Efficient Gaussian Process Inference for Short-Scale Spatio-Temporal Modeling

Jaakko Luttinen
Aalto University, Finland
jaakko.luttinen@aalto.fi

Alexander Ilin
Aalto Universiy, Finland
alexander.ilin@aalto.fi

Abstract

This paper presents an efficient Gaussian process inference scheme for modeling short-scale phenomena in spatio-temporal datasets. Our model uses a sum of separable, compactly supported covariance functions, which yields a full covariance matrix represented in terms of small sparse matrices operating either on the spatial or temporal domain. The proposed inference procedure is based on Gibbs sampling, in which samples from the conditional distribution of the latent function values are obtained by applying a simple linear transformation to samples drawn from the joint distribution of the function values and the observations. We make use of the proposed model structure and the conjugate gradient method to compute the required transformation. In the experimental part, the proposed algorithm is compared to the standard approach using the sparse Cholesky decomposition and it is shown to be much faster and computationally feasible for 100–1000 times larger datasets. We demonstrate the advantages of the proposed method in the problem of reconstructing sea surface temperature, which requires processing of a real-world dataset with 10^6 observations.

1 Introduction

Gaussian processes (GP) provide an elegant method for modeling non-linear functions in the Bayesian framework (Rasmussen and Williams, 2006). They are widely used for modeling spatio-temporal phenomena, where a typical problem is to model an unknown field as a function of location and time using a set of noisy measurements. Many other classical tools, such as empirical orthogonal functions, kriging, Kalman filtering and smoothing, are based on modeling either the temporal or spatial structure but not both.

The main factor limiting the use of GPs for large data sets is the high computational cost. The standard approach involves computing the Cholesky decomposition of an $N \times N$ covariance matrix, where N is the number of data points. This requires, in general, $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory, seriously limiting the tolerable size of the data sets. One approach to reduce the computational cost is based on computing a low-rank approximation of the covariance matrix (e.g., Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006; Titsias, 2009). This is especially suitable when the characteristic length-scale is much larger than the distance between neighboring data points. On the other hand, short-scale variability can be modeled using compactly supported covariance functions to construct a sparse covariance matrix, which makes it possible to use efficient sparse matrix algorithms (e.g., Vanhatalo and Vehtari, 2008; Furrer et al., 2006; Rasmussen and Williams, 2006). It is also possible to cluster the data and model local GPs on these clusters (Snelson and Ghahramani, 2007). However, none of these methods scale to model the short-scale variability of very large spatio-temporal datasets. Efficient methods based on the fast Fourier transform (FFT) require that the data is from a regular equispaced grid (Fritz et al., 2009), which is often an unrealistic restriction.

There has been much research on how to define reasonable covariance functions acting on space-time inputs (Cressie and Huang, 1999; De Iaco et al., 2001; De Cesare et al., 2001; Gneiting, 2002; Stein, 2005). The simplest approach is to use separable covariance functions, which multiply covariances computed separately in the spatial and temporal domain. Although separable covariance functions are known to have shortcom-

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

ings (Cressie and Huang, 1999; Fuentes, 2006; Stein, 2005), they are often used for constructing more complex non-separable covariance functions (e.g., De Iaco et al., 2001; De Cesare et al., 2001). The use of separable covariance functions have the advantage that the resulting covariance matrix is represented in terms of the Kronecker product of smaller covariance matrices acting only in the spatial or temporal domain. This allows using significantly less memory for storing the covariance matrix. In addition, efficient inference is straightforward when the observations are noiseless and there are no missing values (e.g., Rougier, 2008).

The focus of this paper is on deriving an efficient GP inference procedure for short-scale spatio-temporal models. We assume that the covariance function is a sum of spatially and temporally separable covariance functions. In order to concentrate on modeling short-scale variability, we use compactly supported covariance functions. Efficient GP inference in the considered case is not straightforward for several reasons. Firstly, in realistic settings when the covariance function is a sum of separable ones, observations are noisy or there are missing data, the covariance matrix does not have the Kronecker product structure. Secondly, direct methods based on the sparse Cholesky decomposition of the covariance matrix are not applicable because the Kronecker product of two sparse matrices may contain too many non-zero elements. Although similar covariance structures has been studied before, the existing methods are not suitable for modeling short-scale variability of large datasets because they use low-rank approximations of the covariance matrix (Bonilla et al., 2008) or need to explicitly form and invert it (Zhang, 2007). We show how to perform Bayesian inference using sampling methods. The sampling scheme is based on transforming samples from the joint prior distribution to posterior samples using the conjugate gradient method.

The rest of the paper is organized as follows. Section 2 briefly presents existing GP regression ideas for spatio-temporal modeling. Section 3 proposes a novel method by constructing a covariance function as a sum of separable covariance functions and showing how to perform Bayesian inference using Gibbs sampling. Section 4 compares the proposed algorithm with the traditional methods using both artificial data and a large sea surface temperature data set.

2 Background

2.1 Gaussian Process Regression

Gaussian processes are used for setting distributions over unknown functions. Let $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ be a set of given input-output pairs:

$$y_n = f(\mathbf{x}_n) + \text{noise}, \tag{1}$$

where f is the unknown function of interest and the noise is typically Gaussian. In the GP methodology, the prior distribution over f is chosen such that the noiseless function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^{\mathrm{T}}$ are assumed to be normally distributed:

$$p(\mathbf{f}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K_f})$$

with covariance matrix $\mathbf{K_f}$ whose ij-th element is computed using covariance function $k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$ for the corresponding inputs $\mathbf{x}_i$, $\mathbf{x}_j$ and hyperparameters $\boldsymbol{\theta}$. Assuming Gaussian noise in (1) yields likelihood

$$p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \boldsymbol{\Sigma}),$$

in which $\mathbf{y} = [y_1, \ldots, y_N]^{\mathrm{T}}$ and the noise covariance matrix is often chosen to be $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Then applying Bayes' rule results in the following posterior:

$$p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{f}|\mathbf{K_f}(\mathbf{K_f} + \boldsymbol{\Sigma})^{-1}\mathbf{y}, \boldsymbol{\Psi}\right), \tag{2}$$
$$\text{where } \boldsymbol{\Psi} = \mathbf{K_f} - \mathbf{K_f}(\mathbf{K_f} + \boldsymbol{\Sigma})^{-1}\mathbf{K_f}.$$

Typically, one evaluates the posterior mean and variance of the function values for the optimized hyperparameters.

