Journal of Machine Learning Research 17 (2016) 1-29 Submitted 6/15; Revised 8/16; Published 9/16

# Efficient Computation of Gaussian Process Regression for Large Spatial Data Sets by Patching Local Gaussian Processes

**Chiwoo Park** CPARK5@FSU.EDU
*Department of Industrial and Manufacturing Engineering*
*Florida State University*
*2525 Pottsdamer St., Tallahassee, FL 32310-6046, USA*

**Jianhua Z. Huang** JIANHUA@STAT.TAMU.EDU
*Department of Statistics*
*Texas A&M University*
*3143 TAMU, College Station, TX 77843-3143, USA*

**Editor:** Manfred Opper

## Abstract

This paper develops an efficient computational method for solving a Gaussian process (GP) regression for large spatial data sets using a collection of suitably defined local GP regressions. The conventional local GP approach first partitions a domain into multiple non-overlapping local regions, and then fits an independent GP regression for each local region using the training data belonging to the region. Two key issues with the local GP are (1) the prediction around the boundary of a local region is not as accurate as the prediction at interior of the local region, and (2) two local GP regressions for two neighboring local regions produce different predictions at the boundary of the two regions, creating undesirable discontinuity in the prediction. We address these issues by constraining the predictions of local GP regressions sharing a common boundary to satisfy the same boundary constraints, which in turn are estimated by the data. The boundary constrained local GP regressions are solved by a finite element method. Our approach shows competitive performance when compared with several state-of-the-art methods using two synthetic data sets and three real data sets.

**Keywords:** constrained Gaussian process regression, kriging, local regression, boundary value problem, spatial prediction, variational problem

## 1. Introduction

Within its origin in geostatistics and known as kriging, the Gaussian process regression (hereafter abbreviated as GP regression) has been developed to be a useful tool in machine learning (Rasmussen and Williams, 2005). It provides the best linear unbiased prediction computable by a simple closed-form expression, which also has a nice probabilistic interpretation (MacKay, 1998). However, computing the exact solution of a GP regression requires $O(N^3)$ operations when the number of data points is $N$, which is more than 10,000 or 100,000 in a typical geospatial data set. Such a computational complexity is prohibitively

high for data sets of large size. The purpose of this paper is to develop a new computational method to expedite the computation of GP regression for large data sets.

The computation issue for GP regression has received much attention in machine learning and spatial statistics. Since a major computation bottleneck for a GP regression is the inversion of a big sample covariance matrix of size $N \times N$, many approaches proposed to approximate the sample covariance matrix with a more easily invertible one. Covariance tapering (Furrer et al., 2006; Kaufman et al., 2008) tapers the original covariance function to make a sample covariance matrix sparser and applies the sparse matrix computation algorithms for faster inversion of the matrix. Low-rank approximation (Seeger et al., 2003; Snelson and Ghahramani, 2006; Cressie and Johannesson, 2008; Banerjee et al., 2008; Sang and Huang, 2012) introduces $M$ latent variables and assumes a certain conditional independence given the latent variables, which reduces the rank of the resulting sample covariance matrix to $M$. The approximation of a Gaussian random field by a Gaussian Markov random field has also been proposed (Lindgren et al., 2011). When a covariance matrix of the approximated Gaussian random field is a Matérn covariance function, the sparse precision matrix for the Gaussian Markov random field can be explicitly constructed, and the approximation can be efficiently computed.

On the other hand, local GP regression partitions a regression domain into local regions, and an independent GP regression model is learned for each local region. Since the number of observations belonging to a local region is much smaller than the total number of observations, the resulting sample covariance matrix for a local GP regression becomes much smaller. However, because of the independence of the local GP regressions, two local GP regressions for two neighboring local regions produce different predictions at the boundary of the two regions. This discontinuity in prediction is not acceptable in applications. Many proposed methods have combined local GP regressions into a global model. A popular approach is to take a mixture of local GP regressions through a Dirichlet mixture (Rasmussen and Ghahramani, 2002), a treed mixture (Gramacy and Lee, 2008), Bayesian model averaging (Tresp, 2000; Chen and Ren, 2009; Deisenroth and Ng, 2015), or a locally weighted projection (Nguyen-Tuong et al., 2009). Another approach is to use multiple additive covariance functions of a global covariance and a local covariance (Snelson and Ghahramani, 2007; Vanhatalo and Vehtari, 2008), to simply construct a new local model for each testing location (Gramacy and Apley, 2015).

