.0006 (1)	0.0006 (0.87)	0.0006 (0.88)	0.0006 (0.88)	0.0006 (0.88)
Coverage (0.95) $\hat{\beta}_1$	0.92	0.93	0.92	0.91	0.91
Coverage (0.95) $\hat{\beta}_2$	0.87	0.85	0.86	0.85	0.84
Coverage (0.95) $\hat{\theta}_1$	0.83	0.79	0.76	0.73	0.72
Coverage (0.95) $\hat{\theta}_2$	0.86	0.82	0.79	0.75	0.72
Coverage (0.95) $\hat{\theta}_3$	0.94	0.94	0.93	0.94	0.94
MSPE (Mean Asymp. RE)	151 (1)	153 (1)	155 (0.99)	157 (0.97)	161 (0.95)
Mean coverage (0.95)	0.95	0.95	0.95	0.95	0.95
Computing time (sec), no GPU	76	40	17	12	12

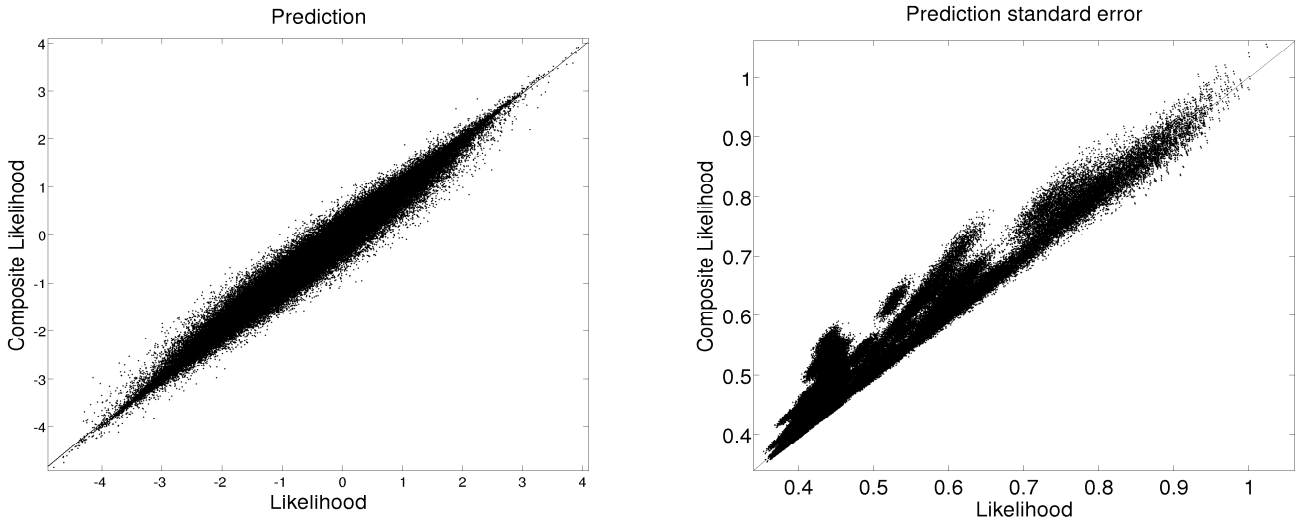


(a) (b)

Figure 2: Synthetic data (**short** effective spatial correlation): (a) Comparison of maximum likelihood estimates (x-axis) and maximum CL estimates using 100 blocks (y-axis). (b) Comparison of asymptotic standard deviation of maximum likelihood estimates (x-axis) and maximum CL estimates using 100 blocks (y-axis). The results are computed over 1,000 replicates of data at $n = 2,000$ observation sites. θ_1 is a precision parameter, θ_2 is a range parameter and θ_3 is a nugget precision parameter.

calculation (y-axis). The points are near the straight line, indicating that the CL model does not lose much prediction efficiency. The clusters of points off above the $y = x$ line in Figure 3(b) correspond to predictions near the block boundaries. The increased prediction variance in these regions is caused by edge effects, where the CL ignores some of the dependency effects outside the block-pair interactions. The sandwich standard errors correctly account for this effect.

The results show that the statistical efficiency decreases moderately with increased number of blocks. For prediction purposes the effect is very small. We also tried other covariance functions, and the type of correlation model does not seem to affect the results much, while the correlation length has the effects indicated by Table



(a) (b)

Figure 3: Synthetic data (**short** effective spatial correlation): (a) Comparison of predictions based on the full likelihood model (x-axis) and the composite likelihood model using 100 blocks (y-axis). (b) Comparison of asymptotic prediction standard deviations using full likelihood model (x-axis) and the CL model using 49 blocks (y-axis). The number of prediction sites is 500, and results are computed over $1,000$ replicates of data at $n = 2,000$ observation sites.

1 and 2. In practice the selection of a blocking scheme might depend on the spatial correlation range and the design of observation and prediction sites. For both parameter estimation and prediction it might be helpful to use overlapping blocks or to include some points outside the block (similar to suggestions in Stein et al., 2004). We have focused on disjoint blocks here, but extensions are possible.