Compactly supported covariance functions are used to model short-scale variability rather than long-scale dependencies (Rasmussen and Williams, 2006). An example of a compactly supported covariance function is

$$(1-r)_+^{j+2}\left((j^2+4j+3)r^2 + (3j+6)r + 3\right)/3, \tag{3}$$

where $r = \|\mathbf{x} - \mathbf{x}'\|/\theta$, $(x)_+ = \max(x, 0)$, $j = \lfloor \frac{\beta}{2} \rfloor + 3$ and β is the dimensionality of the input $\mathbf{x}$. The covariance is zero if the distance between the inputs is greater than the length-scale parameter θ, thus the function produces sparse covariance matrices by construction. One can therefore utilize sparse matrix algorithms gaining significant computational advantage.

2.2 Separable Covariance Functions for Space-Time Inputs

We consider spatio-temporal data sets which are defined on a grid such that the set of input points is a Cartesian product $S \times T = \{(s_m, t_n) : m = 1, \ldots, M, n = 1, \ldots, N\}$ of the set of spatial points $S = \{s_m\}$ and the set of time instances $T = \{t_n\}$. The individual sets S and T do not need to be regular, that is, the spatial points can be arbitrarily located and the time points do not need to be evenly spaced, contrary to the FFT-based methods (Fritz et al., 2009). The

latent function values $\mathbf{f}$ and the observations $\mathbf{y}$ can be presented as $M \times N$ matrices $\mathbf{F}$ and $\mathbf{Y}$, where $[\mathbf{F}]_{mn}$ and $[\mathbf{Y}]_{mn}$ correspond to input (s_m, t_n).

For computational efficiency, the covariance function is often assumed to be separable as

$$k\left((s,t),(s',t');\boldsymbol{\theta}\right) = k_{\mathrm{s}}(s,s';\boldsymbol{\theta}_{\mathrm{s}})k_{\mathrm{t}}(t,t';\boldsymbol{\theta}_{\mathrm{t}}),$$

where k_{s} and k_{t} operate on the spatial and temporal domains, respectively, and $\boldsymbol{\theta}_{\mathrm{s}}$ and $\boldsymbol{\theta}_{\mathrm{t}}$ are the hyperparameters of the covariance functions. As the input set is a Cartesian product, the resulting $MN \times MN$ prior covariance matrix is a Kronecker product

$$\mathbf{K}_{\mathbf{f}} = \mathbf{K}_T \otimes \mathbf{K}_S, \tag{4}$$

where $\mathbf{K}_S$ and $\mathbf{K}_T$ are $M \times M$ and $N \times N$ covariance matrices computed for input sets S and T using k_{s} and k_{t}, respectively. Thus, the memory requirement is reduced from $\mathcal{O}(M^2N^2)$ for full matrices to $\mathcal{O}(M^2 + N^2)$.

If one uses compactly supported covariance functions over the spatial and temporal domains, it is possible that the covariance matrices use even less memory than the data. This can be seen from the following: Let D_S and D_T be the average number of spatial and temporal neighbors (elements that have non-zero covariance) per location and time instance. The memory required is $\mathcal{O}(MN)$ for the data and $\mathcal{O}(D_S D_T MN)$ for the sparse spatio-temporal covariance matrix. However, utilizing the Kronecker product structure reduces the memory requirement for the covariance matrix to $\mathcal{O}(D_S M + D_T N)$. If $D_S \ll N$ and $D_T \ll M$, the memory consumption is dominated by the data matrix instead of the covariance matrix. For instance, a 1000×1000 data matrix takes 8 MB of memory in double precision. If the covariance matrices $\mathbf{K}_S$ and $\mathbf{K}_T$ have only 10 non-zero elements per row on average, they both would only use 0.48 MB of memory in the sparse representation. For comparison, the full covariance matrix would use 2.4 GB, which might be infeasible for matrix decomposition algorithms both in computation time and memory usage.

The Kronecker product structure makes the matrix computations more efficient by utilizing the following properties:

$$(\mathbf{A} \otimes \mathbf{B})[\mathbf{X}]_{:} = [\mathbf{B}\mathbf{X}\mathbf{A}^{\mathrm{T}}]_{:}, \tag{5}$$

$$\mathbf{A}\mathbf{B} \otimes \mathbf{C}\mathbf{D} = (\mathbf{A} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{D}), \tag{6}$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}, \tag{7}$$

where $[\mathbf{X}]_{:}$ is a vector obtained by stacking the column vectors of $\mathbf{X}$. For example, (7) explains why the inverse of $\mathbf{K}_{\mathbf{f}}$ in (4) can be computed efficiently. From (6) it follows that the Cholesky factor of $\mathbf{K}_{\mathbf{f}} = \mathbf{K}_T \otimes \mathbf{K}_S$ is also a Kronecker product. These properties can be exploited efficiently if the data has no noise ($\boldsymbol{\Sigma} = \mathbf{0}$) nor missing values (Rougier, 2008).