Domain decomposition method (DDM, Park et al., 2011) is a specific local GP regression method that attempts to constrain the prediction of local GP regressions to be equal at their shared domain boundaries. DDM was shown in the original paper to numerically outperform several existing local GP methods in terms of computational cost and prediction accuracy. Our proposed approach in this paper follows and advances DDM in several aspects. In particular, we improve the way of constraining the prediction at boundaries of local regions. In the original DDM paper, the predictions of two local GP regressions for two neighboring local regions were constrained to be equal only at a finite number of locations on the boundary of the two regions, so there is no guarantee that the predictions are the same at other boundary locations. Our new approach considers a variational formulation for a collection of boundary constrained local GP regressions to ensure that the predictions of the two local GP regressions for neighboring regions are the same for all points on the shared boundary. The variational formulation allows us to solve the boundary

constrained local GP regressions using the finite element method. This is mathematically more elegant and conceptually simpler than the previously somewhat ad hoc treatment. In addition, we significantly improve the DDM by proposing two approaches for estimating the boundary constraints (i.e., boundary values of local regions). The improved accuracy of estimating these constraints leads to better prediction accuracy of our constrained local GP regressions. Last, our new approach has better numerical stability than the DDM. It was previously reported that the predictive variance estimate of the DDM can be negative for some numerical examples (Pourhabib et al., 2014). Since the expression of the DDM's predictive variance estimate cannot be theoretically negative, the negative estimate is due to numerical issues. Our new approach provides positive predictive variances for all the examples. The computation speed of the new approach is comparable to the DDM. When the domain in $\mathbb{R}^d$ is partitioned into $S$ local regions and each local region has $N_S$ training data points, the computational cost of the proposed method is $O(NN_S^2 + dS)$, where $N_S \ll N$. Since our method can be viewed as "patching" a collection of local GP regressions, we refer to our method as *patched Gaussian Process regression* or *patched GP* for short.

The proposed patched GP method is theoretically applicable for an arbitrary input dimension $d$, but the implementation of the approach may be practically difficult for $d > 2$ mainly due to hardness in generating finite element meshes for high dimensions. Therefore, the practical application of the proposed approach would be a GP regression with a spatial data set of large volume (i.e., the data fall in a domain in $\mathbb{R}^2$), which finds broad applications in spatial statistics and remote sensing (Stein, 2012; Curran and Atkinson, 1998). We will still describe and derive our approach for a general dimension to ease a possible future extension of the approach to high dimensional problems. As a byproduct of this work, we develop a finite element method for the boundary constrained GP regression problem, which may have potential applications in GP solutions for partial differential equations (Graepel, 2003) or for linear stochastic differential equations having boundary conditions (Steinke and Schölkopf, 2008).

The rest of the paper is organized as follows. Section 2 reformulates the GP regression as an optimization problem and also provides an equivalent variational formulation. Section 3 considers the boundary constrained GP regression problem through optimization and develops a finite element method for solving the equivalent variational problem. Section 4 presents the core methodology of the proposed patched GP method for efficient computation of GP regressions, including a finite element method for solving a boundary constrained local GP regression, estimation of the boundary constraints, and estimation of the parameters of the Gaussian process. Section 5 shows the numerical performance of the patched GP method for different tuning parameters, and compares it to the full GP regression (i.e., full implementation of GP regression without approximation) and its precursor, the DDM, using both synthetic and real data sets. Section 6 provides additional numerical comparisons of the patched GP with several state-of-the-art approaches using real data sets. Finally Section 7 concludes the paper.

