Title

Author 1*
Department of YYY, University of XXX
and
Author 2
Department of ZZZ, University of WWW

November 20, 2017

Abstract

We use a fully Bayesian framework to semiparametrically estimate covariance functions of stationary Gaussian processes from irregularly-spaced data. The covariance functions are modeled in the spectral domain on the log-log scale using penalized natural splines, which results in a flexible class of smooth densities that have power law tails. The Fourier transforms required to construct likelihoods from the spectral densities are computed using standard Monte Carlo integration. These calculations require massive computational effort, but are highly parallelizable. By utilizing GPUs, computational speeds are competitive with standard parametric covariance model estimation.

Keywords: 7 or fewer keywords

1 Background

1.1 Gaussian Processes

Consider a *spatial stochastic process* [6] $\{Y(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D} \subseteq \mathbb{R}^d\}$ that varies over some continuous spatial domain $\mathcal{D}$. We will focus on the most common case for spatial statistics, where the dimension is $d = 2$, although the methods presented here are valid for any d. For a finite collection of spatial locations $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\} \subset \mathcal{D}$, $\boldsymbol{Y} = (Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n))^T$ is a random vector where each element is associated with one location. The multivariate distribution of $\boldsymbol{Y}$ contains information about the spatial dependencies between all n locations.

*The authors gratefully acknowledge

The distribution of the process $\{Y(\boldsymbol{s})\}$ may be described through its finite-dimensional joint distributions

$$F(y_1, \ldots, y_n; \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) = P(Y(\boldsymbol{s}_1) \leq y_1, \ldots, Y(\boldsymbol{s}_n) \leq y_n) \tag{1}$$

for every value of n and every set of n spatial locations in $\mathcal{D}$. A *Gaussian process* is a special case in which every distribution in (1) is multivariate normal [6]. As a result, the distributions are all completely characterized by their means and covariance matrices, which makes Gaussian processes much easier to work with than non-Gaussian spatial stochastic processes.

1.2 Stationarity and Isotropy

Throughout this work we will be making two key assumptions: *stationarity* and *isotropy*. A stationary Gaussian process is one that does not vary with spatial shifts. This implies that, for any lag vector $\boldsymbol{h} \in \mathbb{R}^d$,

$$E[Y(\boldsymbol{s})] = E[Y(\boldsymbol{s} + \boldsymbol{h})] = \boldsymbol{\mu}$$

and

$$\text{Cov}(Y(\boldsymbol{s}), Y(\boldsymbol{s} + \boldsymbol{h})) = \text{Cov}(Y(\boldsymbol{0}), Y(\boldsymbol{h})) = C(\boldsymbol{h}).$$

Here $C(\boldsymbol{h})$ is called the *covariance function*.

For isotropic Gaussian processes, the covariance function depends on $\boldsymbol{h}$ only through its magnitude $||\boldsymbol{h}||$. In this setting, we can express the covariance function simply as $C : [0, \infty) \to \mathbb{R}$. If we assume without loss of generality that the process is standardized, we can further stipulate that $C(0) = 1$ and $E[Y(\boldsymbol{s})] = 0$ for all $\boldsymbol{s} \in \mathcal{D}$.

1.3 Covariance Function Estimation

Because Gaussian processes are completely characterized by their mean and covariance functions, estimating C given a collection of observations $\{y_1, \ldots, y_n\}$ is a topic of great interest [15] [1]. The problem of estimating C differs from the general problem of function estimation because only a restricted class of functions will produce a valid Gaussian process. In particular, C must belong to the class of *positive definite functions*, which is defined as the set of all functions C such that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j C(\boldsymbol{s_i} \boldsymbol{s}_j) > 0$$

for any n, any $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\} \in \mathcal{D}$, and any $a_1, \ldots, a_n \in \mathbb{R}$. This condition guarantees that the joint distribution of any finite collection of observations $\{Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n)\}$ has a positive definite covariance matrix. This turns out to be a strong restriction, and it is very difficult in general to directly estimate C in such a way that forces positive definiteness.

The classical approach to overcoming this difficulty is to select a parametric family of covariance functions that is known to be positive definite for a given range of the parameters, and to estimate the parameters using moment- or likelihood-based methods[6]. A popular choice of parametric families is the *Matérn* class of functions [8], which take the form

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)} \left(\frac{2\nu^{1/2}h}{\rho} \right)^{\nu} \mathcal{K}_\nu \left(\frac{2\nu^{1/2}h}{\rho} \right), \qquad \sigma^2, \nu, \rho > 0, \tag{2}$$

where $\mathcal{K}_\nu$ is a modified Bessel function of the third kind. This is a three-parameter family, with σ controlling the marginal variance, ρ controlling the range of correlation, and ν controlling the smoothness of the sample paths. All functions in this class are positive definite for observations in any dimension [13]. Some examples of Matérn covariance functions are shown in Figure 1.

Another positive definite family, which we will use in examples below, is the *exponentially damped cosine* class,

$$C(h) = \sigma^2 \exp\left(-\tau \frac{h}{\lambda}\right) \cos\left(\frac{h}{\lambda}\right), \qquad \tau \geq \frac{1}{\tan\frac{\pi}{2d}}, \quad \sigma^2, \lambda > 0. \tag{3}$$

Unlike the Matérn class, the damped cosine class allows for negative correlations at certain distances. This non-monotonic behavior is known in geostatistical applications as a *hole effect* [16], and is shown in Figure 2. The restriction on τ that depends on d is necessary for functions of this form to be positive definite. The damped cosine family is not nearly as commonly used as the Matérn family, but it is useful in some applications because allows for stochastically periodic behavior as well as negative correlations, which a Matérn family cannot capture. However, it does not share the attractive feature of the Matérn family that the smoothness of the process (i.e. the degree of differentiability) is controlled by a parameter that can be estimated from the data. The smoothness is closely related to the local behavior of the process, which crucial to obtaining good performance when interpolating the process at unobserved locations [13].

The strategy of fitting the data to a parametric family of covariance functions can produce a good estimate if the true covariance structure is close to the structure assumed by the choice of parametric family. However, there is no guarantee that this will be the case. For instance, if

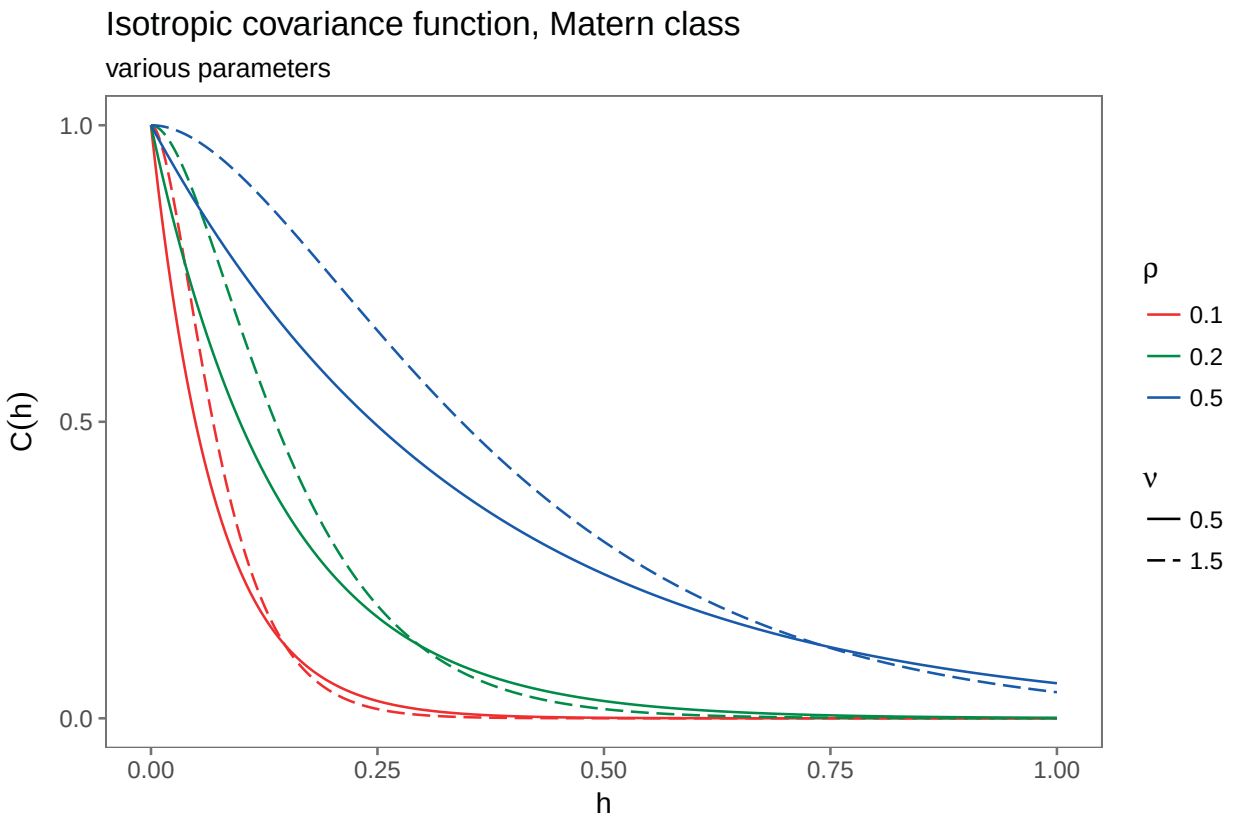

Figure 1: The Matérn isotropic covariance function (2) for various choices of ν and ρ, each with $\sigma^2 = 1$.

the true covariance contains a hole effect, there is no way for a Matérn model to capture it since $C(h) > 0$ for all h. More preferable would be a flexible method that allows the spatial dependence in the data itself, rather than the choice of parametric family, to determine the shape of C.

1.4 The Spectral Domain and Bochner's Theorem

The method proposed in this work takes advantage of the result from Bochner [2] which says that a real-valued continuous function C is positive definite if and only if it is the Fourier transform of a symmetric, non-negative, finite measure F on $\mathbb{R}^d$. That is, C is positive definite if and only if

$$C(\boldsymbol{h}) = \int_{\mathbb{R}^d} \exp(i\boldsymbol{h}^T\boldsymbol{\omega})\, dF(\boldsymbol{\omega}). \tag{4}$$

In most circumstances, the measure F has a Lebesgue density $f(\boldsymbol{\omega})$, which is referred to as the *spectral density* [6]. Furthermore, because we are assuming that the Gaussian process described

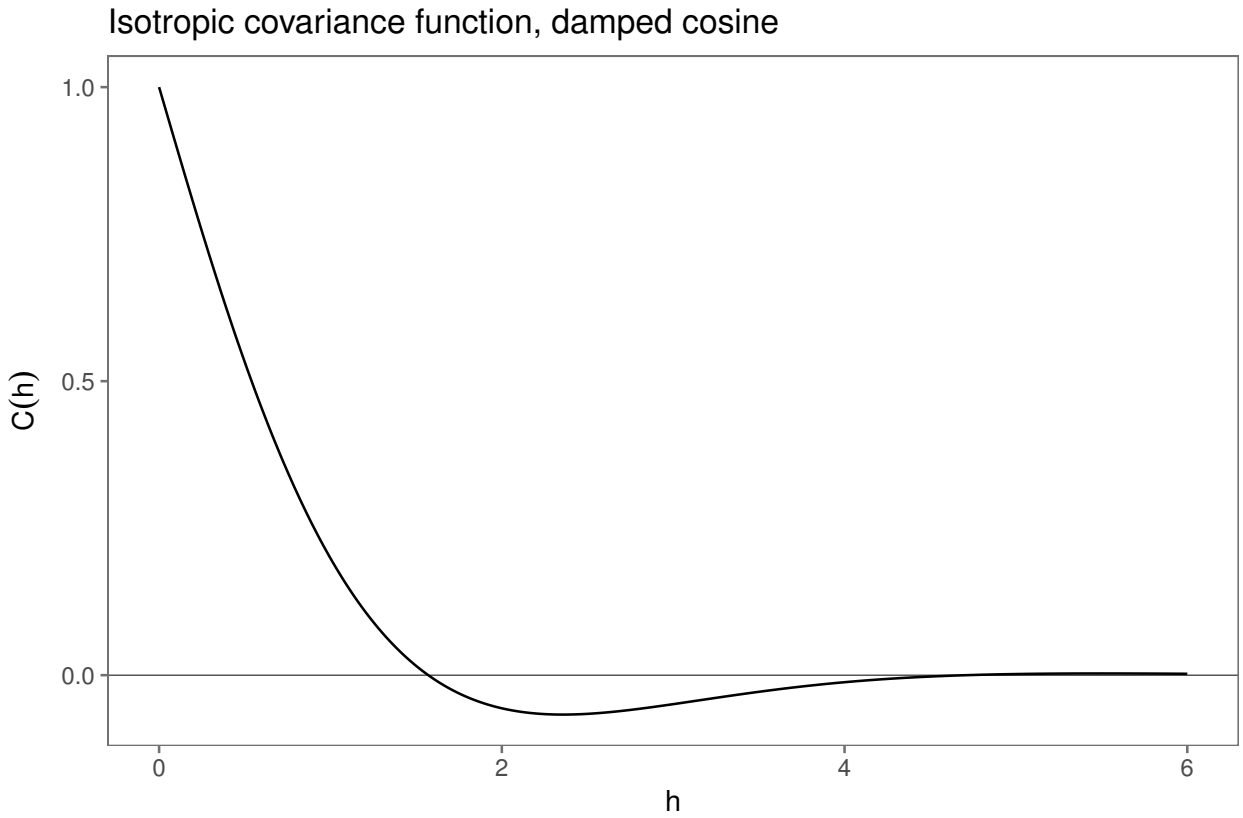

Figure 2: The damped cosine isotropic covariance function (3) for $\tau = 1$, $\lambda = 1$, and $\sigma^2 = 1$. Notice that $C(h) < 0$ for certain values of h.

by C is isotropic, C is a function of a scalar h, so that

$$C(h) = \int_{\mathbb{R}} \cos(h\omega)\ f(\omega)\ d\omega. \tag{5}$$

Stein [13] argues that for Gaussian process covariance functions to be realistic models for spatial phenomena, spectral densities $f(\omega)$ must be heavy tailed.