3 Short-Scale Spatio-Temporal Gaussian Process

We now consider a more realistic scenario when the observations are noisy, that is, $\boldsymbol{\Sigma} \neq \mathbf{0}$, and when there are missing observations for some of the pairs (s_m, t_n). We also consider a more complex covariance function which is a sum of separable covariance functions:

$$k((s,t),(s',t');\boldsymbol{\theta}) = \sum_{d=1}^{D} k_{\mathrm{s}}^d(s,s';\boldsymbol{\theta}_{\mathrm{s}}^d)k_{\mathrm{t}}^d(t,t';\boldsymbol{\theta}_{\mathrm{t}}^d), \tag{8}$$

where the magnitudes are controlled by a scale parameter in k_{s}^d or k_{t}^d. This form of the covariance function is used to overcome some of the known shortcomings of the separable covariance functions. The resulting covariance matrix is then

$$\mathbf{K}_{\mathbf{f}} = \sum_{d=1}^{D} \mathbf{K}_{\mathbf{f}}^d, \quad \text{and} \quad \mathbf{K}_{\mathbf{f}}^d = \mathbf{K}_T^d \otimes \mathbf{K}_S^d,$$

where $\mathbf{K}_S^d$ and $\mathbf{K}_T^d$ are computed using k_{s}^d and k_{t}^d, respectively. The function values corresponding to the prior covariance matrix $\mathbf{K}_{\mathbf{f}}^d$ are denoted as $\mathbf{F}_d$. If there are missing values in the data matrix $\mathbf{Y}$, the covariance matrix can be constructed as

$$\mathbf{K}_{\mathbf{y}} = \mathbf{P}(\mathbf{K}_{\mathbf{f}} + \boldsymbol{\Sigma})\mathbf{P}^{\mathrm{T}}, \tag{9}$$

where $\mathbf{P}$ is a matrix obtained by removing the rows corresponding to the missing observations from the $MN \times MN$ identity matrix. The use of matrix $\mathbf{P}$ results in removing the elements that correspond to the missing observations. Note that the computation of the covariance matrix $\mathbf{K}_{\mathbf{y}}$ would require a large amount of memory, thus it is not computed explicitly.

Similar models have been explored in the context of multi-task learning (Bonilla et al., 2008) and a multivariate linear coregionalization model (Zhang, 2007). The method by Zhang (2007) requires the direct computation and inversion of $\mathbf{K}_{\mathbf{y}}$, which is prohibitive even for small M and N. Bonilla et al. (2008) use a separable covariance matrix (i.e., $D = 1$) and it is approximated by using a lower dimensional representation, making short-scale modeling inaccurate. Thus, these methods are not well suited for modeling short-scale phenomena in large data sets.

3.1 Evaluation of the Posterior of Latent Function Values F

In the present setting, efficient evaluation of the posterior $p(\mathbf{f}|\mathbf{y})$ in (2) is not straightforward because ma-

trix $\mathbf{K_y}$ which must be inverted is not, in general, a Kronecker product of two matrices. The Kronecker product property is lost because of the presence of the noise, the missing observations (the rows and columns corresponding to the missing data are removed from $\mathbf{K_y}$ in (9)) and the use of the sum of separable covariance functions in (8). Thus, the Cholesky factor is not a Kronecker product, causing high computational costs and large memory requirements for computing the Cholesky decomposition.

Instead of computing the Cholesky decomposition, one can use iterative methods that solve systems of linear equations, such as the conjugate gradient method, to evaluate terms of the form $(\mathbf{K_f} + \mathbf{\Sigma})^{-1}[\mathbf{X}]_:$. In this approach, one has to compute matrix-vector products which can efficiently be done using (5):

$$(\mathbf{K_f} + \mathbf{\Sigma})[\mathbf{A}]_: = \sum_{d=1}^{D} \left[\mathbf{K}_S^d \mathbf{A} \mathbf{K}_T^d\right]_: + \mathbf{v} \circ [\mathbf{A}]_:,$$

where $\mathbf{A}$ is an arbitrary matrix, $\circ$ denotes the entry-wise product and $\mathbf{\Sigma} = \mathrm{diag}(\mathbf{v})$, that is, $\mathbf{\Sigma}$ is a diagonal matrix and vector $\mathbf{v}$ contains its diagonal elements. This approach can be used to compute at least the posterior means. However, the computation of the posterior variances in a similar way is too expensive: One would need to apply a similar procedure for computing the posterior variance of *each* latent function value. Therefore, this approach is impractical for large data sets.

In this paper, we propose to examine the posterior distribution of the latent function values by drawing samples from it. This can be done efficiently by using the following result. Let $\mathbf{f}$ and $\mathbf{y}$ be multivariate Gaussian variables:

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_\mathbf{f}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{K_f} & \mathbf{K_{fy}}\\ \mathbf{K_{yf}} & \mathbf{K_y}\end{bmatrix}\right).$$

Now, if we can draw samples $(\tilde{\mathbf{f}}, \tilde{\mathbf{y}})$ from the joint distribution $p(\mathbf{f}, \mathbf{y})$ and transform them to

$$\mathbf{f}_* = \tilde{\mathbf{f}} - \mathbf{K_{fy}}\mathbf{K_y}^{-1}(\tilde{\mathbf{y}} - \mathbf{y}) \tag{10}$$

using fixed $\mathbf{y}$, then the transformed samples $\mathbf{f}_*$ would come from the conditional distribution $p(\mathbf{f}|\mathbf{y})$ defined in (2). This can easily be checked by computing the first two moments for (10).

The presented approach is feasible for our GP model because it is possible to draw samples from $p(\mathbf{f}, \mathbf{y})$ and to compute transformation (10) efficiently. Let $\mathbf{L}_T^d$ and $\mathbf{L}_S^d$ be the sparse Cholesky factors of matrices $\mathbf{K}_T^d$ and $\mathbf{K}_S^d$, respectively, that is, $\mathbf{K}_T^d = \mathbf{L}_T^d {\mathbf{L}_T^d}^\mathrm{T}$ and $\mathbf{K}_S^d = \mathbf{L}_S^d {\mathbf{L}_S^d}^\mathrm{T}$. Then, the Cholesky factor of $\mathbf{K}_\mathbf{f}^d$ is $\mathbf{L}_\mathbf{f}^d = \mathbf{L}_T^d \otimes \mathbf{L}_S^d$. Let now $\mathbf{Z}_0, \ldots, \mathbf{Z}_D$ be matrices of size $M \times N$ with the elements drawn independently from $\mathcal{N}(0,1)$. The samples from the joint distribution $p(\{\mathbf{F}_d\}, \mathbf{Y}|\boldsymbol{\theta})$ can be obtained by

$$\tilde{\mathbf{F}}_d = \mathbf{L}_S^d \mathbf{Z}_d (\mathbf{L}_T^d)^\mathrm{T}, \quad [\tilde{\mathbf{Y}}]_: = \sum_{d=1}^{D} [\tilde{\mathbf{F}}_d]_: + \mathrm{diag}(\mathbf{v})^{\frac{1}{2}} [\mathbf{Z}_0]_:.$$