## 2. Reformulation of Gaussian Process Regression as An Optimization Problem

A GP regression is formulated as follows: given a training set $\mathcal{D} = \{(x_n, y_n), n = 1, \ldots, N\}$ of $N$ pairs of inputs $x_n$ and noisy outputs $y_n$ of a latent function $f$, obtain the predictive distribution of $f$ at a test location $x_*$, denoted by $f_* = f(x_*)$. We assume that the latent function comes from a zero-mean Gaussian process with a covariance function $k(\cdot, \cdot)$ and the noisy observations $y_i$ are given by

$$y_i = f(x_i) + \epsilon_i, \qquad i = 1, \ldots, N,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ are white noises independent of $f(x_i)$. Denote $\mathbf{x} = [x_1, x_2, \ldots, x_N]'$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]'$. The joint distribution of $(f_*, \boldsymbol{y})$ is

$$P(f_*, \boldsymbol{y}) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k_{**} & \boldsymbol{k}'_{\mathbf{x}*} \\ \boldsymbol{k}_{\mathbf{x}*} & \sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}} \end{bmatrix}\right),$$

where $k_{**} = k(x_*, x_*)$, $\boldsymbol{k}_{\mathbf{x}*} = (k(x_1, x_*), \ldots, k(x_N, x_*))'$ and $\boldsymbol{K}_{\mathbf{xx}}$ is an $N \times N$ matrix with $(i, j)^{th}$ entity $k(x_i, x_j)$. The subscripts of $k_{**}, \boldsymbol{k}_{\mathbf{x}*}$, and $\boldsymbol{K}_{\mathbf{xx}}$ represent two sets of locations between which the covariance is computed, and $x_*$ is abbreviated as $*$. The predictive distribution of $f_*$ given $\boldsymbol{y}$ is

$$P(f_*|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{k}'_{\mathbf{x}*}(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}'_{\mathbf{x}*}(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{k}_{\mathbf{x}*}). \tag{1}$$

The predictive mean $\boldsymbol{k}'_{\mathbf{x}*}(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{y}$ gives the point prediction of $f(x)$ at location $x_*$, whose uncertainty is measured by the predictive variance $k_{**} - \boldsymbol{k}'_{\mathbf{x}*}(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{k}_{\mathbf{x}*}$. Efficient calculation of the predictive mean and variance has been the focus of much research.

The predictive mean and variance can be derived using the viewpoint of the best linear unbiased predictor (BLUP) as follows. Consider all linear predictors

$$\mu(x_*) = \boldsymbol{u}(x_*)'\boldsymbol{y}, \tag{2}$$

which automatically satisfy the unbiasedness requirement $E[\mu(x_*)] = 0$ since all random variables $y_i$ have zero mean. We seek an $N$-dimensional vector $\boldsymbol{u}(x_*)$ such that the mean squared prediction error $E[\mu(x_*) - f(x_*)]^2$ is minimized. Since $E[\mu(x_*)] = 0$ and $E[f(x_*)] = 0$, the mean squared prediction error equals the error variance $\mathrm{var}[\mu(x_*) - f(x_*)]$ and can be expressed as

$$\begin{aligned} \sigma(x_*) &= \boldsymbol{u}(x_*)'E(\boldsymbol{y}\boldsymbol{y}')\boldsymbol{u}(x_*) - 2\boldsymbol{u}(x_*)'E(\boldsymbol{y}f_*) + E(f_*^2) \\ &= \boldsymbol{u}(x_*)'(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})\boldsymbol{u}(x_*) - 2\boldsymbol{u}(x_*)'\boldsymbol{k}_{\mathbf{x}*} + k_{**}, \end{aligned} \tag{3}$$

which is a quadratic form in $\boldsymbol{u}(x_*)$. It is easy to see $\sigma(x_*)$ is minimized if and only if $\boldsymbol{u}(x_*)$ is chosen to be $(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{k}_{\mathbf{x}*}$; moreover, the minimal value of $\sigma(x_*)$ equals the predictive variance given in (1).

The BLUP view of GP regression suggests a reformulation of GP regression as the following optimization problem:

$$\underset{\boldsymbol{u}(x_*)\in\mathbb{R}^N}{\text{Minimize}} \quad z[\boldsymbol{u}(x_*)] = \frac{1}{2}\boldsymbol{u}(x_*)'\boldsymbol{A}\boldsymbol{u}(x_*) - \boldsymbol{f}(x_*)'\boldsymbol{u}(x_*), \tag{4}$$

where $\boldsymbol{A} = (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})$ is an $N \times N$ positive definite matrix, and $\boldsymbol{f}(x_*) = \boldsymbol{k}_{\mathbf{x}*}$ is a $N \times 1$ vectorial function. Note that the objective function in (4) equals half of the error variance of a linear predictor given in (3) subtracting the constant term $k_{**}/2$. The one half factor is introduced here to make subsequent formulas neat. The solution of (4) is $\boldsymbol{u}^\dagger(x_*) = \boldsymbol{A}^{-1}\boldsymbol{k}_{\mathbf{x}*} = (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}\boldsymbol{k}_{\mathbf{x}*}$. Back to the GP regression problem, the predictive mean is given by $\boldsymbol{u}^\dagger(x_*)^t\boldsymbol{y}$ and the predictive variance is $2\, z[\boldsymbol{u}^\dagger(x_*)] + k_{**}$, twice the optimal objective value plus the variance of $f_*$ at the location $x_*$.