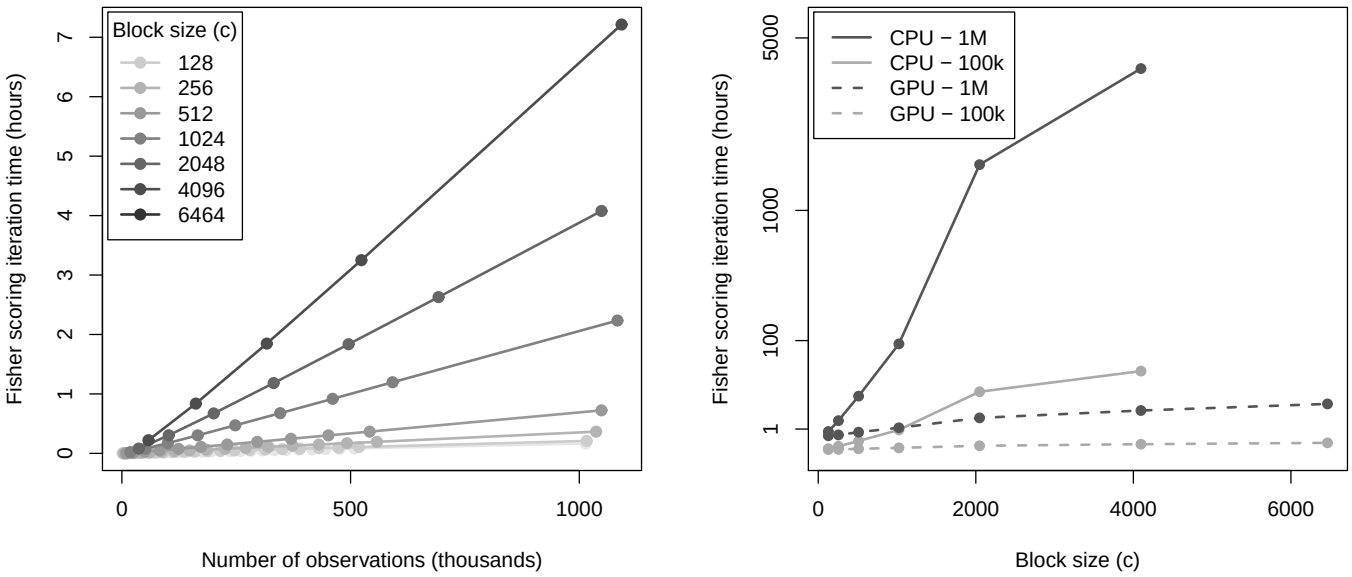
5.2 Computing time

To assess the computational aspects of the CL algorithm and parallel computing environments, we perform simulation studies with varying block sizes and numbers of observations and recorded the total run time. For easier comparison we fix the number of points in the blocks $n_k = c$, for all k. This entails a spatial design with c random sites in every block. We study performance for various c and data sizes n. For simplicity we use regular quadratic grid blocks covering the spatial domain $(0, 1) \times (0, 1)$. We use a Matern covariance function with smoothness parameter 3/2 and an effective correlation range of about 0.4.

First, we instructed MATLAB/Jacket to utilise the `gfor` loop. The best performance gain we attained using `gfor` was less than ten-fold relative to MATLAB code without Jacket. It may be possible to increase these gains with a thorough understanding of how Jacket interfaces with the GPU, and the associated memory allocation.

Instead, we study the computational gain when using parallel computing for matrix decomposition using CUDA C. We next show the resulting CUDA C computing times for the Fisher-scoring algorithm on synthetic datasets of varying sizes and CL models. The points in blocks c ranges from 128 to 4096 on the CPU, the largest block size available on a 32-bit machine, and 6464 on the GPU, the largest available block size within the GPU memory constraints, and for different dimension n. Thus, using quadratic regular grids of blocks, the smallest dataset has $128 \times 3 \times 3 = 1152$ observations and the largest dataset has $6464 \times 13 \times 13 = 1,092,416$ observations.

Figure 4(a) shows GPU computation times as a function of data size for different block sizes. This display clearly shows the linear scaling of the algorithm with data size for fixed number of points c per block. Figure 4(b) shows the associated computation times for fixed data sizes and varying block sizes on both the CPU and GPU. The computation times are plotted on a cube-root scale to emphasize that the Fisher scoring algorithm has cubic complexity in c, the number of points per block. The speed-up when running the equivalent algorithm on the GPU compared with the CPU is linear in block size and essentially constant within a given block size. The speedup was 1.4-fold at $c = 128$, 13-fold at $c = 512$, 29-fold at $c = 1024$, and 112-fold at $c = 4096$. In



(a) (b)

Figure 4: (a) Computation times for a single Fisher scoring iteration on a NVIDIA C2050 GPU as a function of data size for a variety of block sizes. (b) Computation times, plotted on a cube-root scale, on both the CPU and GPU for a single Fisher scoring iteration as a function of block size for specific data sizes.

our experience, the Fisher scoring algorithm takes about 5 iterations to reach convergence. This speed-up would allow a one million observation dataset to be fully analyzed in half a day using $c = 1024$ or about two days using $c = 4096$. Of course, one could also use parameter estimates from smaller block sizes as starting values for larger blocks, etc.

A more sophisticated implementation on the GPU would allow more speed-up for the CL model, utilising a parallel for-loop and running matrix decomposition in parallel. This is future work. Good block design might reasonably depend on two competing factors: computing time and statistical efficiency. On one hand, computation time increases cubicly with block size, so to obtain fast results, smaller blocks are preferable. On the other hand, increasing block size increases the fidelity of the CL to the full likelihood as shown in Tables 1 and 2.

6 Real data examples

To study the performance of the block CL in real-world settings, we test it on one dataset from the mining industry and one of total column ozone generated from satellite retrievals. The block CL model is compared to the predictive process model and fixed rank Kriging.

6.1 Mining dataset

We study a joint frequency dataset acquired in an iron mine in Norway. Clusters of joints represent zones of weakness in the rock mass. Data are useful for predicting the stability requirements in the mine and for avoiding rock-fall. Joint frequency data are acquired from boreholes. The mining company collects this type of data before developing a new part of the mine. The raw data are aggregated to 4m blocks along these boreholes, and the total number of measurements is $n = 11,107$. Ellefmo and Eidsvik (2009) analysed a subset of this dataset.