Now, the transformation (10) can be used to produce samples from $p(\{\mathbf{F}_d\}|\mathbf{Y}, \boldsymbol{\theta})$:

$$[\mathbf{F}_d]_: = [\tilde{\mathbf{F}}_d]_: - \mathbf{K}_\mathbf{f}^d \mathbf{P}^\mathrm{T} \mathbf{K}_\mathbf{y}^{-1} \mathbf{P} \left([\tilde{\mathbf{Y}}]_: - [\mathbf{Y}]_:\right). \tag{11}$$

Transformation (11) can be computed efficiently by using iterative algorithms (e.g., the conjugate gradient method) to avoid the direct inversion of $\mathbf{K_y}$ and by utilizing the Kronecker product structure of $\mathbf{K}_\mathbf{f}^d$ to compute the required products. Note that the conjugate gradient method needs to be applied only once in order to draw a sample of each $\mathbf{F}_1, \ldots, \mathbf{F}_D$ and that these samples are from the true posterior up to the numerical accuracy of the conjugate gradient method (we used error tolerance 10^{-6} as the stopping criterion in the algorithm by Shewchuk (1994)). The conjugate gradient method was also used to draw samples from the Gaussian distribution by Wikle et al. (2001) when the inverse of the covariance matrix is easy to compute. In practice, drawing 10–1000 samples is typically enough to compute the required statistics with reasonable accuracy. For instance, the variance of the sample mean estimator is inversely proportional to the number of samples.

Predictions can be made efficiently given posterior samples of $\{\mathbf{F}_d\}$ and $\boldsymbol{\theta}$. As before, we assume that the input set for the predictions is a Cartesian product $\tilde{S} \times \tilde{T}$ of two sets $\tilde{S} = \{\tilde{s}_{\tilde{m}}\}$ and $\tilde{T} = \{\tilde{t}_{\tilde{n}}\}$. The posterior predictive distribution is

$$p\left(\{\tilde{\mathbf{F}}_d\} \middle| \{\mathbf{F}_d\}, \mathbf{Y}, \boldsymbol{\theta}\right) = \prod_{d=1}^{D} \mathcal{N}\left([\tilde{\mathbf{F}}_d]_: \middle| \boldsymbol{\mu}_d, \mathbf{\Lambda}_d\right),$$

where

$$\begin{aligned}\boldsymbol{\mu}_d &= \left(\mathbf{K}_{\tilde{T}T}^d \otimes \mathbf{K}_{\tilde{S}S}^d\right)\left(\mathbf{K}_T^d \otimes \mathbf{K}_S^d\right)^{-1}[\mathbf{F}_d]_:\\ \mathbf{\Lambda}_d &= \mathbf{K}_{\tilde{T}}^d \otimes \mathbf{K}_{\tilde{S}}^d\\ &\quad - \left(\mathbf{K}_{\tilde{T}T}^d \otimes \mathbf{K}_{\tilde{S}S}^d\right)\left(\mathbf{K}_T^d \otimes \mathbf{K}_S^d\right)^{-1}\left(\mathbf{K}_{T\tilde{T}}^d \otimes \mathbf{K}_{S\tilde{S}}^d\right).\end{aligned}$$

$\mathbf{K}_{\tilde{T}T}^d$ is defined as $\left[\mathbf{K}_{\tilde{T}T}^d\right]_{ij} = k_\mathrm{t}^d(\tilde{t}_i, t_j; \boldsymbol{\theta}_\mathrm{t}^d)$ and the covariance matrices $\boldsymbol{K}_{T\tilde{T}}^d$, $\boldsymbol{K}_{\tilde{T}}^d$, $\boldsymbol{K}_{\tilde{S}S}^d$, $\boldsymbol{K}_{S\tilde{S}}^d$ and $\boldsymbol{K}_{\tilde{S}}^d$ are computed similarly by using the covariance functions k_s^d and k_t^d and the input sets S, $\tilde{S}$, T and $\tilde{T}$ appropriately. Samples from this posterior predictive distribution can be drawn extremely efficiently by using a similar transformation as in (10) (no need for the conjugate gradient method) and the properties of the Kronecker product (5)–(7).

3.2 Evaluation of the Posterior of Hyperparameters $\boldsymbol{\theta}$

The traditional ways to do inference about the hyperparameters are computationally expensive and therefore impractical for large data sets. The general expression for the log-posterior $\log p(\boldsymbol{\theta}|\mathbf{Y})$ of the hyperparameters is up to a constant

$$-\frac{1}{2}[\mathbf{Y}]_{:}^{\mathrm{T}}\mathbf{K}_{\mathbf{y}}^{-1}[\mathbf{Y}]_{:} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{y}}| + \log p(\boldsymbol{\theta}), \qquad (12)$$

which has to be computed many times either for drawing samples from the posterior or for finding the maximum of (12) with respect to the hyperparameters. However, the computation of $\log|\mathbf{K}_{\mathbf{y}}|$ is very expensive in general and it is possible to simplify the computations in the case of separable covariance functions with an isotropic noise model, that is, when $\mathbf{K}_{\mathbf{y}} = \mathbf{K}_T \otimes \mathbf{K}_S + \sigma^2\mathbf{I}$.[1]

In this paper, we propose to use Gibbs sampling to draw samples from the joint posterior $p(\boldsymbol{\theta}, \{\mathbf{F}_d\}|\mathbf{Y})$. Drawing samples from the conditional distribution $p(\{\mathbf{F}_d\}|\boldsymbol{\theta}, \mathbf{Y})$ was discussed in Section 3.1 and therefore we need to design an efficient sampler from the conditional distribution $p(\boldsymbol{\theta}|\{\mathbf{F}_d\}, \mathbf{Y})$ and to set the prior distribution $p(\boldsymbol{\theta})$.