The relationship between the covariance function and the spectral density given in (5) suggests an alternative way to estimate the covariance function $C(h)$. If we can estimate the the symmetric spectral density $f(\omega)$, then we are assured that the corresponding covariance function $C(h)$ is positive definite. Flexibly modeling $C(h)$ directly is difficult because of the restriction of positive definiteness, but flexibly modeling $f(\omega)$ is easy because it can be any symmetric density.

There have been previous approaches for non- and semi-parametric methods that estimate $C(h)$ using the spectral domain. Most closely related to our approach is that of Im, Stein, and Zhu [10], who estimate the spectral density using a combination of cubic B-splines with an explicitly specified

algebraically decaying tail. Constraints are placed on the spline coefficients to ensure that the spectral density is non-negative, thereby resulting in a covariance function is positive definite. Im, Stein, and Zhu use the fact that when under isotropy, (4) can be written as an integral over only one dimension,

$$C(h) = 2^{(d-2)/2}\Gamma(d/2)\int_0^\infty (hu)^{-(d-2)/2} J_{(d-2)/2}(hu)\, dG(u),$$

where $J_\nu(\cdot)$ is the Bessel function of the first kind of order ν. The spline-based spectral density is transformed into a covariance function using numerical integration, which is then used to construct the likelihood of the spline coefficients given the data. This likelihood is be maximized over the constrained set of spline coefficients to produce $d\hat{G}(u)$, and hence $\hat{C}(h)$. Our method follows this same concept, with some notable differences outlined in Section 2.

Another, earlier method was put forth by Hall, Fisher and Hoffmann [7]. They proposed a multistep process which begins with a kernel estimate of the covariogram. Because the kernel estimate is not necessarily positive definite, they numerically compute its Fourier transform, set all frequencies beyond the first negative value to zero, and then numerically Fourier transform it back to the spatial domain. In the simulation study in Section 3, we compare our method to the one from Hall et al., as well as to the Matérn model fit by maximum likelihood.

In Section 4, we apply our method to data from a paper by Singh et al. [12], where we model the thickness of a thin film semiconductor in a photovoltaic cell as a Gaussian process. The results are discussed in Section 5, as well as possibilities for improvements and future directions for this work.

2 Model and Estimation

Throughout this section, assume that the data is observed at n (not necessarily evenly spaced) spatial locations $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$. These locations exist on some bounded domain $\mathcal{D} \subseteq \mathbb{R}^d$. The data, $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, is a realization of the random vector $\boldsymbol{Y} = (Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n))^T$, which constitute observations of single instance of a Gaussian process with mean 0 and stationary isotropic covariance function $C(h)$. We aim to estimate the covariance function.

2.1 Estimating the Covariance Function

Under the assumptions of stationarity and isotropy, the random vector $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{\Sigma})$, where the ijth element of $\boldsymbol{\Sigma}$ is

$$\Sigma_{ij} = \text{Cov}(y_i, y_j) = C(||\boldsymbol{s}_i - \boldsymbol{s}_j||) = C(h_{ij}).$$

The scalar h_{ij} is the Euclidean distance between locations $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$. We can use (5) to obtain an integral representation for each element of $\boldsymbol{\Sigma}$,

$$\Sigma_{ij} = \int \cos(h_{ij}\omega)\ f(\omega)\ d\omega. \tag{6}$$

Rather than model the covariance function directly, we will model the spectral density with a semiparametric Bayesian model and use (6) to construct the data likelihood.

2.2 Calculating the Likelihood

Consider a rich family of spectral densities $f_{\boldsymbol{\theta}}(\omega)$ indexed by a parameter vector $\boldsymbol{\theta}$. Since $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{\Sigma})$, the likelihood of $\boldsymbol{\theta}$ given the data $\boldsymbol{y}$ is

$$\ell(\boldsymbol{\theta}; \boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{y}, \tag{7}$$

where $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is defined as in (6). In general, the solution to the Fourier transform integral in (6) will not be available analytically.

Our strategy is to replace each element $\Sigma_{ij}(\boldsymbol{\theta})$ with a simple Monte Carlo approximation. Suppressing dependence on $\boldsymbol{\theta}$ for notational convenience, we will use

$$\Sigma_{ij} = \int \cos(h_{ij}\omega)\ f(\omega)\ d\omega \approx \frac{1}{M}\sum_{m=1}^{M}\cos(h_{ij}\widetilde{\omega}_m) = \widehat{\Sigma}_{ij}, \tag{8}$$

where $\widetilde{\omega}_1, \ldots, \widetilde{\omega}_M$ are M samples from $f_{\boldsymbol{\theta}}(\omega)$. We then plug $\widehat{\boldsymbol{\Sigma}}$ into the likelihood (7) to estimate $\boldsymbol{\theta}$.

It is natural to ask how closely the likelihood evaluated using (6) approximates the true likelihood. The following theoretical result shows that the estimated likelihood $\hat{\ell}(\boldsymbol{\theta}; \boldsymbol{y})$ converges to the true likelihood $\ell(\boldsymbol{\theta}; \boldsymbol{y})$ almost surely as the number of Monte Carlo samples approaches infinity, even when the spectral density has heavy tails. A proof is given in the Appendix.

Theorem 1. *Let* $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ *be a vector of observations from a mean-zero stationary, isotropic Gaussian process with covariance function* $C(h)$*, taken at locations* $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathcal{D} \subseteq \mathbb{R}^d$ *in some bounded domain* $\mathcal{D}$*. For some symmetric density* $f(\omega)$*, let* $\ell(\boldsymbol{y})$ *be the likelihood*

$$\ell(\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y},$$

where $\Sigma_{ij} = \int \cos(h_{ij}\omega) f(\omega) d\omega$*, and* $\hat{\ell}(\boldsymbol{y})$ *be the Monte Carlo approximated likelihood*

$$\hat{\ell}(\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\widehat{\boldsymbol{\Sigma}}|) - \frac{1}{2}\boldsymbol{y}^T\widehat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{y}$$

where $\widehat{\Sigma}_{ij} = \frac{1}{M}\sum_{m=1}^{M} \cos(\widetilde{\omega}_m h_{ij})$*, and* $\widehat{\omega}_1, \ldots, \widehat{\omega}_M$ *are an iid sample from* $f(\omega)$*. Then as* $M \to \infty$*,*

$$\hat{\ell}(\boldsymbol{y}) \to \ell(\boldsymbol{y}) \text{ a.s.-}f_\omega.$$

Theorem 1 says that as long as we are willing to draw a large enough collection of Monte Carlo samples from a spectral density $f_{\boldsymbol{\theta}}(\omega)$, the Monte Carlo estimated likelihood of $\boldsymbol{\theta}$ will get arbitrarily close to the exact likelihood. Furthermore, this remains true even when $f_{\boldsymbol{\theta}}(\omega)$ has heavy tails, as we expect for realistic models of spatial phenomena.

2.3 Semiparametric Modeling of the Spectral Density

Since we don't want to impose a parametric form for $f(\omega)$, we will model it semiparametrically using splines. According to Bochner's theorem, $f(\omega)$ is a symmetric density, so it suffices to restrict our attention to modeling $f(\omega)$ for $\omega > 0$.

Rather than modeling $f(\omega)$ directly, we apply a straightforward variable transformation and model $\log f(\log \omega)$ instead. Working on the log-log scale is a natural way to flexibly specify densities with heavy tails. It is easy to show that any density with power law tails has linear tails on the log-log scale. Suppose $f_X(x) = cx^{-b}$, $b > 1$, and let $Y = \log X$. Then

$$\begin{aligned} f_Y(y) &= f_X(e^y)\frac{d}{dy}(e^y) \\ &= ce^{-by}e^y \\ &= ce^{-(b-1)y}, \end{aligned}$$

and so

$$\log f_Y(y) \propto -(b-1)y.$$

This fact is illustrated further in Figure 3.

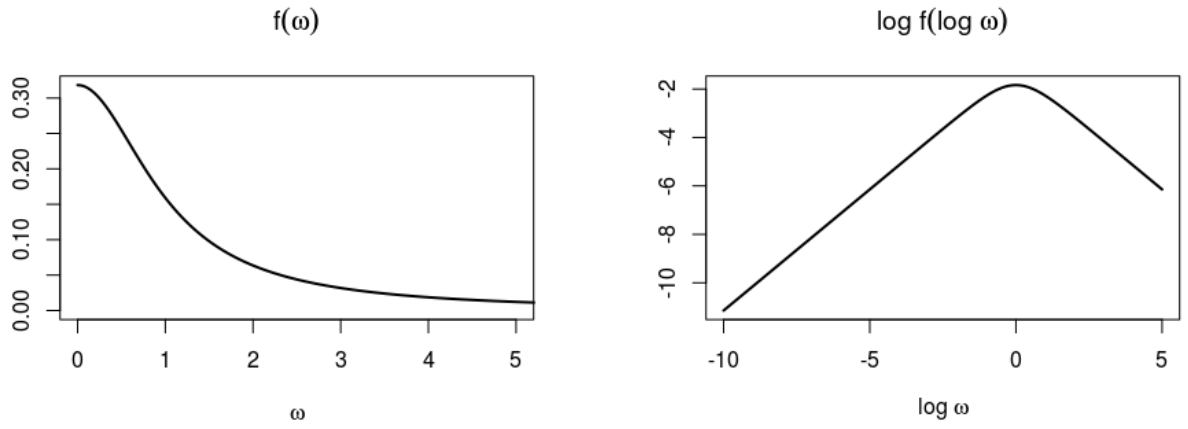

Figure 3: A density $f(\omega)$ with power law tails (left) and the same density transformed to the log-log scale (right). Both the left and right tails of the transformed density are linear.

An important benefit to working in this log-transformed space is that linear tails are ideal for modeling the curve using a natural spline basis, which is constructed to ensure linearity beyond the outermost (*boundary*) knots. If we are willing to assume that the tails of $\log f(\log \omega)$ are truly linear, then fitting it with natural splines will result in a good approximation over then entire domain $\log \omega \in (-\infty, \infty)$ without needing to use a large number of basis functions.

It might appear that requiring that $f(\omega)$ have power law tails is more restrictive than we would like. After all, the goal is to avoid the need to restrict $C(h)$ to a particular parametric family. However, the class of covariance functions that result from the Fourier transform of power law spectral densities is significantly broader than any of the parametric families, yet Bochner's theorem still guarantees that they are positive definite. In fact, this class covers most covariance models of practical interest, including the entire Matérn class with finite ν. Gaussian processes with spectral densities that decay quickly, e.g. with exponential tails, yield realizations that are generally regarded as unrealistically smooth [13].

To model $\log f(\log \omega)$ using natural cubic splines, let

$$\log f(\log \omega) = \sum_{k=1}^{K} \beta_k B_k(\log \omega), \tag{9}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)^T$ are coefficients associated with K known cubic spline basis functions $B_1(\cdot), \ldots, B_K(\cdot)$. The K basis functions are centered around a collection of K locations referred to as knots.

2.4 Estimating the Spline Coefficients

For a given collection of basis functions, the likelihood of the parameter vector $\boldsymbol{\theta} = \boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)^T$ can be approximated by plugging (6) into (7). To be explicit, in (7) the estimated covariance matrix $\boldsymbol{\Sigma}$ is expressed as a function of $\boldsymbol{\beta}$ because

$$\Sigma_{ij} = \int \cos(h_{ij}\omega)\, f_{\boldsymbol{\beta}}(\omega)\, d\omega \approx \frac{1}{M} \sum_{m=1}^{M} \cos(h_{ij}\widetilde{\omega}_m) = \widehat{\Sigma}_{ij},$$

$$\widetilde{\omega}_m \overset{\text{i.i.d.}}{\sim} f_{\boldsymbol{\beta}}(\omega),$$

and

$$\log f_{\boldsymbol{\beta}}(\log \omega) = \sum_{k=1}^{K} \beta_k B_k(\log \omega).$$

The only free parameters in this procedure are the elements of $\boldsymbol{\beta}$. This is summarized in Algorithm 1.

To quickly generate the samples $\widetilde{\omega}_1, \ldots, \widetilde{\omega}_M$, we transform $\log f_{\boldsymbol{\beta}}(\log \omega)$ back to $f_{\boldsymbol{\beta}}(\omega)$, integrate numerically, and sample via the inverse CDF method.

Since we can calculate the likelihood as a function of the data and $\boldsymbol{\beta}$, we can estimate $\boldsymbol{\beta}$ using a Markov chain Monte Carlo (MCMC) algorithm.

To complete the model, we specify priors for $\boldsymbol{\beta}$ according to the Bayesian formulation of penalized splines introduced in [11]. The penalization corresponds to priors that specify to a second-order random walk, with

$$\beta_k \mid \tau^2 \sim \mathcal{N}(2\beta_{k-1} - \beta_{k-2},\ \tau^2)$$

where

$$\beta_2 \mid \tau^2 \sim \mathcal{N}(\beta_1,\ \tau^2)$$

Algorithm 1 Calculating the log likelihood $\hat{\ell}(\boldsymbol{\beta}; \boldsymbol{y})$

procedure LIKELIHOOD($\boldsymbol{\beta}, \boldsymbol{y}$)
  $\log f_{\boldsymbol{\beta}}(\log \omega) = \sum_{k=1}^{K} \beta_k B_k(\log \omega)$ ▷ k preselected knot locations
  Sample $\widetilde{\omega}_1, \ldots, \widetilde{\omega}_M$ from $f_{\boldsymbol{\beta}}(\omega)$ ▷ Inverse CDF method; M large
  for $1 \leq i, j \leq n$ **do**
    $\widehat{\Sigma}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \cos(h_{ij} \widetilde{\omega}_m)$ ▷ h_{ij}: distance between locations $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$
  end for
  $\hat{\ell}(\boldsymbol{\beta}; \boldsymbol{y}) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\widehat{\boldsymbol{\Sigma}}| - \frac{1}{2} \boldsymbol{y}^T \widehat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{y}$
end procedure

and

$$\beta_1 \mid \tau^2 \sim \mathcal{N}(0,\ \tau^2).$$

The variance parameter τ^2 controls the smoothness of the spline fit, and must also be estimated. Its prior is specified as

$$\tau^2 \sim \mathcal{N}^+(0,\ \sigma_\tau^2),$$

the half-normal distribution with some variance hyperparameter σ_τ^2.