In Figure 5(a) we display the three dimensional locations of the measurements. The boreholes are recognised as contiguous measurements that line up in the (north, east and depth) coordinates. The joint frequency data

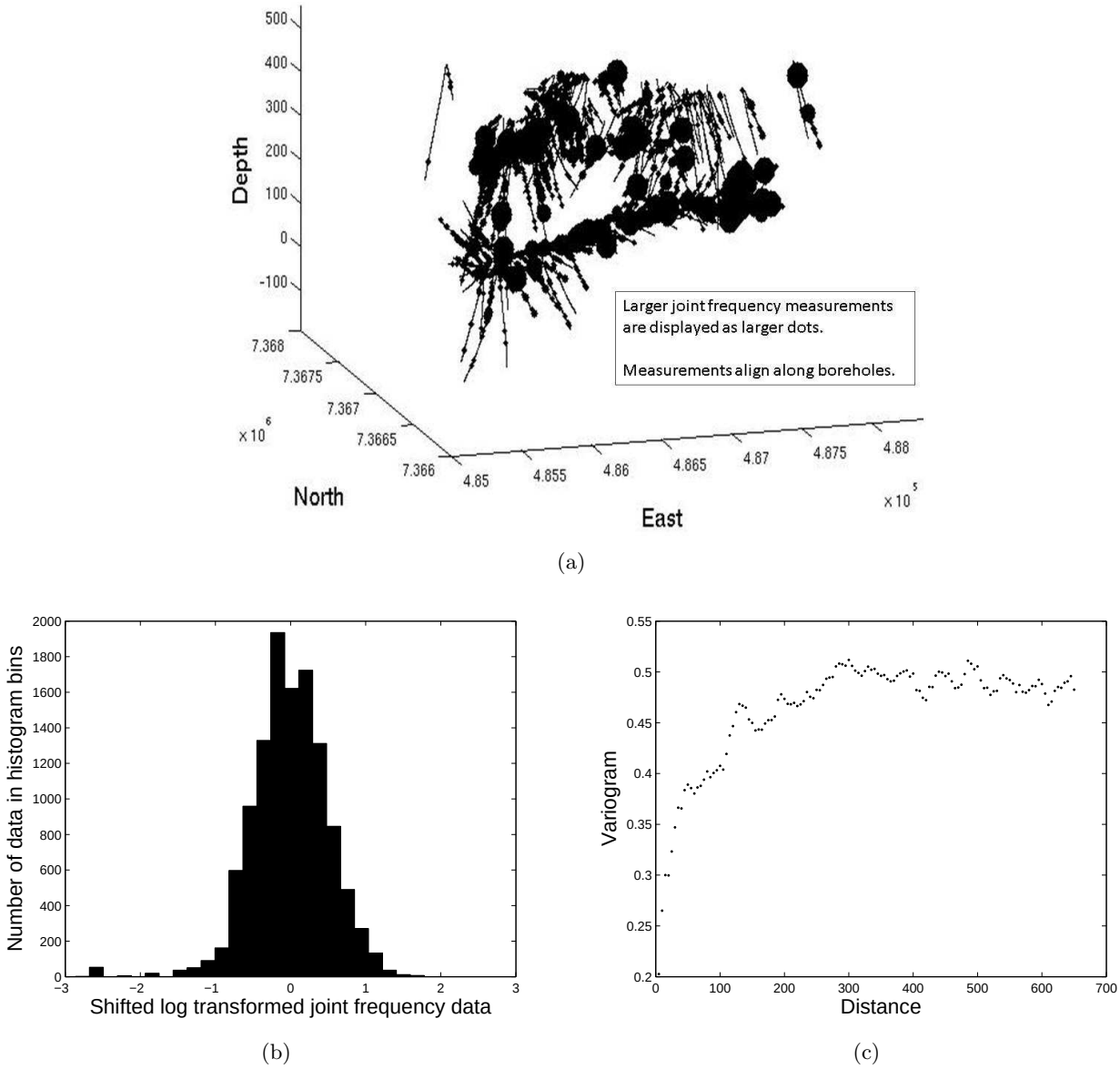


Figure 5: Mining data: (a) Data sites in a Norwegian iron mine. The data size is $n = 11,107$, collected in about 200 boreholes. (b) Histogram of shifted log transformed data of joint frequencies. (c) Empirical variogram.

are transformed using a shifted log transformation, and the resulting data are shown in Figure 5(b). The mean is transformed to 0, and we apply a standard Gaussian geostatistical model to these transformed data. Figure 5(c) displays the empirical variogram, showing that the nugget is about $\tau^2 = 0.2/2 = 0.32^2$, the correlation range is about 100 m, and the variance of the structured effect is about $\sigma^2 = (0.5 - 0.2)/2 = 0.39^2$.

We use the block CL model with different block sizes. The blocks are constructed by a Voronoi / Delaunay tesselation adapted to the (north,east) coordinates of the data, with cells extending for all depths. The tesselation is made by random sampling, without replacement, among all data sites, which on average gives smaller area blocks where sampling locations are dense. We compare the CL results with the predictive process model (Banerjee et al., 2008; Finley et al., 2009), a dimension-reduction technique using a fixed set of knots. The predictive process is a linear (Kriging) combination of the observed data at the knots. Here, we draw the knot locations at random, without replacement, from among the data locations.

Table 3 shows the parameter estimates, the average MSPE and coverage probabilities for a hold-out set of 1000 prediction sites. We compare three common covariance functions: the exponential model specified by $\Sigma(h) = \sigma^2 \exp(-\phi h) + \tau^2 I(h = 0)$, Cauchy(3) which is $\Sigma(h) = \sigma^2(1 + \phi h)^{-3} + \tau^2 I(h = 0)$, and Matern(3/2) with $\Sigma(h) = \sigma^2(1 + \phi h)\exp(-\phi h) + \tau^2 I(h = 0)$. The parameter estimates are very similar for all three CL models,

Table 3: Mining data: Parameter estimates, mean square prediction error (MSPE) and coverage probabilities for prediction distributions. The different columns correspond to different number of blocks for the CL model and different knot sizes for the predictive process (PP) models.

		CL, 200	CL, 40	CL, 10	PP, 1000	PP, 1500
Exponential	$\hat{\sigma}$	0.42	0.42	0.42	0.44	0.45
	$\hat{\phi}$	0.031	0.030	0.028	0.013	0.015
	$\hat{\tau}$	0.30	0.30	0.30	0.32	0.31
	MSPE	145	144	144	182	171
	Pred cov (0.95)	0.95	0.95	0.95	0.94	0.94
Cauchy	$\hat{\sigma}$	0.41	0.42	0.42	0.45	0.47
	$\hat{\phi}$	0.013	0.013	0.012	0.004	0.004
	$\hat{\tau}$	0.29	0.29	0.29	0.34	0.33
	MSPE	145	144	143	182	174
	Pred cov (0.95)	0.95	0.95	0.95	0.95	0.95
Matern (3/2)	$\hat{\sigma}$	0.38	0.39	0.39	0.39	0.41
	$\hat{\phi}$	0.074	0.073	0.069	0.033	0.036
	$\hat{\tau}$	0.33	0.33	0.33	0.35	0.35
	MSPE	148	148	147	173	168
	Pred cov (0.95)	0.95	0.95	0.95	0.94	0.94

but different between the CL and predictive process models. In particular, the range parameter ϕ is smaller for predictive process models. This implies a larger effective spatial correlation, imposing a smoother process. To some extent, the predictive process models compensate for this with larger estimated variance terms.