In order to build an efficient Gibbs sampler, we use the reparameterization method proposed by Murray and Adams (2010). The problem here is that conditioning on $\{\mathbf{F}_d\}$ makes the distribution of $\boldsymbol{\theta}$ extremely narrow and the updates of $\boldsymbol{\theta}$ small, causing the Gibbs sampler to proceed slowly. The sampling efficiency can be improved by reparameterizing the model such that the latent variables are less dependent on the hyperparameters. This can be achieved by using auxiliary variables $\mathbf{G}_d$ which satisfy $[\mathbf{F}_d]_{:} = (\mathbf{L}_T^d \otimes \mathbf{L}_S^d)[\mathbf{G}_d]_{:}$, or equivalently

$$\mathbf{F}_d = \mathbf{F}_d(\boldsymbol{\theta}, \mathbf{G}_d) = \mathbf{L}_S^d \mathbf{G}_d \left(\mathbf{L}_T^d\right)^{\mathrm{T}}. \qquad (13)$$

Note that $\mathbf{L}_S^d$, $\mathbf{L}_T^d$ are the Cholesky factors of matrices $\mathbf{K}_S^d$, $\mathbf{K}_T^d$ and therefore they are functions of $\boldsymbol{\theta}$. This yields the transformed probabilistic model

$$p\left(\mathbf{Y}, \{\mathbf{G}_d\}, \boldsymbol{\theta}\right) = p\left(\mathbf{Y} \,|\{\mathbf{F}_d\}, \boldsymbol{\theta}\right) p\left(\{\mathbf{G}_d\} \,|\, \boldsymbol{\theta}\right) p\left(\boldsymbol{\theta}\right)$$

in which $p\left(\{\mathbf{G}_d\} \,|\, \boldsymbol{\theta}\right)$ is constant with respect to $\boldsymbol{\theta}$ by the choice of (13). For the Gibbs sampler, samples from $p\left(\{\mathbf{G}_d\}|\,\boldsymbol{\theta}, \mathbf{Y}\right)$ can be drawn by first sampling from $p(\mathbf{F}_1, \ldots, \mathbf{F}_D|\boldsymbol{\theta}, \mathbf{Y})$ and then using (13) to solve for $\mathbf{G}_d$, which can be done efficiently because of the sparse, lower-triangular structure of $\mathbf{L}_S^d$ and $\mathbf{L}_T^d$.

Sampling from the conditional distribution

$$p(\boldsymbol{\theta}|\{\mathbf{G}_d\}, \mathbf{Y}) \propto p\left(\mathbf{Y} \,|\{\mathbf{F}_d(\boldsymbol{\theta}, \mathbf{G}_d)\}\right) p(\boldsymbol{\theta}).$$

is efficient because $p\left(\mathbf{Y} \,|\{\mathbf{F}_d\}, \boldsymbol{\theta}\right)$ has a diagonal covariance matrix $\mathrm{diag}(\mathbf{v})$. One can use, for instance, slice sampling (Neal, 2003; Murray and Adams, 2010), which requires little tuning, or Hamiltonian Monte Carlo (Neal, 2011). In our experiments, the hyperparameters usually had very small correlations, thus axis-aligned slice sampling was very efficient. The main challenge for the sampler is not the posterior correlations between the hyperparameters but that the hyperparameters are sampled conditioned on the (transformed) latent function values. Thus, the alternating sampling of $\{\mathbf{F}_d\}$ and $\boldsymbol{\theta}$ may cause high autocorrelation for $\boldsymbol{\theta}$ using any sampling method. Fortunately, for large data sets, the distribution is likely to be sharply peaked, thus obtaining only a few effectively independent samples is often sufficient in practice.

4 Experiments

4.1 2-D Artificial Experiment

We used artificially generated data to compare the computational efficiency of the proposed algorithm based on sampling with the traditional approach which computes the mean and the variances for the latent function values exactly and performs sampling of the hyperparameters. In the traditional approach, the posterior means and variances are computed using (2) and the samples of hyperparameters $\boldsymbol{\theta}$ are drawn using the marginal log-posterior (12). In this case, covariance matrix $\mathbf{K}_{\mathbf{y}}$ and its Cholesky factor are computed explicitly by utilizing sparse matrix algorithms.

We used artificial data in which the latent function values were generated using a GP model with a separable covariance function as in (8) with $D = 1$ and compactly supported $k_{\mathrm{s}}(s, s')$, $k_{\mathrm{t}}(t, t')$ of form (3) with the cut-off length 10. The data were generated on a 2-dimensional regular square lattice with input sets $S = \{1, 2, \ldots, n\}$ and $T = \{1, 2, \ldots, n\}$ resulting in $N = n^2$ data points as there were no missing observations. The variance of the latent function values was set to one. Gaussian noise with standard deviation 0.3 was then added to the observations. In order to estimate the performance as a function of the data set size, the total number N of the data points was varied in the range $[10^3\ 10^8]$. Note that the available amount of computer memory limited the tolerable data set size to 10^5 for the traditional method and to 10^8 for the proposed one.

[1] In that case, one can compute the singular value decomposition (SVD) of two matrices $\mathbf{K}_S = \mathbf{U}_S\boldsymbol{\Lambda}_S\mathbf{U}_S^{\mathrm{T}}$ and $\mathbf{K}_T = \mathbf{U}_T\boldsymbol{\Lambda}_T\mathbf{U}_T^{\mathrm{T}}$, which simplifies computing SVD $\mathbf{K}_{\mathbf{y}} = (\mathbf{U}_S \otimes \mathbf{U}_T)(\boldsymbol{\Lambda}_S \otimes \boldsymbol{\Lambda}_T + \sigma^2\mathbf{I})(\mathbf{U}_S \otimes \mathbf{U}_T)^{\mathrm{T}}$. This can be used for evaluating the log-determinant and the inverse.

The data sets were processed by the GP model which assumes the same covariance function model that was used for generating the data. Thus, the model had four hyperparameters: the two length-scale parameters of the compactly supported covariance functions $k_{\mathrm{s}}(s, s')$, $k_{\mathrm{t}}(t, t')$, the variance of the latent function values and the variance of the isotropic noise $\mathbf{\Sigma} = \sigma^2\mathbf{I}$.