With the priors specified, we can proceed with the MCMC algorithm in Algorithm 2. We use a Metropolis-Hastings update to determine whether or not to accept the joint proposal $\boldsymbol{\beta}$. The result is random draws from the posterior distribution of $\boldsymbol{\beta} \mid \boldsymbol{y}$.

2.5 Computational Challenges

The primary difficulty inherent in this method is that it is extraordinarily computationally intensive, requiring a Monte Carlo approximation for every element of the covariance matrix. For a moderately large number of observations, say $n = 400$, we need to estimate $\frac{n(n-1)}{2} = 79800$ separate elements. For each of these elements, we need to perform a Monte Carlo integration with a large number of samples. This all happens within one likelihood calculation, and we need to compute the likelihood at every iteration of the MCMC algorithm. The computational costs would be prohibitive for a naïve implementation of this method.

Fortunately, most of these difficulties can be assuaged by taking advantage of parallel computing. Algorithm 2 requires previous information $\boldsymbol{\beta}_{i-1}$ to compute $\boldsymbol{\beta}_i$, so unfortunately it would be

Algorithm 2 Metropolis-Hastings Sampler

procedure MCMC($\boldsymbol{\beta}, \boldsymbol{y}$)
  $\boldsymbol{\beta}_1 \leftarrow \boldsymbol{\beta}$ ▷ Initialization
  for $2 \leq i \leq N$ **do** ▷ N large
    Propose new $\boldsymbol{\beta}^* \sim q(\cdot)$
    $\ell_0 = \hat{\ell}(\boldsymbol{\beta}_{i-1}, \boldsymbol{y})$ ▷ See Algorithm 1
    $\ell_1 = \hat{\ell}(\boldsymbol{\beta}^*, \boldsymbol{y})$
    $a = \Big[\pi(\boldsymbol{\beta}^*) \cdot \ell_1 \cdot q(\boldsymbol{\beta}_{i-1}|\boldsymbol{\beta}^*)\Big] / \Big[\pi(\boldsymbol{\beta}_{i-1}) \cdot \ell_0 \cdot q(\boldsymbol{\beta}^*|\boldsymbol{\beta}_{i-1})\Big]$
    Generate $u \sim Unif(0, 1)$
    if $u < \min(a, 1)$ **then**
      $\boldsymbol{\beta}_i \leftarrow \boldsymbol{\beta}^*$ ▷ Accept the proposal
    else
      $\boldsymbol{\beta}_i \leftarrow \boldsymbol{\beta}_{i-1}$ ▷ Reject the proposal
    end if
  end for
end procedure

extremely difficult to restructure it in a way where multiple updates are being performed simultaneously. However, it is possible to parallelize aspects of the likelihood calculation in Algorithm 1.

Consider the **for** loop in Algorithm 1. For each value of i and j, the computation of

$$\widehat{\Sigma}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \cos(h_{ij}\widetilde{\omega}_m)$$

depends on two values: h_{ij}, the distance between $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, and $\widetilde{\boldsymbol{\omega}}$, the vector of M samples from $f(\omega)$. It does not depend on any previously computed element of $\widehat{\boldsymbol{\Sigma}}$. Therefore, performance would improve dramatically if we could calculate the estimates simultaneously, in parallel.

There are several options for speeding up a parallelizable problem. The simplest, and least reliant on expensive additional hardware, is *parallel processing* [14]. Today, most personal computers ship with CPUs that contain 2, 4, or even 8 computing cores. Each core is capable of running a single set of sequential instructions. By using tools such as OpenMP, users can write code that will assign a set of instructions to every available core. The instructions will execute simultaneously, and the results are gathered together upon completion. Of course, the potential speedup is

limited by the number of cores available in the CPU—if there are 4 cores, the instructions will be executed in about one fourth of the time.

This type of parallel processing works well for problems like obtaining multiple MCMC chains at the same time. For the estimation of $\mathbf{\Sigma}$ in this context, however, it is not optimal. Recall that for $n = 400$, we need to estimate 79800 elements of $\mathbf{\Sigma}$. Reducing this by a factor of 4 is helpful, but ideally we would like to perform all 79800 at once, not just take them four at a time.

Another option is to utilize a computing cluster. Clusters usually are large rooms filled with racks of computers, connected in such a way that one process can be spread uniformly across them and use hundreds or thousands of cores at once [14]. One downside to this approach is that access to such a cluster is difficult to get, outside of a large company or academic institution. Beyond that, though, latency starts to become a problem when dealing with clusters. Because the cores are spread over many physical computers, the increases in computational speed becomes overwhelmed by the large amount of time it takes to share data between the cores. As a result, the overall performance increase may not be as large as we might expect.

A third option is *GPU computing*. Originally created to run video games, graphics processing units, or GPUs, have become increasingly popular for parallel computing. CPUs contain a small number of cores that are optimized to handle highly complex instructions quickly. By contrast, GPUs are made up of thousands of simple, highly efficient cores that are optimized for simpler instructions and designed to work in parallel with each other. Because the GPU cores lack many of the sophisticated features of CPU cores (a sacrifice made in exchange for speed and efficient exchange of data), the programmer must have a more intricate knowledge of the architecture of their particular GPU in order to get the best possible speedup. Fortunately, directives such as OpenACC let the compiler make most of the architecture-related decisions instead of leaving them in the hands of the user.

To take full advantage of the highly parallelizable nature of the covariance estimation problem, I have chosen the GPU option. I wrote the code to carry out the Monte Carlo integrations in CUDA C. CUDA is an API provided by Nvidia that allows users to write programs in C or Fortran that interact with Nvidia brand GPUs. Computations were done on three different hardware setups:

- Nvidia GeForce GTX 1060 GPU, on a personal computer running Ubuntu 16.04;
- Nvidia Tesla P100 GPU, provided by Penn State University's Institute for CyberScience

Advanced CyberInfrastructure (ICS-ACI);

- Nvidia Tesla K80 GPU, running on an Amazon EC2 instance.

With these tools, the computations can be performed fairly quickly. On the Tesla P100, one entire likelihood calculation described by Algorithm 1 with $M = 50000$ and $N = 400$ completes in just under one second.

To illustrate the power of parallel computing, and to show that this element-wise approximation to the covariance matrix is a feasible approach, Figure 4 shows the results of an experiment comparing the runtime of the Monte Carlo approximation from Algorithm 1 to the runtime of the exact covariance calculation. That is, for a fixed N, generate 100 sets of points $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$ and their distances $h_{ij} = ||\boldsymbol{s}_i - \boldsymbol{s}_j||$. Also generate $M = 20000$ samples $\widetilde{\omega}_1, \ldots, \widetilde{\omega}_M$ from the spectral density $f(\omega)$ corresponding to a Matèrn covariance function. Since the computational cost of the random sample generation is constant regardless of N, it is not included when timing.

$$\Sigma_{ij} = C(h_{ij})$$

and

$$\widehat{\Sigma}_{ij} = \frac{1}{M} \sum_{m=1}^{M} \cos(h_{ij}\widetilde{\omega}_m)$$

100 times each, and compare the median runtimes as a function of N.

Notice from Figure 4 that for large enough N, it actually becomes faster to perform a Monte Carlo approximation to the elements of Σ than to directly plug in the distances to the covariance function C. In this experiment N needed to be greater than about 2000, but this is of course hardware dependent.

This is not a surprising result considering the parallelized nature of the Monte Carlo approximation. As N increases, the exact calculation must handle all additional computations sequentially, whereas the Monte Carlo approximation can simply allocate more cores that execute their tasks simultaneously. As a result, the Monte Carlo runtime increases more slowly with N than the exact covariance runtime.

However, there are nontrivial computational costs associated with GPU computing that cause it to be significantly slower than the exact alternative when N is not large. In Figure 4, it appears that the Monte Carlo runtime is more or less unchanged from $N = 40$ to $N = 400$.

The bottleneck here is data transfer and allocation. For small N, the time required to do the calculations themselves is negligible compared to the time required to allocate memory on the GPU, transfer the spectral density samples from the CPU to the GPU, combine the results, and transfer them back to the CPU. This is the reason that sometimes, seemingly paradoxically, parallelizing a routine can cause it to execute more slowly than the equivalent sequential routine. But for large enough N, we expect the speedup to be substantial.

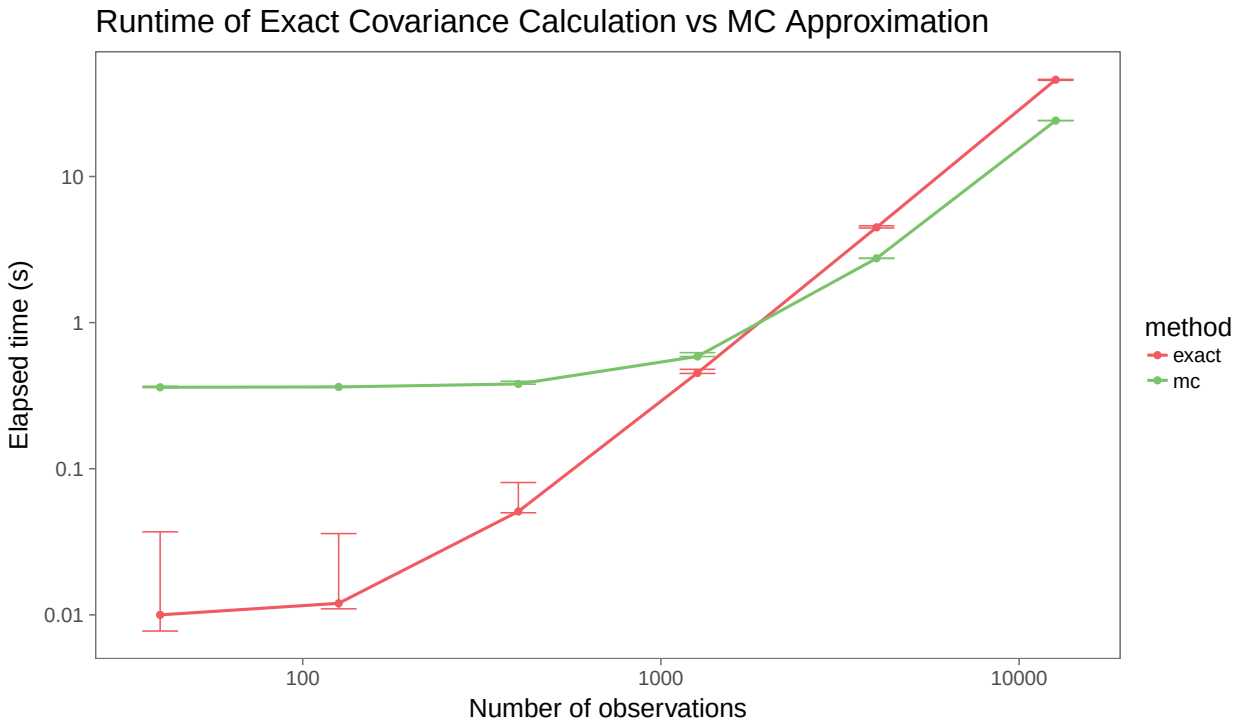

Figure 4: Elapsed time taken to calculate the element-wise Monte Carlo approximation to the covariance matrix corresponding to a Matérn covariance function, versus calculating it directly using $C(h)$. The covariance matrix is $n \times n$, where n is the number of observations. Each method was run 100 times for every value of n. The medians are plotted with dots and connected with a line. The error bars represent the range from the first to third quartiles—some times for the exact calculation were misleadingly high due to CPU wakeup. GPU computations were done on a Tesla P100. Note the logarithmic axes.

3 Simulation Study

In this section we will present some results that compare the method presented in Section 2, which we will refer to as the "spline method," to some existing techniques for estimating covariance functions. In Section 3.1, we examine the accuracy of the likelihood estimate from Algorithm 1. In Sections 3.2 and 3.3, we compare the spline method to existing methods with regard to prediction performance and how closely they estimate the true covariance function, respectively.

3.1 Comparing Exact and Estimated Log Likelihoods

Assume we observe $\boldsymbol{y} = (y_1, \ldots, y_{400})^T$ from a Gaussian process where the covariance function is Matérn (see (2)) with known parameters ν, ρ, and σ. The observations come from locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{400}$ spread randomly over the unit square in $\mathbb{R}^2$. Because we know $C(h)$, we can calculate the covariance matrix exactly:

$$\boldsymbol{\Sigma}_{ij} = C(||\boldsymbol{s}_i - \boldsymbol{s}_j||).$$

From this we can find the exact likelihood.

The spectral density corresponding to the Matérn covariance function (2) is

$$f(\omega) = \frac{\sigma g(\nu, \rho)}{\left(\frac{4\nu}{\rho^2} + \omega^2\right)^{\nu + d/2}},$$

where $d = 2$ is the dimension of the spatial domain and

$$g(\nu, \rho) = \frac{\Gamma\left(\nu + \frac{d}{2}\right)(4\nu)^\nu}{\pi^{d/2}\rho^{2\nu}\Gamma(\nu)}.$$

Because we can calculate the spectral density directly in this scenario, we can avoid fitting the splines and estimating $\boldsymbol{\beta}$. By bypassing this step, we test just the procedures for sampling from $f(\omega)$ and estimating the log likelihood.

To perform the test, we chose two values for ν and six values for ρ. The σ^2 parameter just controls the marginal variance $\text{Var}(\boldsymbol{Y}(\boldsymbol{s}))$, and so it can be factored out of the covariance matrix $\boldsymbol{\Sigma}$. For each combination of parameter settings, we generated 100 Gaussian process realizations and compared the true and estimated log likelihoods. The results are summarized in Figure 5. Note that the estimated likelihoods are close to their true values in all cases, and the choice of ρ does not appear to significantly affect the accuracy.