Usually we are interested in obtaining the predictive mean and variance at multiple locations in a domain $\Omega \subset \mathbb{R}^N$, where all training locations $x_i$'s also belong to. To consider the prediction at all locations in $\Omega$, we consider the following optimization problem:

$$\underset{\boldsymbol{u}(\cdot)\in[L^2(\Omega)]^N}{\text{Minimize}} \quad J(\boldsymbol{u}) = \int_\Omega \left\{ \frac{1}{2}\boldsymbol{u}(x)'\boldsymbol{A}\boldsymbol{u}(x) - \boldsymbol{f}(x)'\boldsymbol{u}(x) \right\} dx, \tag{5}$$

where $[L^2(\Omega)]^N$ is the Cartesian product of $N$ Hilbert spaces $L^2(\Omega)$. The objective function here is simply the integration of the objective function in the single location problem (4). Since the optimal solution $\boldsymbol{u}^\dagger(x_*)$ obtains the minimum objective value of (4) at every location $x_* \in \Omega$, $\boldsymbol{u}^\dagger(x_*)$ as a function of $x_*$ also solves the global problem (5).

According to a standard result from functional analysis (Ern and Guermond, 2004), the problem (5) has an equivalent variational formulation, as follows.

**Proposition 1** *The vector $\boldsymbol{u} \in [L^2(\Omega)]^N$ minimizes $J(\boldsymbol{u})$ if and only if it solves the integral equation*

$$\int_\Omega \boldsymbol{u}(x)'\boldsymbol{A}\boldsymbol{v}(x)\, dx = \int_\Omega \boldsymbol{f}(x)'\boldsymbol{v}(x)\, dx \qquad \text{for each } \boldsymbol{v} \in [L^2(\Omega)]^N. \tag{6}$$

The proof is given in Appendix A.

Throughout the rest of the paper, whenever no confusion may arise and to alleviate the notation, we omit the Lebesgue measure under the integral sign. For example, we shall write $\int_\Omega \boldsymbol{u}'\boldsymbol{A}\boldsymbol{v}$ and $\int_\Omega \boldsymbol{f}'\boldsymbol{v}$ instead of $\int_\Omega \boldsymbol{u}(x)'\boldsymbol{A}\boldsymbol{v}(x)\, dx$ and $\int_\Omega \boldsymbol{f}(x)'\boldsymbol{v}(x)\, dx$.

## 3. Gaussian Process Regression with Boundary Constraints

The optimization problem (5) was introduced in previous section as a reformulation of the GP regression. Now we introduce boundary constraints to the GP regression through this optimization problem and develop a finite element solution for the corresponding variational formulation. This finite element solution will serve as a building block in the next section for the patched GP regression, which consists of a collection of boundary constrained GP regressions.

### 3.1 Constrained Optimization Problem and Its Variational Formulation

We require that the domain $\Omega$ is a Lipschitz bounded open set. Let $\partial\Omega$ denote the boundary of the domain $\Omega$. We need to restrict our attention to smooth functions and specifically consider only solution vectors in the Sobolev space $[H^1(\Omega)]^N$ instead of $[L^2(\Omega)]^N$, where $H^1(\Omega) := \{u \in L^2(\Omega); \partial_i u \in L^2(\Omega), 1 \le i \le d\}$ with $\partial_i u$ denoting the partial derivative of $u$ with respect to the $i$th dimension of input. We cannot use the bigger $L^2(\Omega)$ space because

the value of a function in this space can be unbounded on $\partial\Omega$ and so not well-defined. For example, take $\Omega = (0,1)$ and $u(x) = x^{-1/3}$. It is clear that $u(x) \in L^2(\Omega)$ but $u(0) = \infty$. On the other hand, the values of a function in $H^1(\Omega)$ are bounded at its domain boundary. In fact, if $u \in H^1(\Omega)$, then $u$ restricted on $\partial\Omega$, which we denote by $u_{|\partial\Omega}$, is a member of $H^{1/2}(\partial\Omega) := \{u \in L^2(\Omega) : \frac{u(x)-u(y)}{||x-y||^{(d+1)/2}} \in L^2(\Omega \times \Omega)\}$ (Ern and Guermond, 2004, Theorem B.52).