The MSPE is clearly smaller for the CL models than for the predictive process models. Even with 1500 knots, the MSPE for the predictive process model is 15% larger than the CL models. There are small differences between the various spatial covariance functions. The coverage probabilities are very good for all models considered. The computation times required for obtaining the results of Table 3 range from a few seconds to a few minutes. The fastest is the CL model with 200 blocks, which takes about 10 seconds using no GPU resources, while the slowest model is the predictive process with 1500 knots, which takes a few minutes.

6.2 Total column ozone dataset

We next analyze total column ozone (TCO) data acquired from an orbiting satellite mounted with a passive sensor registering backscattered light. The dataset we consider here was previously analyzed by Cressie and Johannesson (2008), and is displayed in Figure 6. The dataset consists of $n = 173,405$ measurements. Cressie and Johannesson (2008) used fixed rank Kriging (FRK) in their analysis. This approach is based on a basis representation of the spatial Gaussian process. They use 396 local bisquare basis functions at various resolutions recovered from a discrete global grid (Sahr, 2011). The computation time for FRK is $O(n)$, which is highly desirable, but the trade-off is that the low-rank basis representation imposes possibly unrealistic smoothness on the results.

We compare the block CL models using 15 and 24 latitude / longitude blocks, and one based on the discrete global grid of resolution 3 consisting of 272 blocks (Sahr, 2011). We use a fixed and constant mean β_1, and a Cauchy (3) type covariance function. The block CL parameter estimates are very similar: $(\hat{\sigma}^2, \hat{\phi}, \hat{\tau}^2)$ is $(71^2, 0.030, 4.8^2)$ using 15×15 blocks, $(65^2, 0.033, 4.7^2)$ using 24×24 blocks, and $(64^2, 0.032, 4.6^2)$ for the discrete global grid.

We predict on a 180×288 grid, corresponding to the so-called NASA 'level 2' data product. The latitude ranges from -89.5 to 89.5 in 1° steps, while longitude ranges from -179.375 to 179.375 in 1.25° steps. In total, this entails $51,840$ prediction sites. Prediction maps of TCO for the three different block CL models are nearly indistinguishable. Figure 7(a) shows the prediction map of TCO using 15×15 blocks. The marginal prediction standard deviations of TCO are displayed in Figure 7(b).

We notice that the prediction standard deviations are much higher near the arctic because there is no data

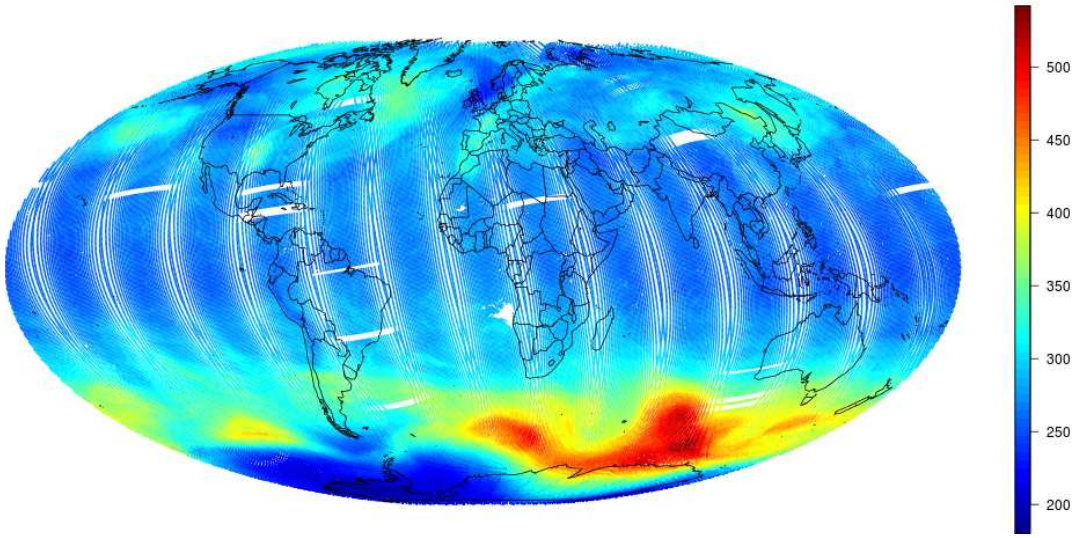


Figure 6: Total column ozone data: Number of measurements is $n = 173,405$. Note the patches of missing data, and the denser sampling of data near satellite orbits. There is less data at the poles due to limited reflection.