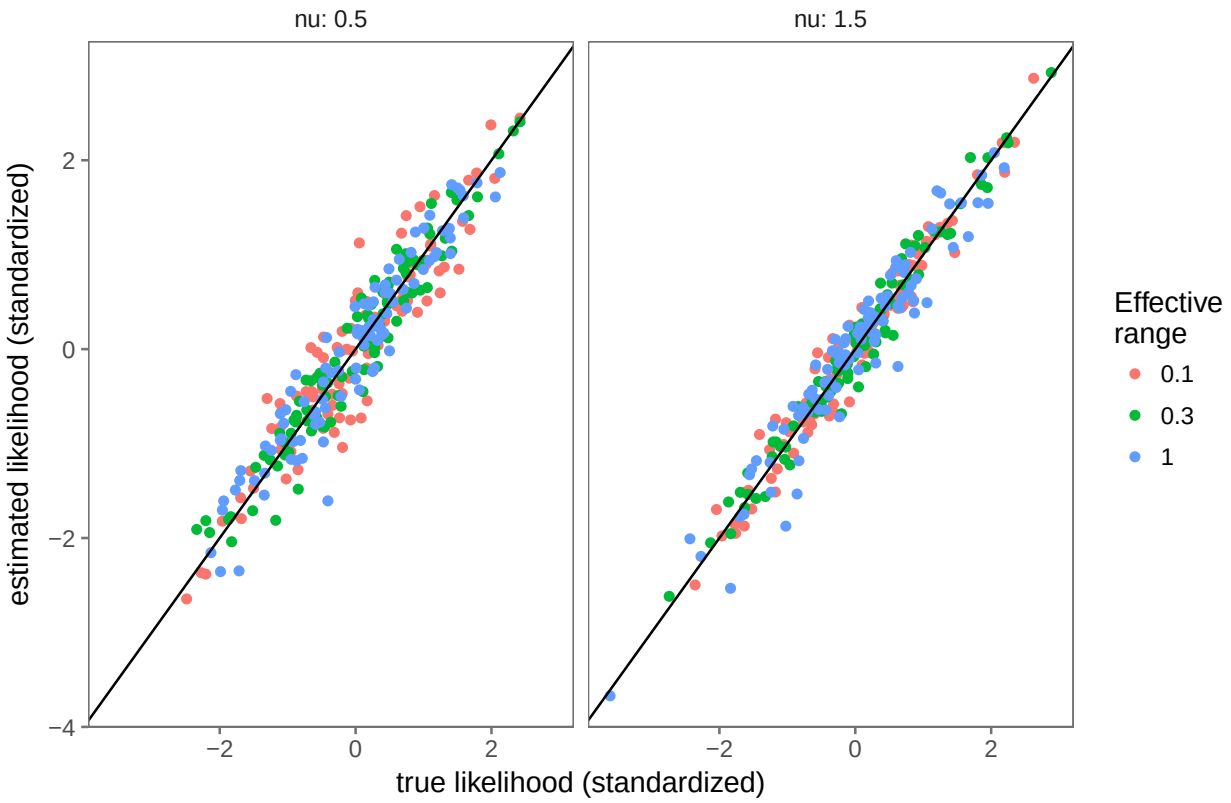

Figure 5: Comparing true and estimated log likelihoods for varying values of ρ (given above each plot). Values in each plot were centered and scaled for easier comparison. The likelihoods for the two values of ν, 0.5 and 1.5, yield a similar pattern.

We ran a second version of this test that incorporated the $\boldsymbol{\beta}$ estimation. Still assuming the true $f(\omega)$ is known, we took 50 equally spaced points along $\log f(\log \omega)$ and fit natural splines to those points. This gave us some $\boldsymbol{\beta}_{\text{opt}}$, the values for $\boldsymbol{\beta}$ that most closely approximate $\log f(\log \omega)$. Then using $\boldsymbol{\beta}_{\text{opt}}$ we estimated the likelihood and again compared it to the true likelihood. Results are shown in Figure 6. The estimated likelihoods are still comparable to their true values, despite the fact that we are sampling from a spline approximation to $f(\omega)$ rather than $f(\omega)$ itself.

As an additional test that we are correctly estimating the true likelihood, we simulated another set of 400 observations $\boldsymbol{y} = (y_1, \ldots, y_{400})^T$ from a Gaussian process with Matérn covariance function. We treat ρ and ν as fixed and known, and used MCMC to find the posterior distribution of $\rho|\boldsymbol{y}$ using the true likelihood function, and again using the estimated likelihood from Algorithm 1. One result from this test is shown in Figure 7. We see that the two posterior distributions are very

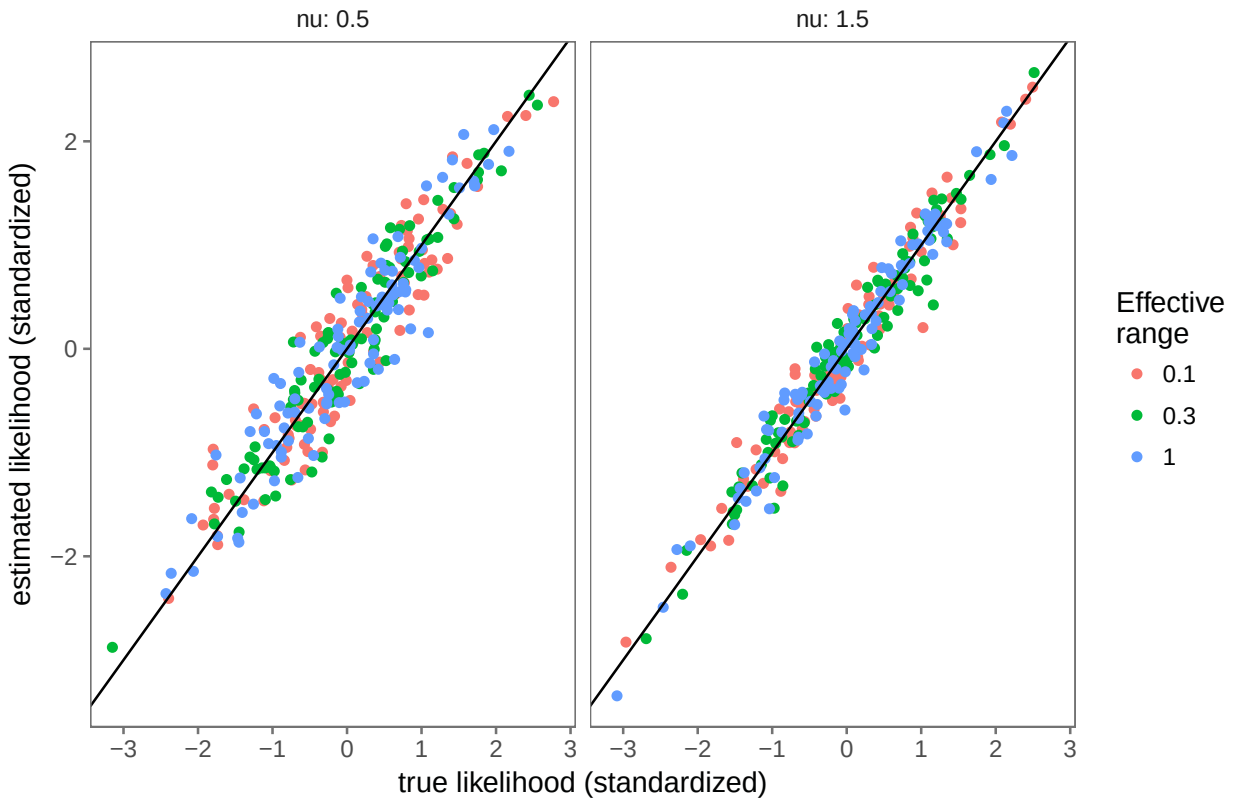

Figure 6: Comparing true and estimated log likelihoods for varying values of ρ (given above each plot). Unlike the results in Figure 5, these estimates were generated not by sampling from $f(\omega)$ directly, but by sampling from a density approximating $f(\omega)$ fit using splines.

similar, futher indicating that our likelihood estimate is reasonable for use in our more complicated semiparametric MCMC (Algorithm 2).

3.2 Prediction Performance

Next, we put the entire procedure from Algorithm 2 into motion. First, $n = 400$ observations $\boldsymbol{y} = (y_1, \ldots, y_{400})^T$ are drawn from a Gaussian process with a Matèrn covariance function with $\nu = 1.5$, $\rho = 0.1548$, and $\sigma = 1$. This particular value of ρ was chosen to obtain an effective range of 0.3. The observation locations are randomly spread across the unit square in $\mathbb{R}^2$ (see Figure 8). An arbitrary initial $\boldsymbol{\beta}_0$ is chosen, and Algorithm 2 is run for 10000 iterations. At each iteration i, one likelihood calculation is made, using $M = 50000$ samples from the spectral density $f_{\boldsymbol{\beta}(i)}(\omega)$. After the chain is finished, the accepted $\boldsymbol{\beta}$ values are samples from the posterior distribution of

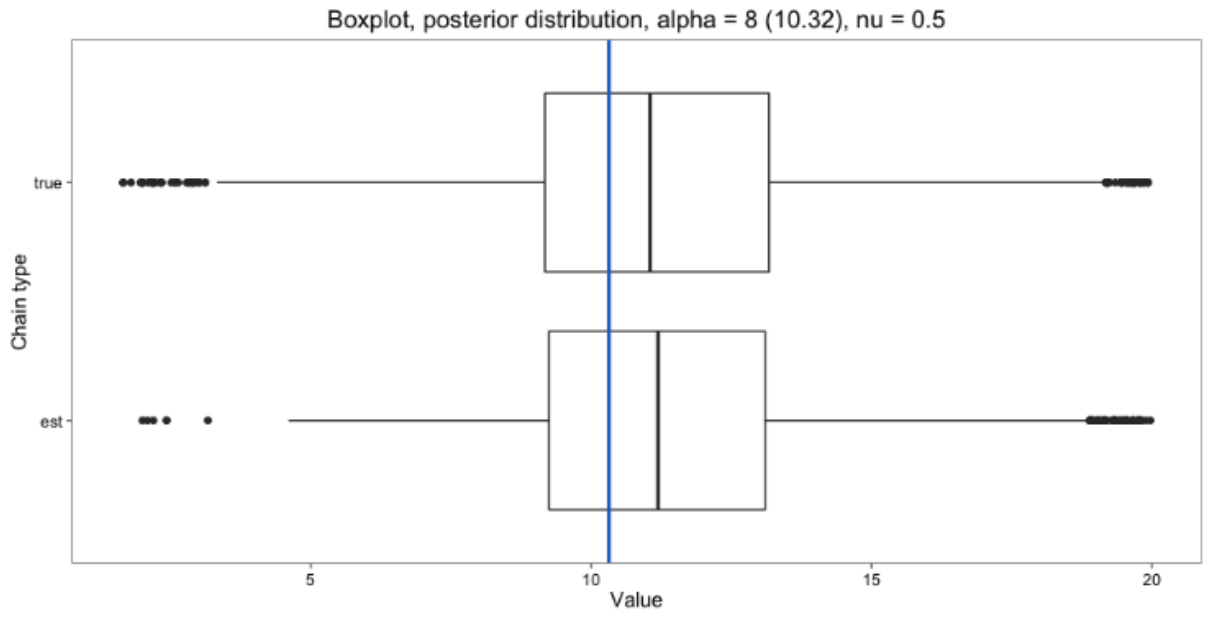

Figure 7: Posterior distributions of $\rho|\boldsymbol{y}$ for the true likelihood function and our estimated likelihood. This plot shows α, which occurs in another parameterization of the Matérn covariance function and is simply $1/\rho$. The observations were simulated with a value of $\alpha = 8$, but the maximum likelihood estimate given the data was $\hat{\alpha}_{MLE} = 10.32$.

$\boldsymbol{\beta} \mid \boldsymbol{y}$. After throwing away the first 2000 values in the chain as burnin, the posterior mean $\widehat{\boldsymbol{\beta}}$ is chosen to be the best estimate of the true $\boldsymbol{\beta}$.

We chose a normal random walk proposal distribution for both the $\boldsymbol{\beta}$ and τ^2 updates. Because it is symmetric, we were able to omit the proposal density from the calculation of the ratio a in Algorithm 2. We also used adaptive tuning to adjust the hyperparameters so that the acceptance rates stayed within an acceptable range.

Because we know the true $f(\omega)$, we can compare the spectral density approximated with $\widehat{\boldsymbol{\beta}}$ with the actual one. See Figure 9 for that comparison. Figure 10 shows that the estimated covariance function agrees with the actual covariance function.

The overall simulation study was set up as follows: first we simulate 100 mean-zero Gaussian processes with Matern covariance functions ($\nu = 1.5, \rho = 0.1548, \sigma = 1$) and 100 mean-zero Gaussian processes with damped cosine covariance functions ($\lambda = 1, \tau = 1$) with 400 observations randomly placed over the unit square, as above. In addition, for every dataset, observations were generated over a 50×50 uniform grid. These were held out for prediction.

We compare three techniques. The first is the "spline method" from Section 2. For the second, we fit a univariate Gaussian spatial regression model using the `spBayes` R package [3] [4]. This package's `spLM` function fits the model

$$y(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})'\boldsymbol{\beta} + w(\boldsymbol{s}) + \varepsilon(\boldsymbol{s})$$

using a Metropolis algorithm. See the package vignette for implementation details. In our case, we have no predictors $\boldsymbol{x}$, just the locations $\boldsymbol{s}$ themselves, so the model reduces to $y(\boldsymbol{s}) = w(\boldsymbol{s}) + \varepsilon(\boldsymbol{s})$, where w is the spatial process and ε is an independent white-noise process. We can specify that w should employ a Matérn covariance function, and `spLM` can estimate the parameter values from (2). In effect, this gives us the Matérn model that best fits the data.

The third technique is the one introduced by Hall et al. [7]. We find a kernel estimate of the covariogram using the kernel

$$K(x) = 0.9375(1 - x^2)^2$$

as the authors do in their original paper. We then numerically calculate the Fourier transform of this covariagram estimate, set all values beyond the first nonnegative one to zero to ensure positive definiteness, and Fourier transform back to the spatial domain.