Consider the boundary constraints of the form $\mathbf{y}'\boldsymbol{u}(x) = b(x)$ for some known function $b \in H^{1/2}(\partial\Omega)$. Since $\mathbf{y}'\boldsymbol{u}(x)$ can be interpreted as a linear predictor at location $x$, the constrains simply require the predictions at the domain boundary to have certain specified functional form. Denote

$$H_b = \left\{ \boldsymbol{u} \in [H^1(\Omega)]^N : \int_{\partial\Omega} \boldsymbol{y}'\boldsymbol{u}_{|\partial\Omega}\, v = \int_{\partial\Omega} b\, v \quad \text{for each } v \in H^{1/2}(\partial\Omega) \right\}.$$

A constrained version of the optimization problem (5) can be written as

$$\underset{\boldsymbol{u}\in H_b}{\text{Minimize}} \quad J(\boldsymbol{u}) = \int_\Omega \left\{ \frac{1}{2}\boldsymbol{u}(x)'\boldsymbol{A}\boldsymbol{u}(x) - \boldsymbol{f}(x)'\boldsymbol{u}(x) \right\} dx. \tag{7}$$

Here, for mathematical convenience, we have replaced the strict boundary constraints $\mathbf{y}'\boldsymbol{u}(x) = b(x)$ by a weaker form for $\boldsymbol{u} \in H_b$.

Similar to Proposition 1, we can show that the optimization problem (7) is equivalent to a variational formulation:

**Proposition 2** *The vector $\boldsymbol{u} \in H_b$ minimizes $J(\boldsymbol{u})$ if and only if it solves the integral equation*

$$\int_\Omega \boldsymbol{u}(x)'\boldsymbol{A}\boldsymbol{v}(x)\, dx = \int_\Omega \boldsymbol{f}(x)'\boldsymbol{v}(x)\, dx \qquad \textit{for each } \boldsymbol{v} \in H_b. \tag{8}$$

The proof is given in Appendix B.

### 3.2 Finite Element Approximation

A finite element method approximates the space $H_b$ of vector-valued functions on a domain $\Omega$ by a finite dimensional vector space. With the approximation, the integral equation (8) is converted to a finite-dimensional linear system of equations.

The finite dimensional approximation scheme requires a mesh and a set of finite elements. A mesh $\mathcal{K}_h = \{K_1, \ldots, K_M\}$ is a set of a finite number of compact, connected and Lipschitz subsets of $\Omega$ with non-empty interior which partitions $\Omega$, where each $K_m$ is called a mesh cell, and $h$ parameterizes the size of $K_m$; e.g. if $K_m$ is a polygon, $h$ is the length of the polygon's side. For each $K_m \in \mathcal{K}_h$, a finite element is defined as a triplet $\{K_m, P_m, \mathcal{A}_m\}$, where $P_m$ is a vector space of functions $q : K_m \to \mathbb{R}$ with $dim(P_m) = p$, and $\mathcal{A}_m$ is a set of $p$ linear forms $\boldsymbol{\alpha}_{mj} : P_m \to \mathbb{R}^N$ spanning the dual vector space of $P_m$, which is called the local degrees of freedom. There exists a basis $\{\phi_{m1}, \ldots, \phi_{mp}\}$ of the vector space $P_m$ satisfying $\boldsymbol{\alpha}_{mj}(\phi_{mk}) = \mathbf{1}_N\delta_{jk}$, where $\mathbf{1}_N$ is a $N$-vector of 1's. In our implementation, we use the Lagrange finite element for $\{K_m, P_m, \mathcal{A}_m\}$, where $K_m$ is a simplex in $\mathbb{R}^d$, $P_m$ is the polynomial of order $k$ and $\boldsymbol{\alpha}_{mj} = \boldsymbol{u}(x_{mj})$ with $x_{mj} \in K_m$; more details can be found in Ern and Guermond (2004, Section 1.2.3).