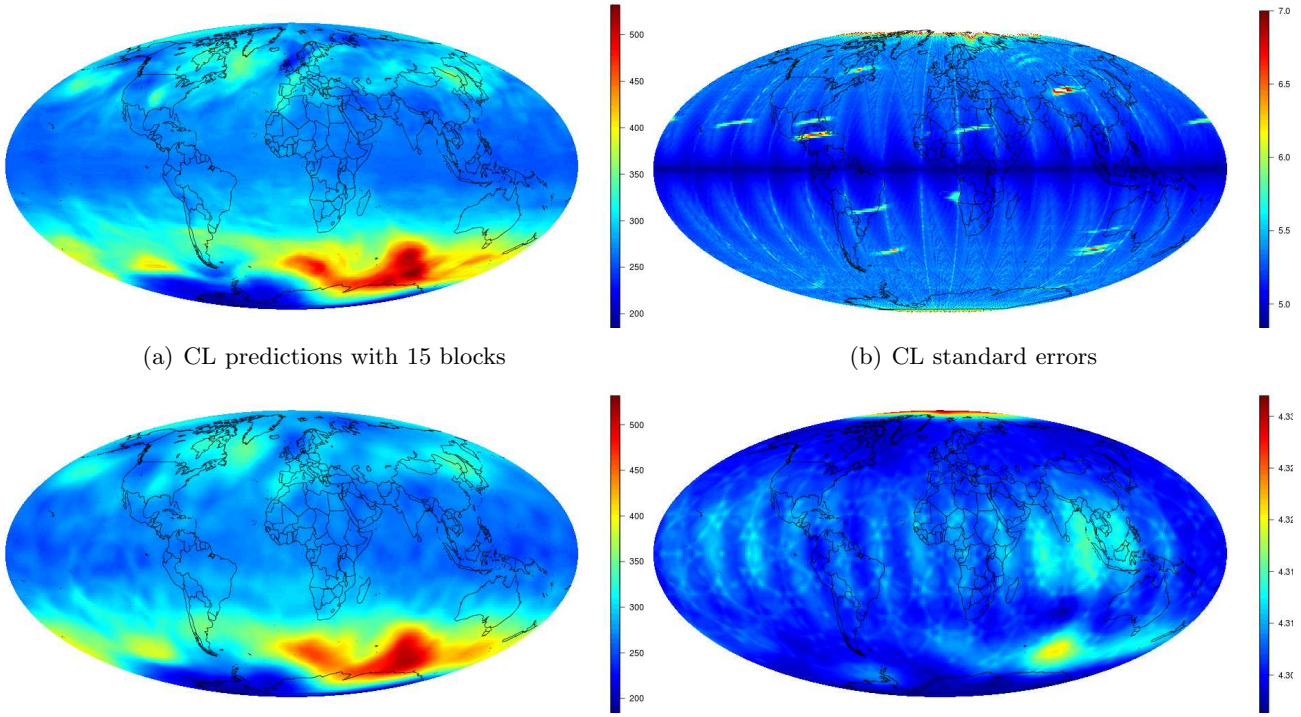
Table 4: Total Column Ozone data: Mean square prediction error (MSPE) and coverage probabilities (95 %) for a 25,000 hold-out set. The results are for fixed rank Kriging (FRK), composite likelihood model with 15×15 (CL, reg15), 24×24 blocks (CL, reg 24), and for a resolution 3 discrete global grid (CL, dgg).

	FRK	CL, reg 15	CL, reg 24	CL, dgg
MSPE	88.0	28.1	28.2	28.5
Pred cov (0.95)	0.71	0.96	0.95	0.96

there. We further note the increased estimated uncertainty in regions of missing data, and light-coloured lines going south-southwest, where there is less dense satellite coverage. In addition, there are visible artifacts of the 15×15 block CL model in the prediction standard deviations. These regions of increased estimated uncertainty where data is lacking and on the border of spatial blocks are desirable—indeed, they indicate that the sandwich variance calculations are correctly accounting for sparse data and block boundary effects. As is also desirable, these block boundary effects are not seen on the prediction map.

We next compare the CL with FRK results. Here, we follow Cressie and Johannesson (2008) in using multi-resolution bisquare basis functions centered at resolution 1, 2 and 3 of the discrete global grid. Resolution 4 of the discrete global grid is used to construct the binning for the moment-based parameter estimation approach (Cressie and Johannesson, 2008). Figure 7 shows the FRK predictions (c) and prediction standard deviations (d) with the nugget effect properly accounted for. Notably, the predictions obtained by FRK are much smoother than the block CL results. Moreover, the estimated prediction standard deviations are smaller for FRK and vary less around the globe. The patches of missing data are not visible in Figure 7(d). Similar to the block edge effects in the CL model, the locations of the basis functions are easily seen as artifacts in the estimated FRK standard error map. In contrast to the CL, these basis function artifacts seem to be the dominant feature in the FRK standard error estimates.

To compare prediction and coverage accuracy, we use a hold-out set of 25,000 randomly-selected data locations around the globe. We estimate the model parameters based on the remaining data and predict at the hold-out locations. Table 4 shows the comparison of the block CL and FRK. All block CL models obtain coverages close to the nominal rate and have similar prediction error. FRK shows a much larger prediction error and it under-covers conspicuously. These results show that for this dataset, the low-dimensional representation in FRK is over-smoothing. One could improve the FRK results by adding more basis functions, with added computation time. The computation time, using no GPU resources, is 5-10 minutes for FRK and around an hour for the various CL models.



(a) CL predictions with 15 blocks

(b) CL standard errors

(c) FRK predictions

(d) FRK standard errors

Figure 7: Total Column Ozone data: Top displays are based on the CL model with 15×15 regular longitude and latitude blocks. Bottom displays are based on fixed rank Kriging. Predictions (a & c) and prediction standard deviations (b & d).

7 Closing remarks

In this paper we use a block composite likelihood model for parameter estimation and prediction in large Gaussian spatial models. The properties of the composite likelihood are well-understood in the context of parameter estimation. Here we also present a method for spatial prediction using the block composite likelihood.

We have shown through a simulation study that the block composite likelihood performs well for reasonably-sized blocks, especially for spatial prediction. The required computation time is reduced considerably relative to likelihood-based calculations using the divide and conquer strategy inherent in the composite likelihood. We recommend testing results with a couple of choices of block sizes (hundreds to thousands sites per block) and blocking designs. The optimal blocking strategy would depend on the spatial correlation and the design of data points. Note that the block composite likelihood model is not restricted to disjoint spatial blocks like we have used for illustration here.

We tested the block composite likelihood on one large dataset from the mining industry ($n = 11,107$) and one massive dataset from satellite measurements ($n = 173,405$). For these datasets we compared the block composite likelihood with predictive process models and with fixed rank Kriging, both of which rely on knot or basis representations for reducing the computational burden of matrix factorisations. The composite likelihood provides better results in terms of both mean square errors and coverage probabilities in the examples we considered.