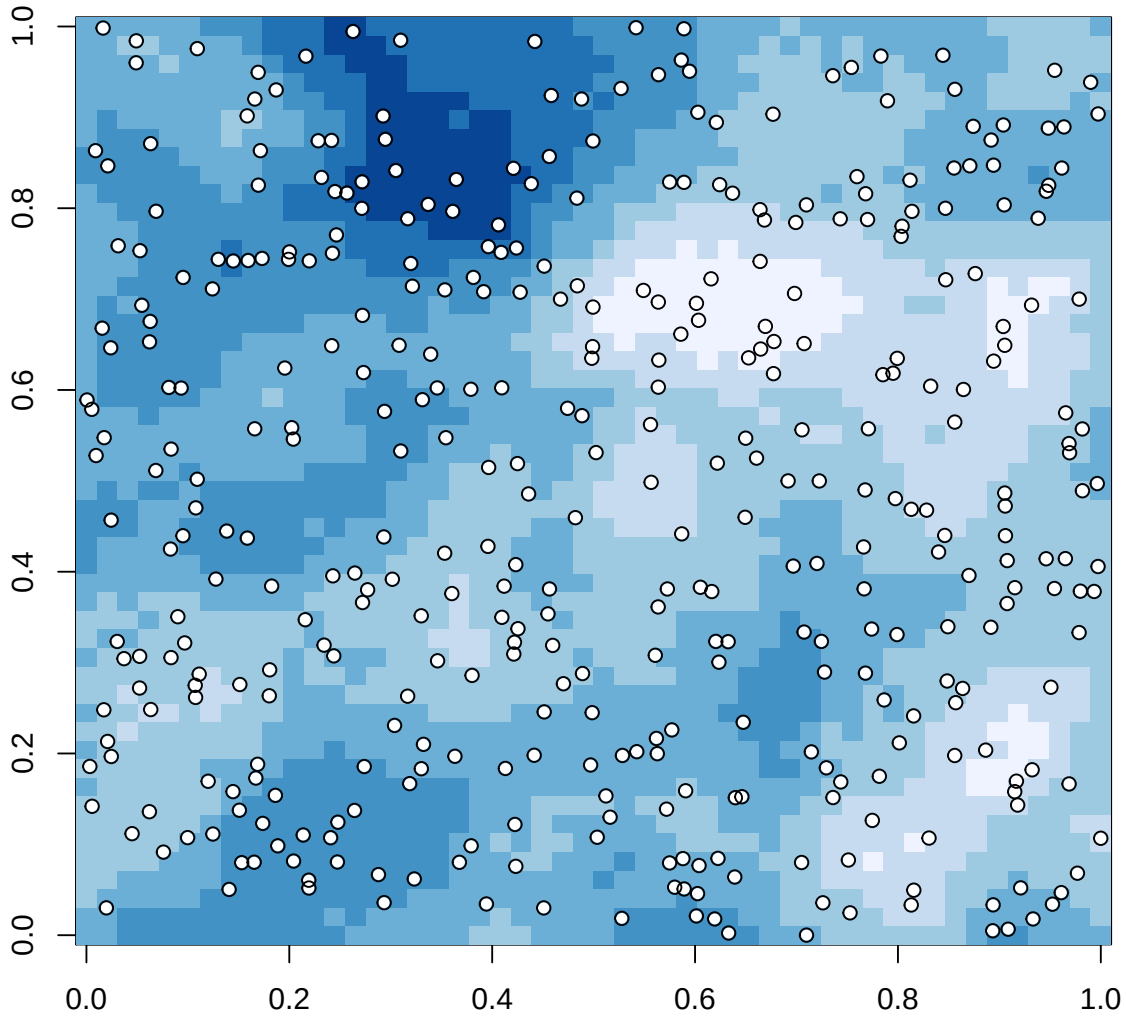

Figure 8: The locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{400} \in \mathbb{R}^2$ of the observations from a Gaussian process with $\nu = 1.5$, $\rho = 0.1548$, and $\sigma = 1$. The locations are random, but generated in such a way that ensures the minimum distance between any two points is greater than 0.005.

For each of these techniques, we estimate the covariance function using the 400 observations, generate predictions for the 2500 held out locations, and calculate the mean squared prediction error. We do this for every simulated dataset. Figure 11 shows the resulting covariance function estimates for every dataset, along with the true covariance functions. The Hall method underesti-

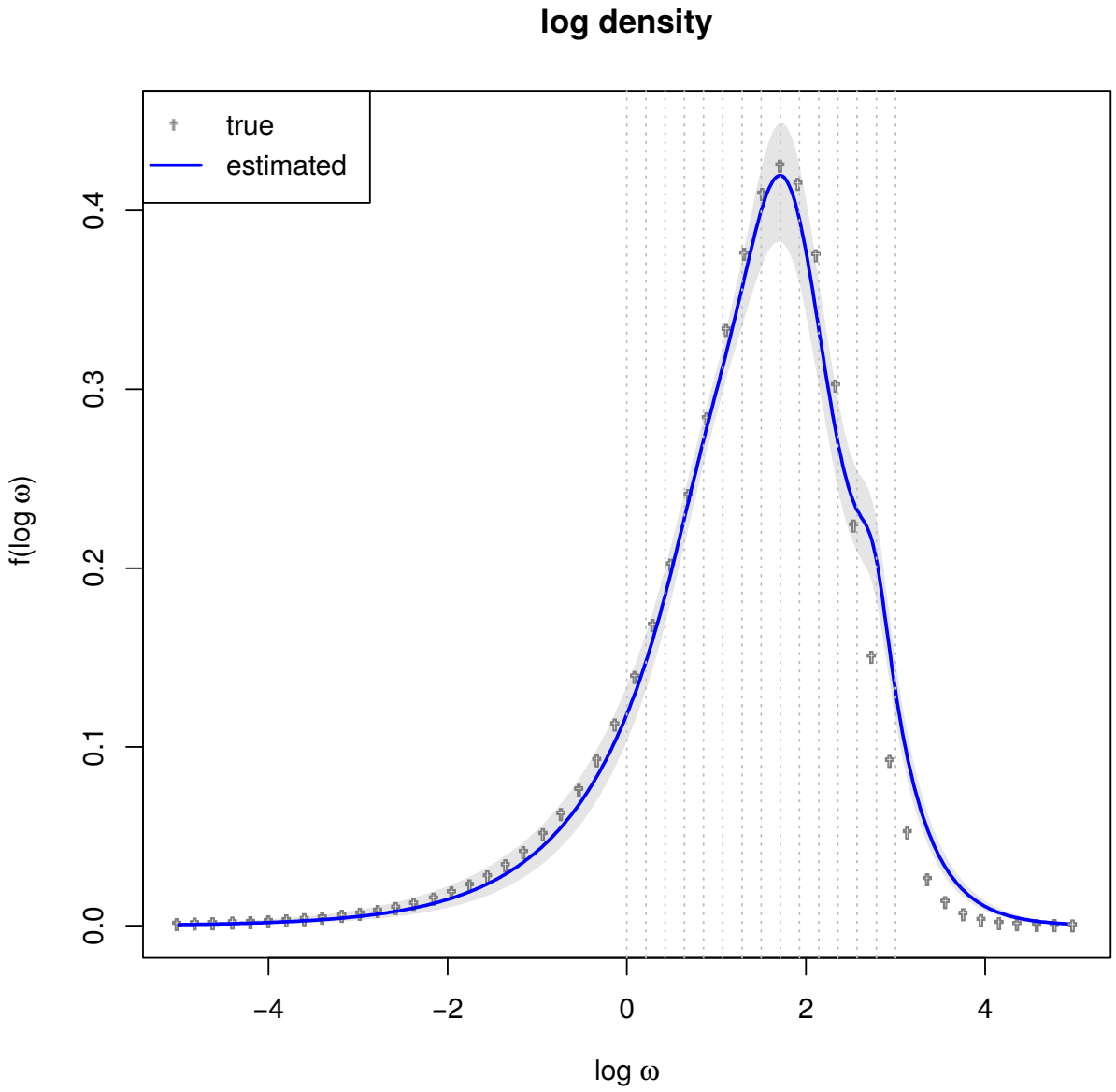

Figure 9: The actual $f(\log \omega)$ (grey dots) versus the estimated $\hat{f}(\omega)$ (blue). The shaded area represents the range of the middle 95% of $\hat{f}(\omega)$ for all draws from $\boldsymbol{\beta} \mid \boldsymbol{y}$. The vertical gray dotted lines indicate where the knots were placed.

mates the covariance at short distances in both the Matérn and damped cosine cases. Notice that the spline method is the only one to capture the hole effect in the damped cosine case, although it does tend to overestimate the magnitude of the effect.

Finally, we examine the integrated mean-square prediction error (IMSPE). For each dataset,

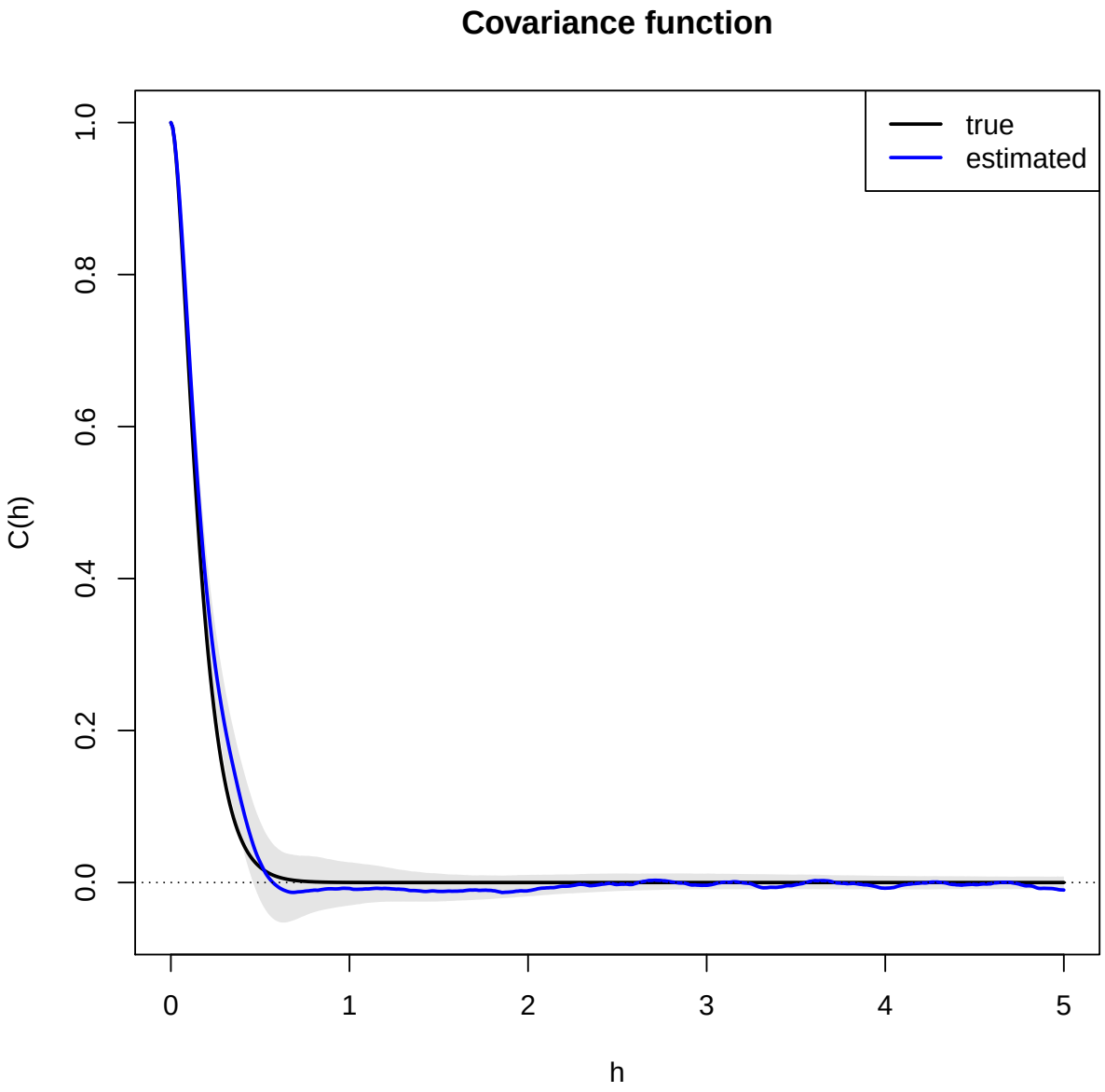

Figure 10: The actual $C(h)$ (black) versus the estimated $\widehat{C}(h)$ (blue). The shaded area represents the range of the middle 95% of $\widehat{C}(h)$ for all draws from $\boldsymbol{\beta} \mid \boldsymbol{y}$.

we calculate the *optimal* IMSPE by generating predictions at each of the 2500 prediction locations according to the true covariance function, and then comparing them to the corresponding observations. The boxplots in Figure 12 show the IMSPE relative to this optimal value for every dataset. The IMSPE is shown on the log scale to make the boxplots a bit more readable relative to one another.

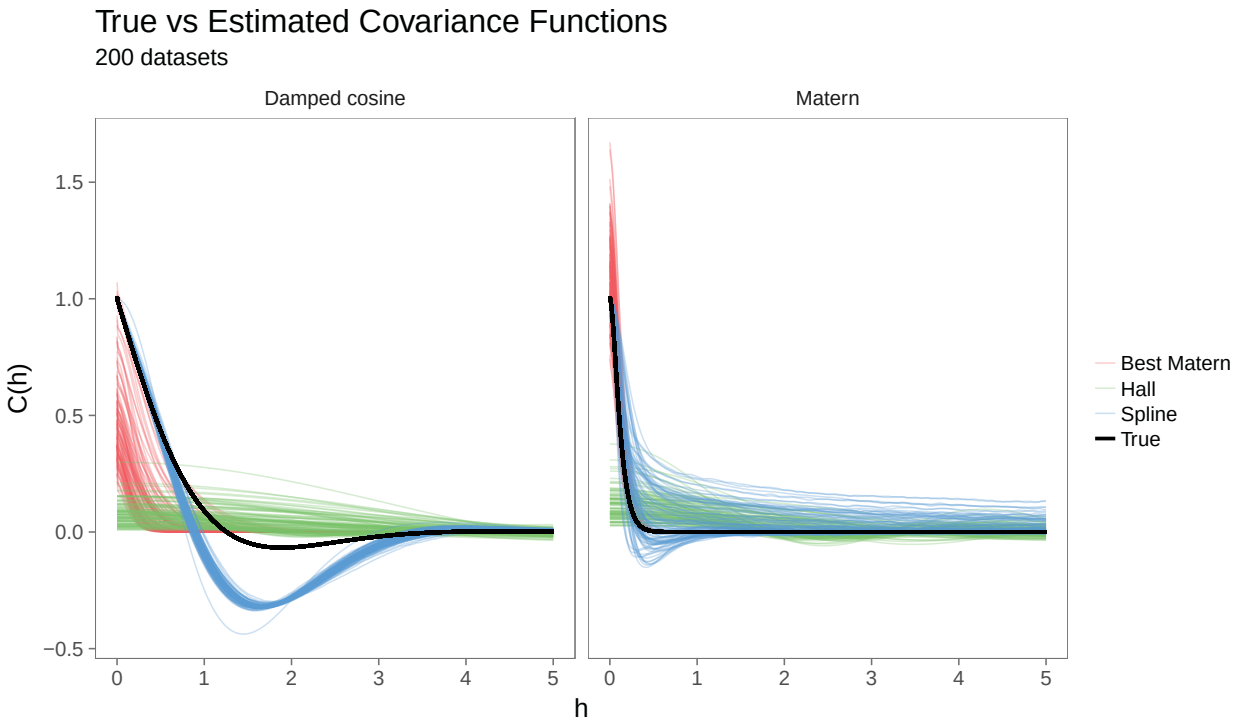

Figure 11: True covariance functions for the Matèrn and damped cosine cases, along with all of the estimates. The spline method estimates are in blue, the Hall method is in green, and the best-fitting Matèrn is in red.