For any vector-valued function $\boldsymbol{u} \in [H^1(\Omega)]^N$, let $\boldsymbol{u}_{|K_m}$ denote its restriction to $K_m$, i.e., $\boldsymbol{u}(x)_{|K_m} = 1_{K_m}(x)\boldsymbol{u}(x)$. The finite element approximation of $\boldsymbol{u}$ on $K_m$ is given by

$$\boldsymbol{u}_{|K_m} \approx \boldsymbol{u}_{mh} := \sum_{j=1}^{p} \boldsymbol{\beta}_{mj}\phi_{mj},$$

where $\boldsymbol{\beta}_{mj}$ is an $N$-vector of coefficients in the basis expansion. The combination of $\boldsymbol{u}_{mh}$'s over $K_m$, $m \in M$, provides a global approximation of $\boldsymbol{u}$ over the whole domain $\Omega$, i.e.,

$$\boldsymbol{u}_h = \sum_{m=1}^{M} \boldsymbol{u}_{mh} = \sum_{m=1}^{M}\sum_{j=1}^{p} \boldsymbol{\beta}_{mj}\phi_{mj}, \tag{9}$$

which has a matrix-vector representation

$$\boldsymbol{u}_h = \boldsymbol{U}'\boldsymbol{\phi}, \tag{10}$$

where $\boldsymbol{U}$ is the $(Mp) \times N$ matrix formed by concatenating row vectors $\boldsymbol{\beta}'_{mj}$ in row-wise and $\boldsymbol{\phi}$ is the $(Mp)$-dimensional column vector of $\phi_{mj}$'s. In (9) and (10) we explicitly extended $\phi_{mj}$ to be zero outside $K_m$. The collection of vector-valued functions with expression (10) is a finite-dimensional linear space. We call this space the finite element space and denote it as $G_h$, where the subscript denotes the mesh size $h$.

The finite element method for solving the unconstrained integral equation (6) seeks $\boldsymbol{u}_h \in G_h$ such that

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) = c(\boldsymbol{v}_h) \qquad \text{for each } \boldsymbol{v}_h \in G_h, \tag{11}$$

where $a(\boldsymbol{u}, \boldsymbol{v}) = \int \boldsymbol{u}'\boldsymbol{A}\boldsymbol{v}$ and $c(\boldsymbol{v}) = \int \boldsymbol{f}'\boldsymbol{v}$. Following (10), we can write $\boldsymbol{u}_h = \boldsymbol{U}'\boldsymbol{\phi}$ and similarly $\boldsymbol{v}_h = \boldsymbol{V}'\boldsymbol{\phi}$. Both of the lhs and rhs of (11) can be simply represented as algebraic forms as follows. First, since

$$\boldsymbol{u}'_h\boldsymbol{A}\boldsymbol{v}_h = \text{trace}(\boldsymbol{\phi}'\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}'\boldsymbol{\phi}) = \text{trace}(\boldsymbol{\phi}\boldsymbol{\phi}'\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}'),$$

we have

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega \boldsymbol{u}'_h\boldsymbol{A}\boldsymbol{v}_h = \text{trace}(\int_\Omega \boldsymbol{\phi}\boldsymbol{\phi}'\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}') = \text{trace}(\boldsymbol{\Phi}\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}'),$$

where $\boldsymbol{\Phi} = \int_\Omega \boldsymbol{\phi}\boldsymbol{\phi}'$. Second, since

$$\boldsymbol{f}'\boldsymbol{v}_h = \text{trace}(\boldsymbol{f}'\boldsymbol{V}'\boldsymbol{\phi}) = \text{trace}(\boldsymbol{\phi}\boldsymbol{f}'\boldsymbol{V}'),$$

we have that

$$c(\boldsymbol{v}_h) = \int_\Omega \boldsymbol{f}'\boldsymbol{v}_h = \text{trace}(\int_\Omega \boldsymbol{\phi}\boldsymbol{f}'\boldsymbol{V}') = \text{trace}(\boldsymbol{F}\boldsymbol{V}'),$$

where $\boldsymbol{F} = \int_\Omega \boldsymbol{\phi}\boldsymbol{f}'$. Therefore, the integral equation (11) is equivalent to the following linear system

$$\text{trace}((\boldsymbol{\Phi}\boldsymbol{U}\boldsymbol{A} - \boldsymbol{F})\boldsymbol{V}') = 0 \qquad \text{for each } \boldsymbol{V} \in \mathbb{R}^{N\times(Mp)},$$

which is equivalent to

$$\boldsymbol{\Phi}\boldsymbol{U} = \boldsymbol{F}\boldsymbol{A}^{-1}. \tag{12}$$