In addition to increases in accuracy of prediction and uncertainty quantification, the block CL methods we have described have other advantages over FRK and predictive process approaches. What we have not highlighted is that unlike FRK and predictive processes, the CL effectively separates modeling and computational considerations. That is, with the CL one may choose whatever covariance model captures the important features of the data and proceed from there, obtaining asymptotically unbiased estimates of the spatial covariance parameters for the chosen model. In contrast, for low-rank methods, the covariance model itself is induced by computational compromises, choice and number of basis functions in the case of FRK, and knot placement and

density in the case of predictive processes. For these methods, computational and modeling considerations are inexorably connected.

We implemented parallel versions of the block CL model on a GPU. A parallelisation of the matrix decomposition seems more promising than running the CL for loop over block pairs in parallel, given the current software for GPU parallelisation. For moderate to large block sizes this parallel implementation gives two-three digit speed-ups, on top of the speed-up achieved by the CL construction. A topic for future work is to tailor the distribution of block data to the GPU for maximum reduction of the computing time, in a software package. CPU and GPU examples of code are available from the authors.

In this paper we considered only spatial Gaussian processes. In future work we aim to look at spatio-temporal processes as well. For example, the satellite data in Section 6.2 is from just one day of retrievals. It is more useful to analyze these types of data over several days. Another extension involves non-stationary processes. For instance, we have started to look at seismic data ($n \sim 1,000,000$), which typically exhibit larger variability or higher correlations in layered parts of the earth. A Bayesian analysis using the composite likelihood model could also be envisioned. Data dimensions will likely become even larger in the future. We foresee a larger future interest in $O(n)$ approximations for spatial and spatio-temporal applications, as well as in the use of parallel computing environments.

8 Acknowledgments

We thank the Statitistical and Applied Mathematical Sciences Insitute (SAMSI) for support during the program on space-time analysis (2009-2010). We thank NVIDIA for supporting us with graphics cards. Rana Gruber provided the joints data, while Noel Cressie and Gardar Johannesson made the TCO data acquired by NASA available.

A Score function and Hessian

Let the response $\boldsymbol{Y}_j = (\boldsymbol{Y}'_k, \boldsymbol{Y}'_l)'$ at block pair k, l be a zero mean Gaussian vector with covariance matrix $\boldsymbol{\Sigma}_j$. The score is expressed as

$$\frac{d\ell_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{d\theta_r} = \sum_j \Big[-\frac{1}{2}\frac{d\log|\boldsymbol{\Sigma}_j|}{d\theta_r} - \frac{1}{2}\frac{d(\boldsymbol{Y}'_j\boldsymbol{\Sigma}_j^{-1}\boldsymbol{Y}_j)}{d\theta_r}\Big].$$

The derivative of the log determinant and the quadratic form are

$$\begin{aligned}
\frac{d\log|\boldsymbol{\Sigma}_j|}{d\theta_r} &= \frac{1}{|\boldsymbol{\Sigma}_j|}|\boldsymbol{\Sigma}_j|\text{trace}\left(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\right) = \text{trace}\left(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\right), \\
\frac{d(\boldsymbol{Y}'_j\boldsymbol{Q}_j\boldsymbol{Y}_j)}{d\theta_r} &= -\boldsymbol{Y}'_j\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j\boldsymbol{Y}_j.
\end{aligned}$$

Here, the precision matrix is $\boldsymbol{Q}_j = \boldsymbol{\Sigma}_j^{-1}$. The score function then becomes

$$\frac{dl_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{d\theta_r} = \sum_j -\frac{1}{2}\text{trace}\left(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\right) + \frac{1}{2}\boldsymbol{Y}'_j\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j\boldsymbol{Y}_j. \tag{17}$$

The Godambe sandwich requires the variance of this score. We can ignore the non-random trace terms. The computation involves several variances and covariances of quadratic forms $\boldsymbol{Y}'\boldsymbol{B}\boldsymbol{Y}$. We have

$$\text{Var}(\boldsymbol{Y}'\boldsymbol{B}\boldsymbol{Y}) = 2\text{trace}(\boldsymbol{B}\boldsymbol{\Sigma}\boldsymbol{B}\boldsymbol{\Sigma}), \quad \text{Cov}(\boldsymbol{Y}'\boldsymbol{B}_r\boldsymbol{Y}, \boldsymbol{Y}'\boldsymbol{B}_s\boldsymbol{Y}) = 2\text{trace}(\boldsymbol{B}_r\boldsymbol{\Sigma}\boldsymbol{B}_s\boldsymbol{\Sigma}).$$

Define next $\boldsymbol{B}_{jr} = \boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j$. The variance of the score becomes

$$\begin{aligned}
\boldsymbol{J} &= \text{Var}\left(\frac{dl_{CL}}{d\theta_r}\right) = \text{Var}(\sum_j \frac{1}{2}\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j) = \sum_j\sum_{j'}\frac{1}{4}\text{Cov}(\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j, \boldsymbol{Y}'_{j'}\boldsymbol{B}_{j'r}\boldsymbol{Y}_{j'}) \\
&= \frac{1}{4}\sum_j \text{Var}(\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j) + \frac{1}{2}\sum_j\sum_{j'>j}\text{Cov}(\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j, \boldsymbol{Y}'_{j'}\boldsymbol{B}_{j'r}\boldsymbol{Y}_{j'}).
\end{aligned} \tag{18}$$

The sum of the variance terms is directly available:

$$\frac{1}{4}\sum_j \mathrm{Var}(\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j) = \frac{1}{2}\sum_j \mathrm{trace}\left(\boldsymbol{B}_{jr}\boldsymbol{\Sigma}_j\boldsymbol{B}_{jr}\boldsymbol{\Sigma}_j\right) = \frac{1}{2}\sum_j \mathrm{trace}\left(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\right),$$

when inserting for $\boldsymbol{B}_{jr}$. This is the same as the negative expected Hessian in (7). Hence, if we just use the variance parts for the computation of $\boldsymbol{J}$, we get $\boldsymbol{J} = \boldsymbol{H}$, which gives under-estimation of the variance.