Notice that both the spline method and the best-fitting Matérn model outperform the Hall method with regard to prediction, and are fairly close to each other. As expected, the best-fitting Matérn performs the best when the true process was also Matérn. Interestingly, it performs a bit better than the spline method when the true process was not Matérn as well. This further validates the argument in [13] that the Matérn model is an excellent choice in most practical situations when prediction accuracy is the goal. It does not pick up the hole effect present at moderate distances in the damped cosine model (see Figure 11), but interpolation is largely governed by the behavior at small distances, i.e. the smoothness of the process. The Matérn covariance function 2 has a parameter ν that controls this smoothness. Because we're estimating ν directly, we are able to precisely adjust it and get very good prediction performance.

There were a few Matérn datasets for which the spline method performed very poorly, as the long right tail of the boxplot indicates. This occurred when the Metropolis algorithm described in Algorithm 2 either mixed very poorly or started with poor initial $\boldsymbol{\beta}$ values. These issues could be corrected on any single dataset by manual intervention (e.g. by tuning the algorithm, running more iterations, etc.), but because the method was applied to all 100 datasets in bulk, we did not tune them individually. Note that these issues arose in only a few datasets, and the median relative IMSPE is still competitive with the best-fitting Matérn results.

3.3 Covariance Function Estimation Performance

We continue to evaluate these techniques by introducing another measure of performance. Broadly, we would like to know how closely the estimated covariance function aligns with the true covariance function. Let $C(h)$ be the true covariance and $\widehat{C}_m(h)$ be the estimate from method m. Define the function

$$g_m(h) = \big(C(h) - \widehat{C}_m(h)\big)^2.$$

Then the integrated mean squared error with respect to method m is

$$IMSE_m = \int_0^\infty g_m(h)\, dh.$$

To approximate this integral, we evaluate $g_m(h)$ on a uniform grid $\{h_0, \ldots, h_N\}$, $0 \le h_i \le 5$, and use the trapezoidal rule:

$$\widehat{IMSE}_m = \sum_{i=1}^{N} \frac{g_m(h_{i-1}) + g_m(h_i)}{2}(h_i - h_{i-1}).$$

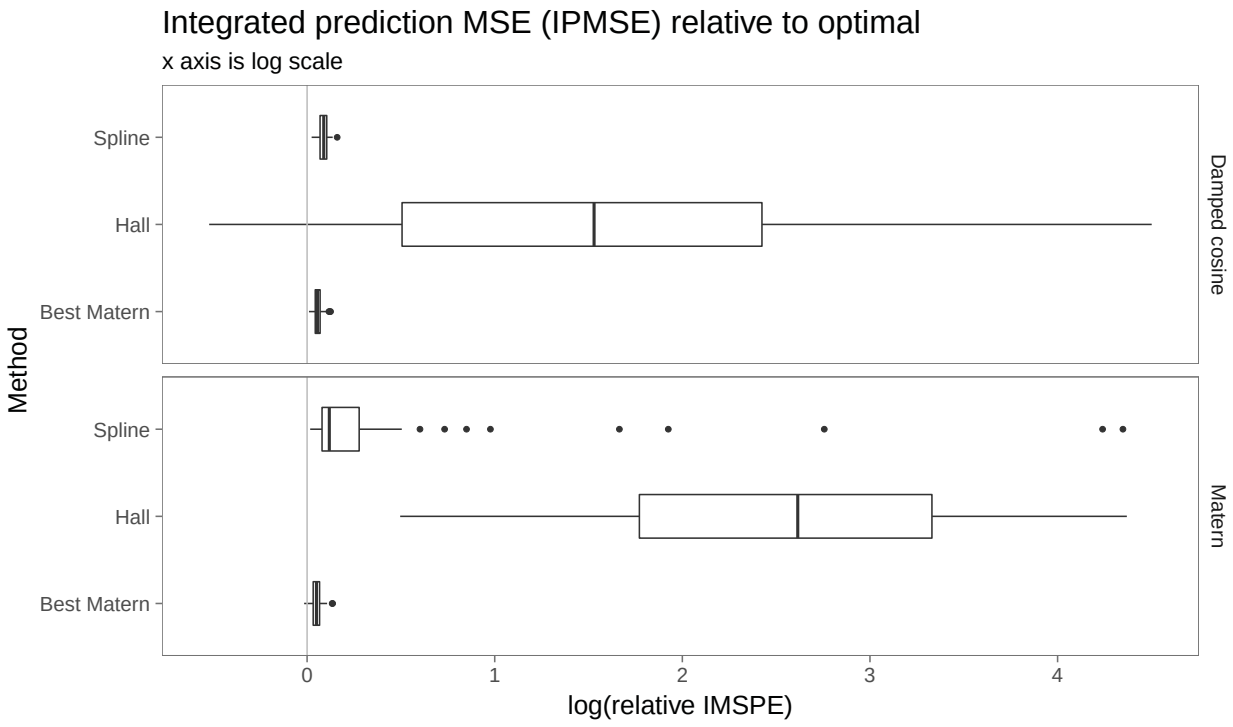

Figure 12: Mean squared prediction error relative to the optimal. The x axis is on the log scale, so the optimal MSPE would fall at zero.

In Figure 13, the functions $g_m(h)$ are shown for all methods and both covariance function types.

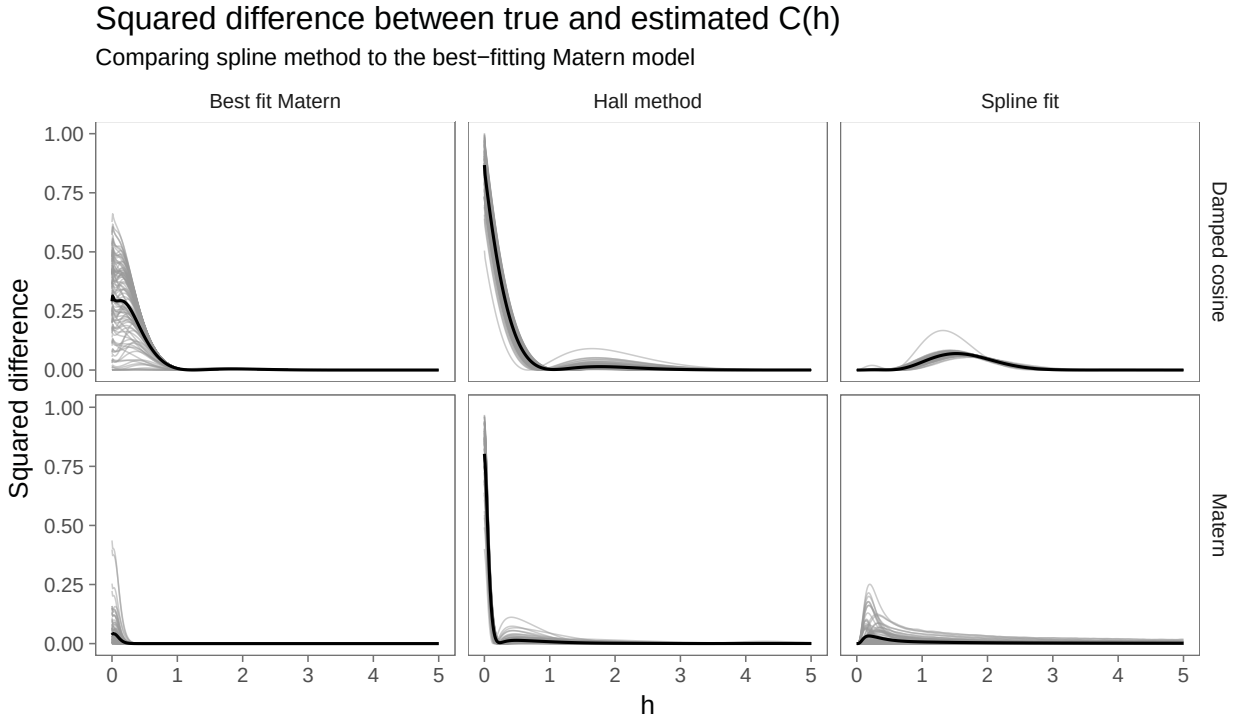

Figure 13: Squared discrepancy functions $g_m(h)$ for every method, both Matérn and damped cosine covariance models. The result from each simulated dataset is shown in gray, and the thicker black lines are the means of each facet of the plot.

There are a number of interesting features of this figure. First of all, the squared difference functions of the best-fit Matérn model and the spline method are, for the most part, quite small in magnitude. This further confirms what we observed in Figure 11 that these two methods fit the true Matérn covariance function very well. The Hall method does not perform as well for small h, although it does recover and match the true function once h is greater than 2. The best-fit Matérn tends to be slightly inaccurate for very small h, but its errors are smaller and it recovers even more quickly than the Hall method. On the other hand, the spline method appears to be at its most inaccurate not at $h = 0$, but around $0.25 < h < 1$.

Not surprisingly, the best-fit Matérn does not match the true covariance as well when the truth is not actually a Matérn covariance function. The spline method is closer to the truth, but it does tend to consistently overestimate the magnitude of the hole effect at $1 < h < 2$. The Hall method is the least accurate of the three on these data.

Figure 14 shows the IMSE (the integrals of the curves in Figure 11) for all datasets. Again, the best-fit Matérn and the spline method performed comparably on the Matérn datasets, with the Hall method doing worse. For the damped cosine datasets, the spline method performed better on average than the best-fit Matérn.

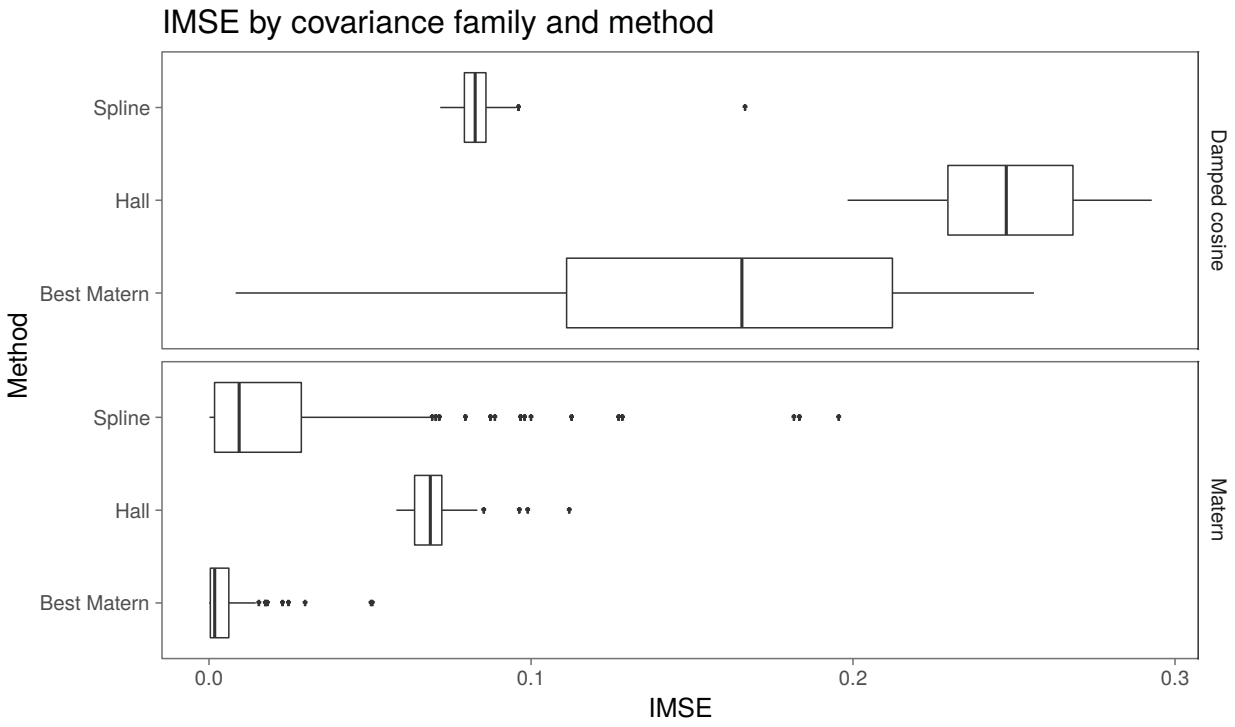

Figure 14: Boxplots of IMSE by method.

4 Data Application

To evaluate the performance of our method on real data, we use an atomic force microscopy (AFM) image of a solar cell. The image (see Figure 15) comes from a paper by Singh et al. [12], who wanted to improve efficiency of converting light into electricity by optimizing the concentration of carbon nanotubes in solar cells coated with a particular compound called poly(3-hexylthiophene): phenyl-C61-butyric acid methyl ester (P3HT:PCBM). This compound is spread in a microscopically thin film over the cell.