We now introduce the boundary constraints. We partition (after reordering the elements) the basis function vector $\boldsymbol{\phi}$ used in $\boldsymbol{u}_h = \boldsymbol{U}'\boldsymbol{\phi}$ into two vectors $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}_b$ in column-wise such that, $\boldsymbol{\phi}_0$ is a column vector of the $\phi_{mj}(x)$'s satisfying $\phi_{mj|\partial\Omega} = 0$ and $\boldsymbol{\phi}_b$ is a column vector of the $\phi_{mj}(x)$'s satisfying $\phi_{mj|\partial\Omega} \neq 0$. Suppose that $\boldsymbol{\phi}_0$ has $Q$ elements and $\boldsymbol{\phi}_b$ has $R$ elements. Since the number of columns of $\boldsymbol{\phi}$ is $Mp$, $Q + R = Mp$. With the partition, we have that

$$\boldsymbol{u}_h = \boldsymbol{U}_0'\boldsymbol{\phi}_0 + \boldsymbol{U}_b'\boldsymbol{\phi}_b, \tag{13}$$

where $\boldsymbol{U}_0$ and $\boldsymbol{U}_b$ are submatrices consisting of $\boldsymbol{U}$'s rows corresponding to $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}_b$ respectively. Substituting (13) into equation (12), we have that

$$\boldsymbol{\Phi}_0\boldsymbol{U}_0 + \boldsymbol{\Phi}_b\boldsymbol{U}_b = \boldsymbol{F}\boldsymbol{A}^{-1}, \tag{14}$$

where $\boldsymbol{\Phi}_0 = \int_\Omega \boldsymbol{\phi}\boldsymbol{\phi}_0'$ and $\boldsymbol{\Phi}_b = \int_\Omega \boldsymbol{\phi}\boldsymbol{\phi}_b'$.

The boundary constraints restricted to $G_h$ can be written as

$$\int_{\partial\Omega} \boldsymbol{y}'\boldsymbol{u}_{h|\partial\Omega}\, v = \int_{\partial\Omega} b\, v \quad \text{for each } v \in G_h.$$

Using (14) and dropping the basis functions whose values are zero at the boundary, we obtain

$$\int_{\partial\Omega} \boldsymbol{y}'\boldsymbol{U}_b'\,\boldsymbol{\phi}_{b|\partial\Omega}\,\phi_{mj|\partial\Omega} = \int_{\partial\Omega} b\,\phi_{mj|\partial\Omega} \quad \text{for each } \phi_{mj|\partial\Omega} \neq 0,$$

which implies

$$\int_{\partial\Omega} \boldsymbol{y}'\boldsymbol{U}_b'\,\boldsymbol{\phi}_{b|\partial\Omega}\,\boldsymbol{\phi}_{b|\partial\Omega}' = \int_{\partial\Omega} b\,\boldsymbol{\phi}_{b|\partial\Omega}'.$$

Letting $\boldsymbol{B} = \int_{\partial\Omega} \boldsymbol{\phi}_{b|\partial\Omega}\boldsymbol{\phi}_{b|\partial\Omega}'$ and $\boldsymbol{b} = \int_{\partial\Omega} b\,\boldsymbol{\phi}_{b|\partial\Omega}'$, we have

$$\boldsymbol{y}'\boldsymbol{U}_b'\boldsymbol{B} = \boldsymbol{b}', \tag{15}$$

We decompose $\boldsymbol{U}_b$ into two components: one orthogonal to $\boldsymbol{y}$ and the residual. Let $\boldsymbol{O}_{\boldsymbol{y}}$ be $N \times (N-1)$ matrix of $N-1$ column vectors orthogonal to $\boldsymbol{y}$, which can be obtained by the Gram-Schmidt process. There exist $\boldsymbol{Z} \in \mathbb{R}^{R\times(N-1)}$ and $\boldsymbol{z} \in \mathbb{R}^{R\times 1}$ satisfying

$$\boldsymbol{U}_b = \boldsymbol{Z}\boldsymbol{O}_{\boldsymbol{y}}' + \boldsymbol{z}\boldsymbol{y}'. \tag{16}$$