In the first experiment, we compared the computing times required for the evaluation of the posterior of the latent function values $\mathbf{F}$ when the hyperparameters are fixed to the values which were used for generating the data. Figure 1a shows the CPU time needed for the exact inference with the traditional approach and the average time for drawing one sample and 100 samples for the proposed algorithm. Note that both axes are in log-scale, thus the slope of each line is roughly the exponent α of the computational complexity $\mathcal{O}(N^\alpha)$. The computational cost of the traditional approach is approximately $\mathcal{O}(N^2)$ while for the proposed approach it is $\mathcal{O}(N)$. The results show that for the considered data sets, the proposed method can do inference for approximately 100–1000 times larger data sets than the traditional method in the same CPU time, depending on how many samples is considered sufficient. In this experiment, already 10 samples gave a good approximation of the posterior distribution (see Figure 1b) and with 100 samples the approximation was almost indistinguishable from the true posterior.

In the second experiment, we compared the efficiency of drawing posterior samples of hyperparameters $\boldsymbol{\theta}$. Both the traditional and the proposed algorithms used slice sampling. For simplicity, the hyperparameters were initialized at the true values and the prior for each hyperparameter was broad. Slice sampling does not produce independent samples, thus we computed the effective sample size (ESS) based on the autocorrelation of the hyperparameters (Neal, 1993). The smallest ESS among the four hyperparameters was taken as the ESS for each method. Figure 1c shows the CPU time needed for producing one actual sample and one effectively independent sample using the two approaches. For the traditional algorithm, ESS was close to the actual sample size, but for the proposed method the autocorrelation was significant for some hyperparameters. Again, the proposed method works equally fast for approximately 100 times larger data sets. The main computational cost of the proposed algorithm comes from the conjugate gradient iterations required for sampling latent function values $\mathbf{F}$.

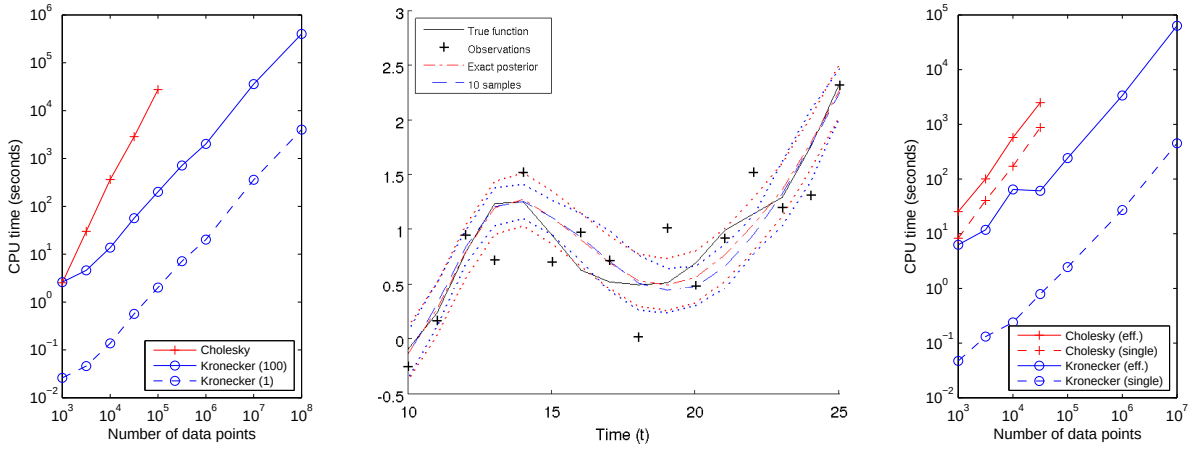
4.2 Reconstruction of Historical Sea Surface Temperatures

In this section, we consider a real-world problem which is the reconstruction of the global sea surface temperatures (SST) using historical observations. We used the U.K. Met Office historical SST data set MOHSST5, which contains monthly anomalies for $5^\circ \times 5^\circ$ latitude-longitude bins covering the period from 1856 to 1991 (Bottomley et al., 1990; Kaplan et al., 1998). Thus, there are 1727 spatial locations, 1632 time instances, and about 55% of the measurements are missing (primarily in the early periods). The goal is to reconstruct the missing information using the available data. One of the main challenges here is the big size of the data set, as there are about $1.3 \cdot 10^6$ observations in total. One therefore needs efficient algorithms that scale well to high-dimensional data.

We used the proposed algorithm to test its skills in the reconstruction problem. We used two short-scale GP models with $D = 1$ and $D = 2$ in (8) and compactly supported spatial and temporal covariance functions of form (3). For simplicity, the latitude-longitude coordinates were transformed to points in the 3-D Euclidean space as the spherical distance is approximately equal to the Euclidean distance on the local scale. The noise covariance matrix was assumed to be diagonal $\mathbf{\Sigma} = \sigma^2 \operatorname{diag}(\mathbf{v})$, where the elements of $\mathbf{v}$ were fixed to the inverse of the size of the corresponding spatial bins.[2] The two models had four and seven hyperparameters which were the variance of the latent function values, the length-scale parameters of $k_{\mathrm{s}}^d(s, s')$ and $k_{\mathrm{t}}^d(t, t')$, and the noise variance σ^2. The hyperparameters were assigned broad priors. We drew 1000 samples of the hyperparameters and the latent function values using the proposed method and discarded the first half as burn-in.

For comparison, we used the variational Bayesian variant of principal component analysis (VBPCA) (Bishop, 1999) and Gaussian-process factor analysis (GPFA), which sets Gaussian-process priors over the latent spatial and temporal components (Luttinen and Ilin, 2009). These models used 80 latent components and a similar diagonal noise covariance matrix as the short-scale GP. The setup for GPFA was copied from (Luttinen and Ilin, 2009), although we used the rational quadratic covariance functions instead of the squared exponential. 10 temporal components were modeled with the rational quadratic covariance function, 5 with a quasi-periodic covariance function and 65 with a compactly supported covariance function. The spatial components were modeled with the rational quadratic covariance function. The sparse approximations using pseudo-inputs (Titsias, 2009) were applied to the components without the compact support.