For moderate-size datasets we can compute the covariance in (18) for all block variables directly. For larger datasets it is convenient to look only at neighbouring edges, and introduce $\boldsymbol{Y}_{jj'} = (\boldsymbol{Y}'_j, \boldsymbol{Y}'_{j'})'$. Whenever edges j and j' have a common node, two of the sub-blocks of $\boldsymbol{Y}_{jj'}$ are copies. The quadratic forms are

$$\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j = \boldsymbol{Y}'_{jj'}\tilde{\boldsymbol{B}}_{jr}\boldsymbol{Y}_{jj'}, \quad \boldsymbol{Y}'_{j'}\boldsymbol{B}_{j'r}\boldsymbol{Y}_{j'} = \boldsymbol{Y}'_{jj'}\tilde{\boldsymbol{B}}_{j'r}\boldsymbol{Y}_{jj'},$$

where $\tilde{\boldsymbol{B}}_{jr}$ is 0 except at the upper left block containing $\boldsymbol{B}_{jr}$, while $\tilde{\boldsymbol{B}}_{j'r}$ is 0 except in the lower right block containing $\boldsymbol{B}_{j'r}$. We let $\boldsymbol{\Sigma}_{jj'} = \mathrm{Var}(\boldsymbol{Y}_{jj'})$. Then,

$$\begin{aligned}
\mathrm{Cov}(\boldsymbol{Y}'_j\boldsymbol{B}_{jr}\boldsymbol{Y}_j, \boldsymbol{Y}'_{j'}\boldsymbol{B}_{j'r}\boldsymbol{Y}_{j'}) &= \mathrm{Cov}\left(\boldsymbol{Y}'_{jj'}\tilde{\boldsymbol{B}}_{jr}\boldsymbol{Y}_{jj'}, \boldsymbol{Y}'_{jj'}\tilde{\boldsymbol{B}}_{j'r}\boldsymbol{Y}_{jj'}\right) \\
&= 2\mathrm{trace}\left(\tilde{\boldsymbol{B}}_{jr}\boldsymbol{\Sigma}_{jj'}\tilde{\boldsymbol{B}}_{j'r}\boldsymbol{\Sigma}_{jj'}\right) \\
&= 2\mathrm{trace}\left(\boldsymbol{B}_{jr}\boldsymbol{\Sigma}_{jj'}(1,2)\boldsymbol{B}_{j'r}\boldsymbol{\Sigma}_{jj'}(2,1)\right),
\end{aligned}$$

where only the cross-covariance $\mathrm{Cov}(\boldsymbol{Y}_j, \boldsymbol{Y}_{j'}) = \boldsymbol{\Sigma}_{jj'}(1,2)$ enters the equation.

This exercise can similarly be done for the off-diagonal entries of $\boldsymbol{J}$. In summary, we get entries $J(r,s)$, $r,s, = 1,\ldots,R$ given by

$$\begin{aligned}
\mathrm{Cov}\left(\frac{dl_{CL}}{d\theta_r}, \frac{dl_{CL}}{d\theta_s}\right) &= \frac{1}{2}\sum_j \mathrm{trace}\left(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\right) \\
&+ \sum_j\sum_{j'>j} \mathrm{trace}\left(\boldsymbol{B}_{jr}\boldsymbol{\Sigma}_{jj'}(1,2)\boldsymbol{B}_{j's}\boldsymbol{\Sigma}_{jj'}(2,1)\right).
\end{aligned}$$

In practice we do not sum over all other vertices $j' > j$, but only over vertices j' that have nodes in common with j.

For second derivatives we need to differentiate (17), involving a trace expression with derivative $d\mathrm{trace}(AB) = \mathrm{trace}(dAB) + \mathrm{trace}(AdB)$, and the former quadratic form. The Hessian becomes

$$\begin{aligned}
\frac{d^2 l_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{d\theta_r d\theta_s} &= \sum_j \Big[-\frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d^2\boldsymbol{\Sigma}_j}{d\theta_r d\theta_s}) + \frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}) \\
&- \boldsymbol{Y}'_j\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}\boldsymbol{Q}_j\boldsymbol{Y}_j + \frac{1}{2}\boldsymbol{Y}'_j\boldsymbol{Q}_j\frac{d^2\boldsymbol{\Sigma}_j}{d\theta_s d\theta_r}\boldsymbol{Q}_j\boldsymbol{Y}_j\Big].
\end{aligned}$$

The expected Hessian becomes

$$\begin{aligned}
E(\frac{d^2 l_{CL}(\boldsymbol{Y};\boldsymbol{\theta})}{d\theta_r d\theta_s}) &= \sum_j \Big[-\frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d^2\boldsymbol{\Sigma}_j}{d\theta_r d\theta_s}) + \frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}) \\
&- \mathrm{trace}(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}) + \frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d^2\boldsymbol{\Sigma}_j}{d\theta_s d\theta_r})\Big] \\
&= -\sum_j \frac{1}{2}\mathrm{trace}(\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_s}\boldsymbol{Q}_j\frac{d\boldsymbol{\Sigma}_j}{d\theta_r}).
\end{aligned}$$

B GPU implementation

MATLAB/Jacket defines a set of GPU-friendly data types and a host of new GPU-friendly functions. Modifying existing MATLAB code for compatibility with Jacket can be as simple as updating data types and functions to GPU-friendly versions. At a cost of $350 for a perpetual, academic, Jacket Base license, this provides easy access to the vast power of GPU-computing. The Jacket command `gfor` is a for loop which simultaneously launches all iterations on the GPU, provided the iterations are independent. This has been a large success for Monte Carlo computations. The `gfor` routine allows parallelisation of the CL expression, if we can allocate data effectively on the GPU. Few function names need modification since most standard functions such as `sqrt` and `lu` already run on the GPU if given GPU-friendly arguments. The ease of implementation using MATLAB/Jacket obscures how MATLAB software interfaces with the GPU hardware. It is sometimes hard to go beyond the standard outlook and streamline the code for your purposes.