Figure 15 shows a P3HT:PCBM film doped with a 0.1% concentration of carbon nanotubes. By treating the film thickness as observations from an unknown isotropic stationary Gaussian

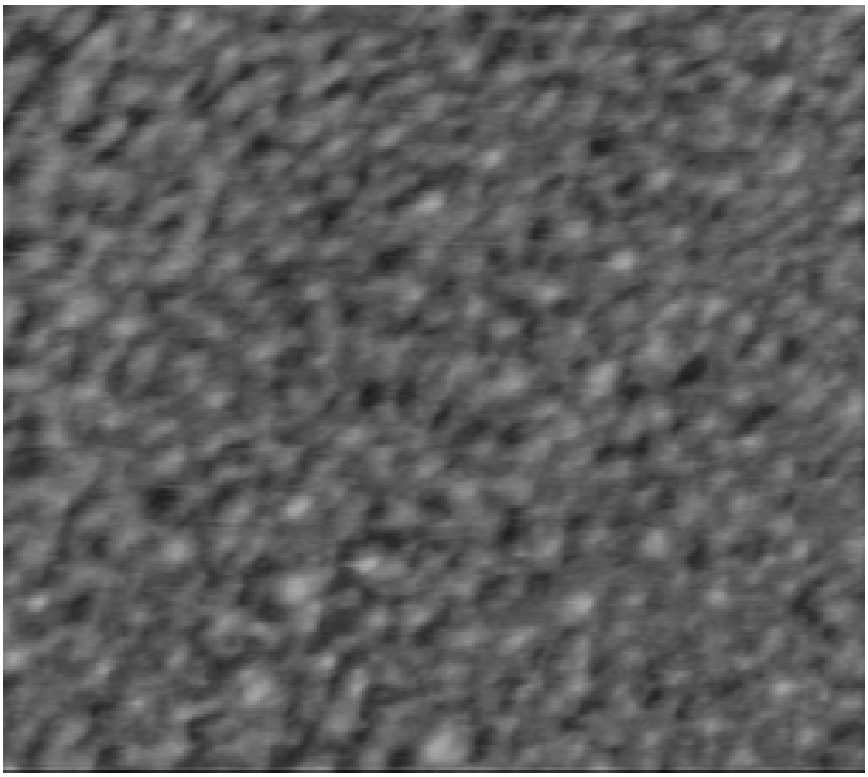

Figure 15: Atomic force microscopy image of a thin film of the P3HT:PCBM compound on a solar cell. Dark clusters represent areas where the film is thinner, and lighter clusters are thicker areas.

process, our goal was to train a covariance model on some of these observations and predict the others.

To prepare the data, we selected ten 25-by-25 pixel squares to use to estimate the covariance function. We treated the squares as 10 independent observations of a Gaussian process on the 25-by-25 grid. Once we obtained our estimate, we selected an out of sample 25-by-25 square, removed a 4-by-4 section of that square made predictions conditionally on the remainder as observations in the square. One of the training squares is shown in Figure 16, and Figure 17 shows the observation square, with the withheld prediction points highlighted in red.

We fit the semiparametric model and a Matérn model, and compare the prediction performance of each. The mean squared prediction error for the Matérn model was 0.0008707, while the mean squared prediction error for the semiparametric model was 0.0006414. The semiparametric model performed 26.3% better on the 16 prediction points, indicating that it is capturing the local behavior of the Gaussian process more effectively than the best-fit Matérn model can. Figure 18 shows the estimated covariance function from the semiparametric model. There is a significant hole effect that a Matérn model is not capable of capturing.

5 Conclusions and Future Work

This paper presents a method for estimating the covariance function of a stationary isotropic Gaussian process that does not rely on choosing a parametric family that may or may not fit the data well. It allows the data alone to drive the estimation, yet still guarantees that the estimated covariance function will turn out to be positive definite.

It is generally believed [13] that fitting a Matérn covariance function, via maximum likelihood or something similar, will give good interpolation performance even if the covariance of the true underlying Gaussian process can't be approximated very closely by a function from the Matérn family. Our simulations in Section 3 support that claim. However, our method has performed comparably with regard to prediction (Figure 12). In addition, we observed that our method can more closely match the covariance function (Figure 14) in cases where a "hole effect", or distances that correspond to a negative covariance, exists. One example of this phenomenon is the damped cosine covariance function (3).

Our method involves estimating the covariance matrix element by element, which is an ex-

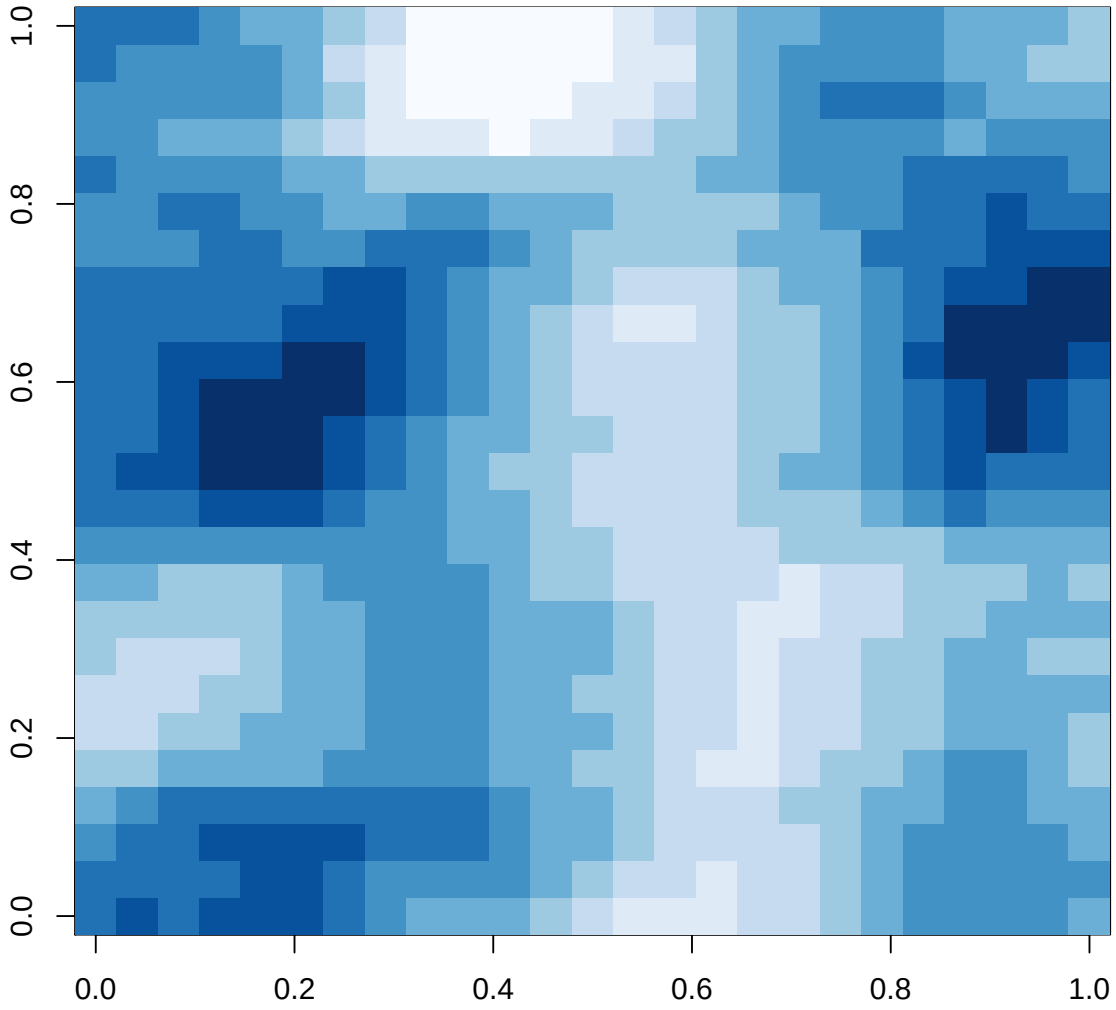

Figure 16: One of the 10 25-by-25 sections of the film image used to fit the spline model.

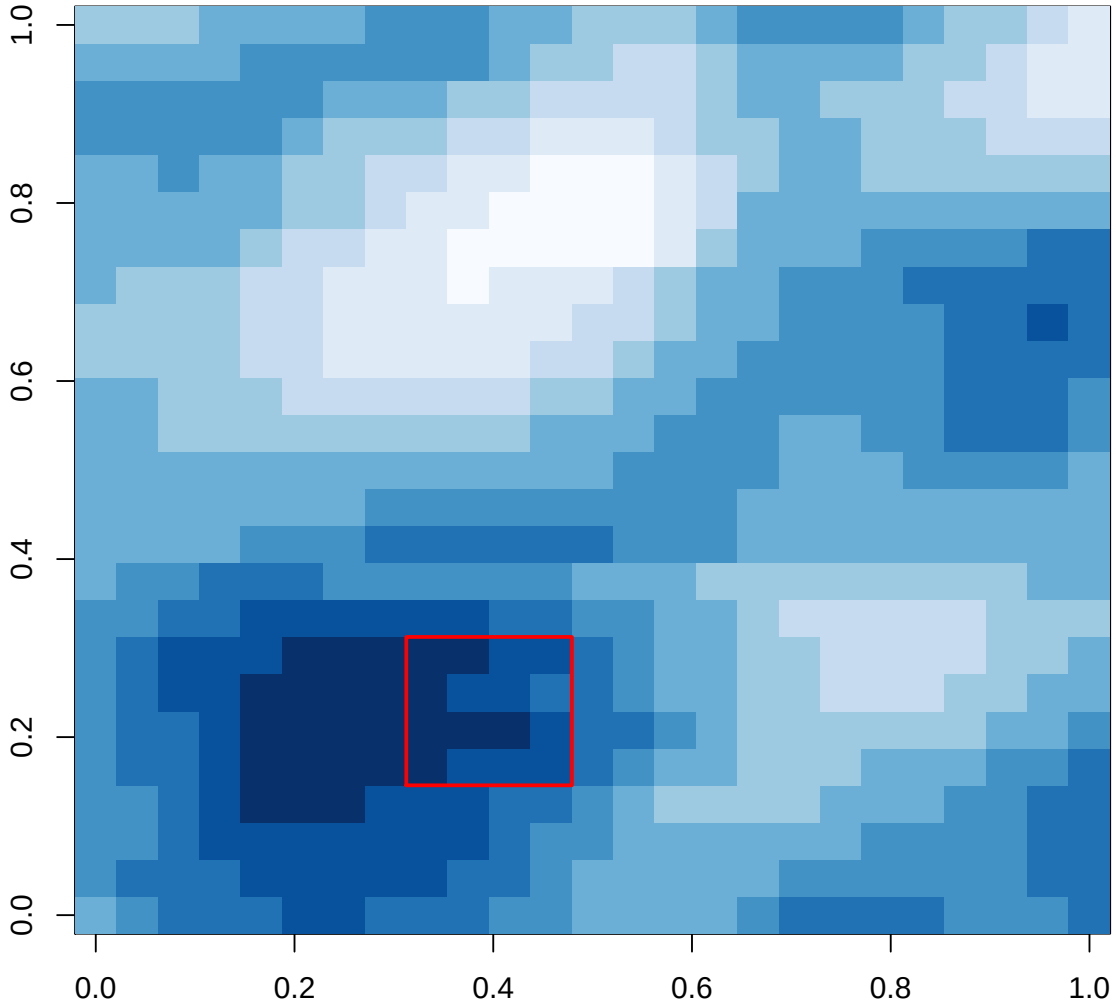

Figure 17: This 25-by-25 section of the film image was held out from the training step. The 16 outlined pixels are used for prediction, and the others are treated as observations.

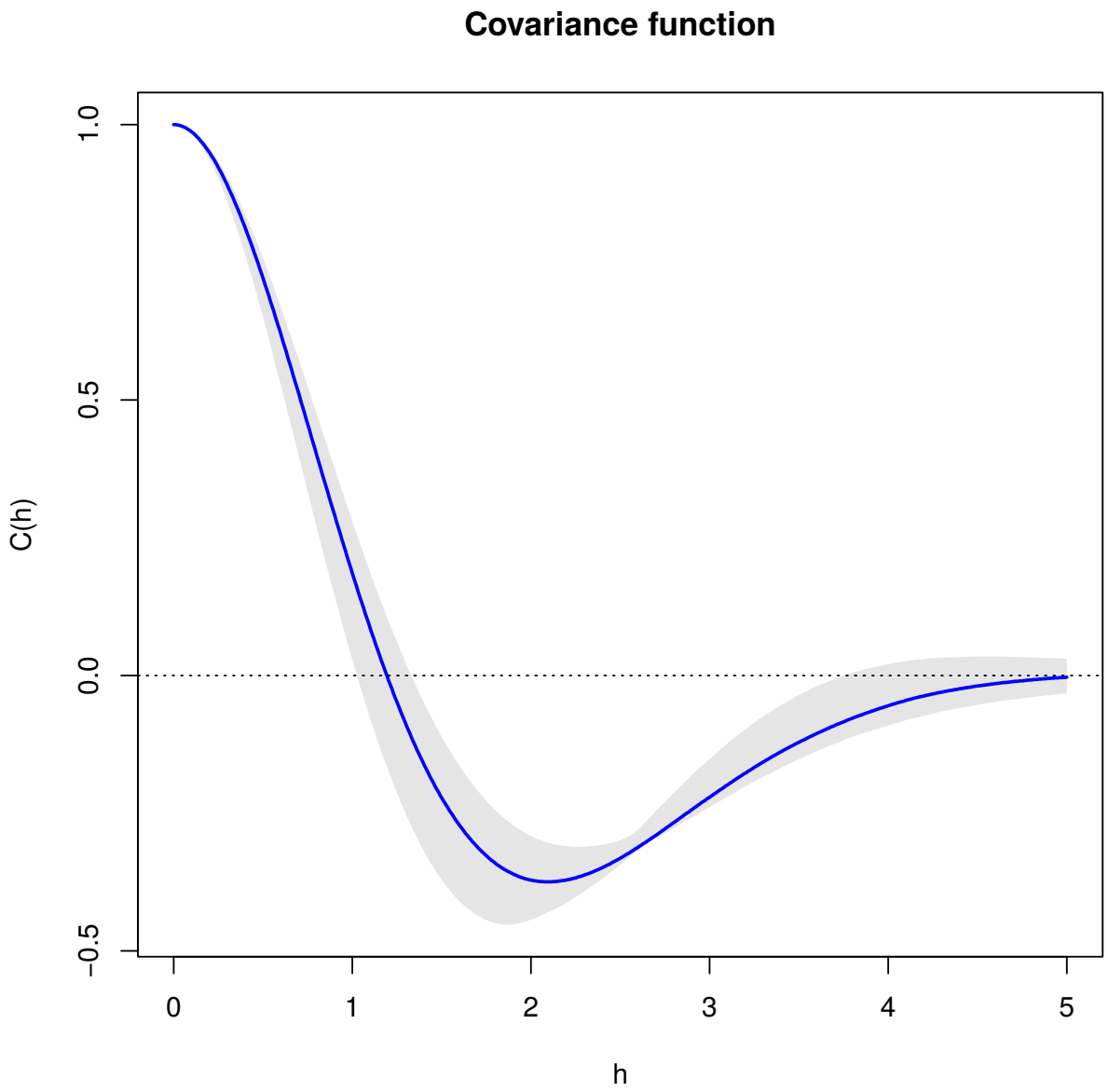

Figure 18: The estimated covariance function based on the 10 25-by-25 sections of the film image used to fit the spline model.

tremely computationally intensive problem, and one that would scale quite poorly if it were executed sequentially. However, this issue is largely alleviated through the use of parallel computing on a GPU. For a moderately sized problem (here $n = 400$), the GPU approach results in a speedup of roughly $100\times$.