[2] Using such a noise model is essentially equivalent to weighting the observations according to the square root of the corresponding area, which is a standard preprocessing step to take into account the uneven distribution of the spatial locations.



(a) Inference time of $\mathbf{F}$ (b) Comparison of accuracy (c) Sampling time of $\boldsymbol{\theta}$

Figure 1: A comparison of the standard method (sparse Cholesky) and the proposed method (Kronecker). (a) The required CPU time for computing the mean and variance from the analytic equation (solid red) and using 100 samples (solid blue) or 1 sample (dashed blue) by the proposed method. (b) A comparison of the exact posterior distribution (red) to mean and two standard deviations computed using samples (blue) from one spatial location. (c) The CPU time per effective posterior sample of the hyperparameters $\boldsymbol{\theta}$ using standard Cholesky method (solid red) or the proposed method (solid blue). The dashed lines show the CPU time for a single auto-correlated sample in the chain.

In order to test the reconstruction performance, we used parts of the available data as a test set. In the first experiment, the test set was created by taking data points uniformly with probability 0.2. In this setting, most of the test data are surrounded by training samples, which is beneficial for the local model. In the second experiment, we used the spatial pattern of the values that are missing in the period 1856–1875 to take the corresponding observations from the period 1972–1991 as the test set. We also reversed the temporal order of the patterns so that the same locations which were not covered in January 1856 were used as the test data for December 1991 and so on. In this case, the amount of the test data was again about 20% of the total number of the observations. This setting is more realistic because the coverage in the test data resembles the pattern of the missing data in the real reconstruction problem. It is also more difficult for the local GP model, as the test data are farther away from the training data.

The reconstruction performance was evaluated using the weighted root mean square error (WRMSE)

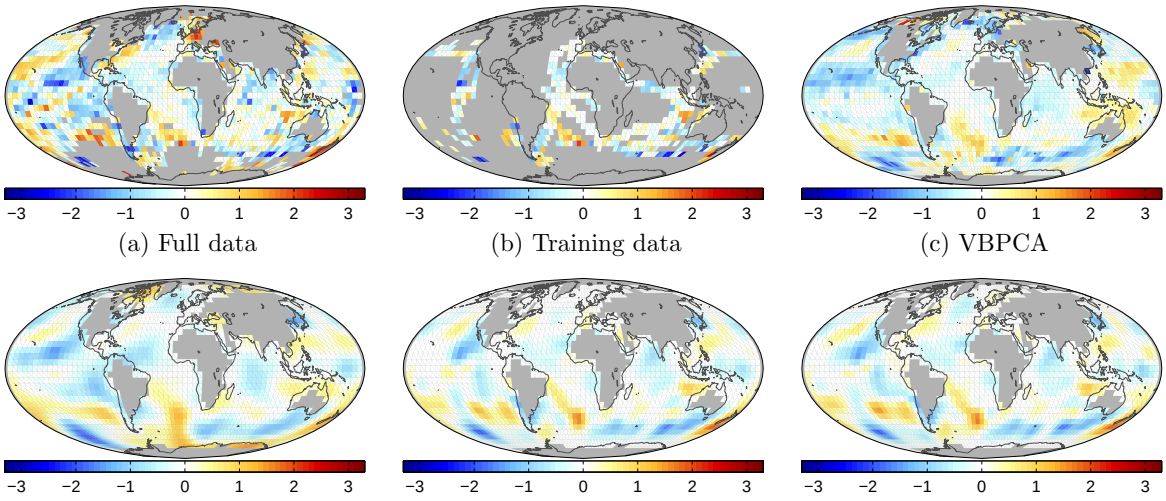
$$\sqrt{\left(\sum_i w_i \left([\mathbf{Y}]_i - [\mathbf{F}]_i\right)^2\right) / \left(\sum_i w_i\right)},$$

where the weights $\{w_i\}$ are the areas of the corresponding bin, $\mathbf{F}$ is the mean reconstruction and the sums are over those indices of the data $\mathbf{Y}$ that are in the test set. The proposed short-scale GP performed best for both test sets (see Table 1), VBPCA was the worst and GPFA was in between. For the short-scale GP, adding another separable covariance function (i.e., $D = 2$) improves the performance as the model is able to learn different length-scales, which is important especially for the pattern test set. Adding more components to VBPCA or GPFA did not significantly change the results as irrelevant components were automatically pruned out. The fact that the short-scale GP performed the best suggests that local models can alone perform relatively well in the reconstruction tasks. However, one of the most impressive results was the feasibility to process such a huge spatio-temporal data set with a short-scale GP model.

Table 1: Reconstruction WRMSE

Model	Uniform	Pattern	CPU time
VBPCA	0.5926	0.7820	$1 \cdot 10^4$s
GPFA	0.5767	0.7675	$2 \cdot 10^5$s
GP ($D = 1$)	0.5399	0.7445	$2 \cdot 10^5$s
GP ($D = 2$)	0.5322	0.7265	$4 \cdot 10^5$s

Figure 2 shows the data and the reconstructions for one month by the different methods. The reconstruc-



(a) Full data (b) Training data (c) VBPCA

(d) GPFA (e) Short-scale GP ($D = 1$) (f) Short-scale GP ($D = 2$)

Figure 2: Reconstructions of SST for one month using VBPCA, GPFA and GP for the training set from which the test set was removed using the pattern from early years. (a) All available measurements. (b) The measurements used for training. (c)-(f) The mean reconstructions of the methods. Grey color is used for missing values.

tions by the short-scale GPs and GPFA are smoother than by VBPCA because the latter does not take into account the spatial and temporal structure. The effect of the local modeling can be noticed in the GP reconstructions: In the larger regions without observations, the posterior mean gets close to zero, which is the prior mean. This effect may not be very prominent in the presented plots because the short-scale GP model is able to use even a few neighboring observations (in space or in time) to reconstruct missing data.