CUDA C is a NVIDIA extension to the C language that allows code to run in parallel on a CUDA-enabled GPU. At a high level, CUDA C defines a way to execute functions on the GPU and copy variables to the GPU, but otherwise looks similar to standard C code. We used the CUBLAS library, an extension of the BLAS library for GPUs, to implement efficient calculation of determinants and quadratic forms in the Fisher scoring algorithm. This kind of parallelisation speeds up all terms in the CL expressions, but (at current) no parallelisation is applied to the CL sum. Specifically a parallelised Cholesky decomposition, based largely on (Bouckaert, Bouckaert), was developed. This decomposition has two main kernels: one to perform Cholesky decomposition down the block diagonal and the second to perform the Cholesky decomposition below each block diagonal. The block diagonal was chosen to fit within a single thread-block, i.e. 32×32 on CUDA 2.0 devices and 16×16 on previous devices. In addition, kernels were required for each parameter derivative of the Fisher scoring algorithm.

References

Abramowitz, M. and I. Stegun (1964). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications.

Banerjee, S., B. P. Carlin, and A. E. Gelfand (2004). *Hierarchical Modeling and Analysis for Spatial Data*. Monographs on Statistics and Applied Probability. Boca Raton: Chapman & Hall/CRC.

Banerjee, S., A. E. Gelfand, A. O. Finley, and H. Sang (2008). Gaussian predictive process models for large spatial data sets. *J. R. Stat. Soc. Ser. B Stat. Methodol. 70*(4), 825–848.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *J. Roy. Statist. Soc. Ser. B 36*, 192–236. With discussion.

Bouckaert, R. GPU Cholesky decomposition source code.

Caragea, P. C. and R. L. Smith (2007). Asymptotic properties of computationally efficient alternative estimators for a class of multivariate normal models. *J. Multivariate Anal. 98*(7), 1417–1440.

Cressie, N. and G. Johannesson (2008). Fixed rank kriging for very large spatial data sets. *J. Roy. Statist. Soc. Ser. B 70*(1), 209–226.

Curriero, F. C. and S. Lele (1999). A composite likelihood approach to semivariogram estimation. *J. Agric. Biol. Environ. Stat. 4*(1), 9–28.

Diggle, P. and S. Lophaven (2006). Bayesian geostatistical design. *Scand. J. Statist. 33*(1), 53–64.

Ellefmo, S. and J. Eidsvik (2009). Local and spatial joint frequency uncertainty and its application to rock mass characterisation. *Rock mechanics and rock engineering 42*(4), 667–688.

Finley, A., H. Sang, S. Banerjee, and A. Gelfand (2009). Improving the performance of predictive process modeling for large datasets. *Computational statistics & data analysis 53*(8), 2873–2884.

Fuentes, M. (2007). Approximate likelihood for large irregularly spaced spatial data. *J. Amer. Statist. Assoc. 102*(477), 321–331.

Furrer, R., M. G. Genton, and D. Nychka (2006). Covariance tapering for interpolation of large spatial datasets. *J. Comput. Graph. Statist. 15*(3), 502–523.

Galoppo, N., N. Govindaraju, M. Henson, and D. Manocha (2005). LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, pp. 3. IEEE Computer Society.

Godambe, V. (1960). An optimum property of regular maximum likelihood estimation. *The Annals of Mathematical Statistics 31*(4), 1208–1211.

Godambe, V. P. and C. C. Heyde (1987). Quasi-likelihood and optimal estimation. *Internat. Statist. Rev. 55*(3), 231–244.

Higham, N. J. (2008). *Functions of matrices*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). Theory and computation.

Kaufman, C., M. Schervish, and D. Nychka (2008). Covariance tapering for likelihood-based estimation in large spatial datasets. *J. Amer. Statist. Assoc. 103*(484), 1545–1569.

Krüger, J. and R. Westermann (2005). Linear algebra operators for GPU implementation of numerical algorithms. In *ACM SIGGRAPH 2005 Courses*, pp. 234. ACM.

Lee, A., C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *J. Comput. Graph. Statist. 19*(4), 769–789.

Li, H. and L. Petzold (2010). Efficient parallelization of the stochastic simulation algorithm for chemically reacting systems on the graphics processing unit. *International Journal of High Performance Computing Applications 24*(2), 107.

Lindgren, F., J. Lindstrø m, and H. Rue (2011). An explicit link between gaussian fields and gaussian markov random fields: The spde approach. *J. Roy. Statist. Soc. Ser. B 73*(3), To appear.

Lindsay, B. G. (1988). Composite likelihood methods. In *Statistical inference from stochastic processes (Ithaca, NY, 1987)*, Volume 80 of *Contemp. Math.*, pp. 221–239. Providence, RI: Amer. Math. Soc.

Mardia, K. V. and R. J. Marshall (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika 71*(1), 135–146.

Niemi, J. and M. Wheeler (2011). Efficient bayesian inference in stochastic chemical kinetic models using graphical processing units. Submitted, `http://arxiv.org/abs/1101.4242`.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press.

Sahr, K. (2011). Dggrid version 3.1b. user documentation for discrete global grid generation software. *Southern Oregon*. Available from www.sou.edu/cs/sahr/dgg.

Shaby, B. and D. Ruppert (2011). Tapered covariance: Bayesian estimation and asymptotics. *J. Comp. Graph. Statist.*. To appear.

Stein, M. L. (2008). A modeling approach for large spatial datasets. *J. Korean Statist. Soc. 37*(1), 3–10.

Stein, M. L., Z. Chi, and L. J. Welty (2004). Approximating likelihoods for large spatial data sets. *J. R. Stat. Soc. Ser. B Stat. Methodol. 66*(2), 275–296.

Suchard, M. and A. Rambaut (2009). Many-core algorithms for statistical phylogenetics. *Bioinformatics 25*(11), 1370.

Suchard, M., Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West (2010). Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *J. Comput. Graph. Statist. 19*(2), 419–438.

Varin, C. (2008). On composite marginal likelihoods. *AStA Adv. Stat. Anal. 92*(1), 1–28.

Vecchia, A. V. (1988). Estimation and model identification for continuous spatial processes. *J. Roy. Statist. Soc. Ser. B 50*(2), 297–312.

Volkov, V. and J. Demmel (2008). Benchmarking GPUs to tune dense linear algebra. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pp. 1–11. IEEE Press.