This semiparametric covariance estimation method could be adapted to allow us to relax the isotropy and stationarity assumptions. Modeling an anisotropic Gaussian process would be a straightforward extension; it would involve replacing the simplification in (5) with a two-dimensional Fourier transform

$$C(\boldsymbol{h}) = \iint_{-\infty}^{\infty} \exp(i\boldsymbol{h}^T\boldsymbol{\omega})\ f(\boldsymbol{\omega})\ d\boldsymbol{\omega}.$$

It is possible to relax the stationarity assumption as well. In the literature there are several methods for dealing with spectral densities of nonstationary Gaussian processes. One notable example is the work of Fuentes [5], who proposed a nonstationary periodogram that is a nonparametric estimator of the spectral density $f(\omega_1, \omega_2)$.

SUPPLEMENTAL MATERIALS

Lemma 1. *Let f be a real-valued function that is continuous on $[a, b]$, differentiable on (a, b), and concave over (a, b). Then*

$$f(x) - f(\xi) \leq f'(\xi)(x - \xi) \qquad \forall \xi \in (a, b).$$

Proof. Case 1. Suppose $x > \xi$. By the mean value theorem, there exists some $\eta \in (\xi, x)$ such that

$$f'(\eta) = \frac{f(x) - f(\xi)}{x - \xi}.$$

Since f is concave, f' is a decreasing function. Therefore $f'(\eta) \leq f'(\xi)$, and so

$$f(x) - f(\xi) \leq f'(\xi)(x - \xi).$$

Case 2. Suppose $x < \xi$. By the mean value theorem, there exists some $\eta \in (x, \xi)$ such that

$$f'(\eta) = \frac{f(\xi) - f(x)}{\xi - x}.$$

Since f is concave, f' is a decreasing function. Therefore $f'(\eta) \geq f'(\xi)$, and so

$$f(\xi) - f(x) \geq f'(\xi)(\xi - x)$$
$$f(x) - f(\xi) \leq f'(\xi)(x - \xi).$$

□

Lemma 2. *For all $x \in (0, \infty)$,*

$$\log x \leq x - 1.$$

Proof. $f(x) = \log x$ is a concave function over the interval $(0, \infty)$. Use Lemma 1 with $\xi = 1$.

$$\log x - \log 1 \leq \frac{d}{dx}(\log x)\Big|_{x=1}(x - 1)$$
$$\log x - 0 \leq 1(x - 1)$$
$$\log x \leq x - 1.$$

□

Let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$ be a random vector consisting of observations from a stationary, isotropic Gaussian process with mean 0 and covariance function C. The observations are located at $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathcal{D} \subseteq \mathbb{R}^d$ in some bounded domain $\mathcal{D}$. Then the distribution of $\boldsymbol{Y}$ is

$$\boldsymbol{Y} \sim \mathcal{N}_n(0, \boldsymbol{\Sigma})$$

where $\Sigma_{ij} = C(||\boldsymbol{s}_i - \boldsymbol{s}_j||) = C(h_{ij})$. The likelihood function is

$$\ell(\boldsymbol{\Sigma}; \boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}) - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y}.$$

Theorem 2. *Let $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ be a vector of observations from a mean-zero stationary, isotropic Gaussian process with covariance function $C(h)$, taken at locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathcal{D} \subseteq \mathbb{R}^d$ in some bounded domain $\mathcal{D}$. For some symmetric density $f(\omega)$, let $\ell(\boldsymbol{y})$ be the likelihood*

$$\ell(\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y},$$

where $\Sigma_{ij} = \int \cos(h_{ij}\omega) f(\omega) d\omega$, and $\hat{\ell}(\boldsymbol{y})$ be the Monte Carlo approximated likelihood

$$\hat{\ell}(\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\widehat{\boldsymbol{\Sigma}}|) - \frac{1}{2}\boldsymbol{y}^T\widehat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{y}$$

where $\widehat{\Sigma}_{ij} = \frac{1}{M}\sum_{m=1}^{M} \cos(\widetilde{\omega}_m h_{ij})$, and $\widehat{\omega}_1, \ldots, \widehat{\omega}_M$ are an iid sample from $f(\omega)$. Then as $M \to \infty$,

$$\hat{\ell}(\boldsymbol{y}) \to \ell(\boldsymbol{y}) \text{ a.s. } f_\omega.$$

Proof. We will show that

$$\left|\hat{\ell}(\boldsymbol{\beta};\boldsymbol{y}) - \ell(\boldsymbol{\Sigma};\boldsymbol{y})\right| = -\frac{1}{2}\left|\log\det(\widehat{\boldsymbol{\Sigma}}) - \log\det(\boldsymbol{\Sigma})\right| - \frac{1}{2}\boldsymbol{y}^T\left|\widehat{\boldsymbol{\Sigma}}^{-1} - \boldsymbol{\Sigma}^{-1}\right|\boldsymbol{y} \to 0.$$

The notation in the final term refers to

$$\left|\widehat{\boldsymbol{\Sigma}}^{-1} - \boldsymbol{\Sigma}^{-1}\right| = \begin{bmatrix} [2]\left|\widehat{\Sigma}_{11}^{-1} - \Sigma_{11}^{-1}\right| & \left|\widehat{\Sigma}_{12}^{-1} - \Sigma_{12}^{-1}\right| & \cdots \\ \left|\widehat{\Sigma}_{21}^{-1} - \Sigma_{21}^{-1}\right| & \left|\widehat{\Sigma}_{22}^{-1} - \Sigma_{22}^{-1}\right| & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

Consider the first term. Let $\lambda_1, \ldots, \lambda_n$ and $\hat{\lambda}_1, \ldots, \hat{\lambda}_n$ be the eigenvalues of $\boldsymbol{\Sigma}$ and $\widehat{\boldsymbol{\Sigma}}$ respectively.

$$\left|\log\det(\widehat{\boldsymbol{\Sigma}}) - \log\det(\boldsymbol{\Sigma})\right| = \left|\sum_{i=1}^{n}\log\hat{\lambda}_i - \sum_{i=1}^{n}\log\lambda_i\right| = \left|\sum_{i=1}^{n}\log\frac{\hat{\lambda}_i}{\lambda_i}\right|.$$

From Lemma 2,

$$\begin{aligned}
\sum_{i=1}^{n}\log\frac{\hat{\lambda}_i}{\lambda_i} &\leq \sum_{i=1}^{n}\left(\frac{\hat{\lambda}_i}{\lambda_i} - 1\right) \\
&= \sum_{i=1}^{n}\frac{\hat{\lambda}_i - \lambda_i}{\lambda_i} \\
&\leq \sum_{i=1}^{n}\frac{|\hat{\lambda}_i - \lambda_i|}{\lambda_i} \\
&\leq \sqrt{\sum_{i=1}^{n}(\hat{\lambda}_i - \lambda_i)^2}\sqrt{\sum_{i=1}^{n}\frac{1}{\lambda_i^2}}. \qquad \text{(Cauchy-Schwarz)}
\end{aligned}$$

Because for all i, j, $\widehat{\Sigma}_{ij} \xrightarrow[M\to\infty]{} \Sigma_{ij}$ uniformly a.s. f_ω, the Frobenius norm

$$||\widehat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left|\widehat{\Sigma}_{ij} - \Sigma_{ij}\right|^2} \xrightarrow[M\to\infty]{} 0.$$

Theorem 4.2.8 of [9] tells us that

$$\sup_{k\geq 0}|\hat{\lambda}_{k+1} - \lambda_{k+1}| \leq ||\widehat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}||.$$

Therefore

$$\begin{aligned}
\sup_{k\geq 0}|\hat{\lambda}_{k+1} - \lambda_{k+1}| &\to 0 \\
\sum_{i=1}^{n}(\hat{\lambda}_i - \lambda_i)^2 &\to 0 \\
\left|\log\det(\widehat{\boldsymbol{\Sigma}}) - \log\det(\boldsymbol{\Sigma})\right| &\to 0 \quad \text{a.s. } f_\omega.
\end{aligned}$$

Now consider the second term. We need to show that

$$\boldsymbol{y}^T\left|\widehat{\boldsymbol{\Sigma}}^{-1}-\boldsymbol{\Sigma}^{-1}\right|\boldsymbol{y}\xrightarrow[M\to\infty]{}0.$$

$$\begin{aligned}\left|\left|\boldsymbol{y}^T\left|\widehat{\boldsymbol{\Sigma}}^{-1}-\boldsymbol{\Sigma}^{-1}\right|\boldsymbol{y}\right|\right| &\leq ||\widehat{\boldsymbol{\Sigma}}^{-1}-\boldsymbol{\Sigma}^{-1}||\;||\boldsymbol{y}||^2\\ &\leq ||\widehat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}-\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}||\;||\boldsymbol{y}||^2\\ &\leq ||\widehat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\Sigma}-\widehat{\boldsymbol{\Sigma}})\boldsymbol{\Sigma}^{-1}||\;||\boldsymbol{y}||^2\\ &\leq ||\widehat{\boldsymbol{\Sigma}}^{-1}||\;||\boldsymbol{\Sigma}-\widehat{\boldsymbol{\Sigma}}||\;||\boldsymbol{\Sigma}^{-1}||\;||\boldsymbol{y}||^2.\end{aligned}$$

As previously shown, $||\boldsymbol{\Sigma}-\widehat{\boldsymbol{\Sigma}}||\xrightarrow[M\to\infty]{}0$, so

$$\boldsymbol{y}^T\left|\widehat{\boldsymbol{\Sigma}}^{-1}-\boldsymbol{\Sigma}^{-1}\right|\boldsymbol{y}\xrightarrow[M\to\infty]{}0\quad\text{a.s. } f_\omega$$

as desired.

□

Title: Brief description. (file type)

R-package for MYNEW routine: R-package MYNEW containing code to perform the diagnostic methods described in the article. The package also contains all datasets used as examples in the article. (GNU zipped tar file)

HIV data set: Data set used in the illustration of MYNEW method in Section 3.2. (.txt file)

References

[1] Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2014). *Hierarchical modeling and analysis for spatial data*. Crc Press.

[2] Bochner, S. (1955). Harmonic analysis and probability theory. *Berkeley and Los Angeles*.

[3] Finley, A. O., Banerjee, S., and Carlin, B. P. (2007). spBayes: An R package for univariate and multivariate hierarchical point-referenced spatial models. *Journal of Statistical Software*, **19**(4), 1–24.

[4] Finley, A. O., Banerjee, S., and E.Gelfand, A. (2015). spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, **63**(13), 1–28.

[5] Fuentes, M. (2002). Spectral methods for nonstationary spatial processes. *Biometrika*, **89**(1), 197–210.

[6] Gelfand, A. E., Diggle, P. J., Fuentes, M., and Guttorp, P. (2010). *Handbook of Spatial Statistics*. CRC press.

[7] Hall, P., Fisher, N. I., and Hoffmann, B. (1994). On the Nonparametric Estimation of Covariance Functions. *The Annals of Statistics*, **22**(4), 2115–2134.

[8] Handcock, M. S. and Wallis, J. R. (1994). An approach to statistical spatial-temporal modeling of meteorological fields. *Journal of the American Statistical Association*, **89**(426), 368–378.

[9] Hsing, T. and Eubank, R. (2015). *Theoretical foundations of functional data analysis, with an introduction to linear operators*. John Wiley & Sons.

[10] Im, H. K., Stein, M. L., and Zhu, Z. (????). Semiparametric Estimation of Spectral Density With Irregular Observations. *Journal of the American Statistical Association*, (478), 726–735.

[11] Lang, S. and Brezger, A. (2004). Bayesian p-splines. *Journal of computational and graphical statistics*, **13**(1), 183–212.

[12] Singh, V., Arora, S., Arora, M., Sharma, V., and Tandon, R. P. (2014). Optimizing P3HT/PCBM/MWCNT films for increased stability in polymer bulk heterojunction solar cells. *Physics Letters, Section A: General, Atomic and Solid State Physics*, **378**(41), 3046–3054.

[13] Stein, M. L. (1999). *Interpolation of Spatial Data*.

[14] Suchard, M. A., Wang, Q., Chan, C., Frelinger, J., Cron, A., and West, M. (2010). Understanding gpu programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of computational and graphical statistics*, **19**(2), 419–438.

[15] Ver Hoef, J. M. and Cressie, N. (1993). Multivariable spatial prediction. *Mathematical Geology*, **25**(2), 219–240.

[16] Ye, J., Lazar, N. A., and Li, Y. (2015). Nonparametric variogram modeling with hole effect structure in analyzing the spatial characteristics of fMRI data. *Journal of Neuroscience Methods*, **240**, 101–115.