The reconstructions could probably be improved by combining the proposed GP model and GPFA. The latter is more suited to capture global phenomena (the dominating patterns in the system) whereas the proposed method models local phenomena. GPFA is based on a lower-dimensional representation of data, which means that it may require a large number of latent components in order to describe local phenomena well. In addition, using inducing inputs for the GPFA components strengthens the global structure assumption. On the contrary, the proposed model does not make low-rank approximations and it is not able to capture long-distance correlations. Thus, GPFA and the proposed method look at different features of the process and it could be beneficial to combine them.

Although the short-scale GP model performed well compared to the alternative methods, the main purpose of the experiment was to show that the proposed spatio-temporal GP method can be applied to large real-world data sets and it can be useful in practical reconstruction problems.

5 Conclusions

This paper presented an efficient Bayesian inference method for short-scale GPs on large spatio-temporal data sets. The model uses a sum of separable covariance functions defined for space-time inputs, which yields a covariance matrix described in the form of a sum of Kronecker products of two smaller matrices. The posterior inference scheme was based on Gibbs sampling which drew posterior samples of the latent function values using the conjugate gradient method and the hyperparameters using slice sampling.

The method was experimentally compared to the traditional inference algorithms on both artificial and real-world data. In comparison with the traditional Gaussian process method using the sparse Cholesky decomposition, the proposed inference method was shown to scale to orders of magnitude larger data sets with the same computational cost and memory usage. Using the proposed method, it was possible to apply short-scale GP modeling to a large sea surface temperature data set with 10^6 observations achieving good reconstruction performance.

The proposed GP method can be used as a building block for more complex model constructions. As the method models only short-scale phenomena, it could be combined with an efficient long-scale model for distant correlations. Such global-scale methods do not usually model the short-scale variability well and therefore they might be improved by combining them with the proposed model.

References

Bishop C.M. (1999). Variational principal components. In *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN'99)*, pp. 509–514.

Bonilla E.V., Chai K.M.A., Williams C.K.I. (2008). Multi-task Gaussian process regression. In Platt J.C., Koller D., Singer Y., Roweis S., editors, *Advances in Neural Information Processing Systems 20*. MIT Press.

Bottomley M., Folland C.K., Hsiung J., Newell R.E., Parker D.E. (1990). *Global Ocean Surface Temperature Atlas*. Her Majesty's Stn. Off.

Cressie N., Huang H.C. (1999). Classes of nonseparable, spatio-temporal stationary covariance functions. *Journal of the American Statistical Association*, 94(448):1330–1340.

De Cesare L., Myers D.E., Posa D. (2001). Estimating and modeling space-time correlation structures. *Statistics & Probability Letters*, 51:9–14.

De Iaco S., Myers D.E., Posa D. (2001). Space-time analysis using a general product-sum model. *Statistics & Probability Letters*, 52:21–28.

Fritz J., Neuweiler I., Nowak W. (2009). Application of FFT-based algorithms for large-scale universal kriging problems. *Mathematical Geosciences*, 41(5):509–533.

Fuentes M. (2006). Testing for separability of spatial-temporal covariance functions. *Journal of Statistical Planning and Inference*, 136:447–466.

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

Gneiting T. (2002). Nonseparable, stationary covariance functions for space-time data. *Journal of the American Statistical Association*, 97(458):590–600.

Hoffman Y., Ribak E. (1991). Constrained realizations of Gaussian fields: a simple algorithm. *The Astrophysical Journal*, 380:L5–L8.

Kaplan A., Cane M.A., Kushnir Y., Clement A.C., Blumenthal M.B., Rajagopalan B. (1998). Analyses of global sea surface temperature 1856–1991. *Journal of Geophysical Research*, 103(C9):18567–18589.

Luttinen J., Ilin A. (2009). Variational Gaussian-process factor analysis for modeling spatio-temporal data. In Bengio Y., Schuurmans D., Lafferty J., Williams C.K.I., Culotta A., editors, *Advances in Neural Information Processing Systems 22*. MIT Press.

Murray I., Adams R.P. (2010). Slice sampling covariance hyperparameters of latent Gaussian models. In Lafferty J., Williams C.K.I., Shawe-Taylor J., Zemel R., Culotta A., editors, *Advances in Neural Information Processing Systems 23*. MIT Press.

Neal R.M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Department of Computer Science, University of Toronto.

Neal R.M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–767.

Neal R.M. (2011). MCMC using Hamiltonian dynamics. In Brooks S., Gelman A., Jones G., Meng X.L.M., editors, *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC.

Quiñonero-Candela J., Rasmussen C.E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen C.E., Williams C.K.I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rougier J. (2008). Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843.

Shewchuk J.R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University.

Snelson E., Ghahramani Z. (2006). Sparse gaussian processes using pseudo-inputs. In Weiss Y., Schölkopf B., Platt J., editors, *Advances in Neural Information Processing Systems 18*, pp. 1257–1264. MIT Press.

Snelson E., Ghahramani Z. (2007). Local and global sparse Gaussian process approximations. In Meila M., Shen X., editors, *Proceedings of the 11th International Workshop on Artificial Intelligence and Statistics (AISTATS'07)*. Society for Artificial Intelligence and Statistics.

Stein M.L. (2005). Space-time covariance functions. *Journal of the American Statistical Association*, 100(469):310–321.

Titsias M.K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In van Dyk D., Welling M., editors, *Proceedings of the 12th International Workshop on Artificial Intelligence and Statistics (AISTATS'09)*, pp. 567–574. Society for Artificial Intelligence and Statistics.

Vanhatalo J., Vehtari A. (2008). Modelling local and global phenomena with sparse Gaussian processes. In McAllester D.A., Myllymäki P., editors, *Proceedings of the 24th Conference in Uncertainty in Ar-*

tificial Intelligence (UAI'2008), pp. 571–578. AUAI Press.

Wikle C.K., Milliff R.F., Nychka D., Berliner L.M. (2001). Spatio-temporal hierarchical Bayesian modeling: Tropical ocean surface winds. *Journal of the American Statistical Association*, 96:382–397.

Zhang H. (2007). Maximum-likelihood estimation for multivariate spatial linear coregionalization models. *Environmetrics*, 18(2):125–139.