Therefore, (14) becomes

$$\boldsymbol{\Phi}_0\boldsymbol{U}_0 + \boldsymbol{\Phi}_b\boldsymbol{Z}\boldsymbol{O}_{\boldsymbol{y}}' + \boldsymbol{\Phi}_b\boldsymbol{z}\boldsymbol{y}' = \boldsymbol{F}\boldsymbol{A}^{-1} \tag{17}$$

Since

$$\boldsymbol{y}'\boldsymbol{U}_b' = \boldsymbol{y}'(\boldsymbol{O}_{\boldsymbol{y}}\boldsymbol{Z}' + \boldsymbol{y}\boldsymbol{z}') = \boldsymbol{y}'\boldsymbol{y}\boldsymbol{z}',$$

equation (15) gives us

$$\boldsymbol{y}'\boldsymbol{y}\boldsymbol{z}'\boldsymbol{B} = \boldsymbol{b}',$$

therefore,

$$(\boldsymbol{y}'\boldsymbol{y})\boldsymbol{z} = \boldsymbol{B}^{-1}\boldsymbol{b}. \tag{18}$$

In summary, we first solve (18) for $\boldsymbol{z}$, then solve (17) for $\boldsymbol{U}_0$ and $\boldsymbol{Z}$, and then calculate $\boldsymbol{U}_b$ using (16). The finite element solution of the variational problem (8) is given by $\boldsymbol{u}_h = \boldsymbol{U}_0\boldsymbol{\phi}_0 + \boldsymbol{U}_b\boldsymbol{\phi}_b$. By the theory of finite element method (Ern and Guermond, 2004), the approximate solution $\boldsymbol{u}_h$ converges to the solution of the original problem as the mesh size $h$ tends to zero.

## 4. Patching Constrained Local GPs for Efficient Computation of Global GP regression

This section presents our patched GP regression method. We start from some notations. We partition the domain $\Omega$ of the data into small local regions $\{\Omega_s, s = 1, \ldots, S\}$ and partition the training data set $\mathcal{D} = \{(x_n, y_n) : n = 1, \ldots, N\}$ accordingly into $S$ data sets $\mathcal{D}_s := \{(x_n, y_n) \in \mathcal{D} : x_n \in \Omega_s\}$. We then calculate the local prediction function $f_s$ for the local region $\Omega_s$ using the data set $\mathcal{D}_s$. There are some issues with this localized solution. First, the prediction at around $\partial\Omega_s$ is not as accurate as the prediction at the interior of $\Omega_s$ mainly because of the less number of observations available around $\partial\Omega_s$. In particular, when $S$ becomes large, the boundary regions also increase. Therefore, the inaccuracy at the boundaries $\partial\Omega_s$ can have significant negative effects on the overall prediction accuracy. Second, two local GP regressions from two neighboring local regions $\Omega_s$ and $\Omega_t$ produce different predictions $f_s$ and $f_t$ at the shared boundary, making the prediction discontinuous on the boundary. This discontinuity is unacceptable since continuity of the prediction is often desired. We propose to impose boundary constraints such that the two neighboring local GP regressions give the same predictions on the shared boundary.

Section 4.1 applies the finite element method of Section 3 to solve a boundary constrained local GP problem for each local region when the boundary constraints are given. Section 4.2 gives two methods for estimating the boundary constraints. Section 4.3 discusses some implementation details, including calculation of the integrations involving the finite elements, and learning parameters of the covariance function of the GP regression.

### 4.1 Boundary-constrained Local GP

For two neighboring local regions $\Omega_s$ and $\Omega_t$, let $\Gamma_{st} = \overline{\Omega}_s \cap \overline{\Omega}_t$ denote the shared boundary, where $\overline{\Omega}_s$ is the closure of $\Omega_s$. The prediction function $f$ specialized on $\Gamma_{st}$ is denoted by a boundary function $b_{st}(x)$ for $x \in \Gamma_{st}$. For the time being, we assume that $b_{st}(x)$ is known and shall discuss in next section how to estimate $b_{st}(x)$. Fix a domain $\Omega_s$, suppose all its boundary functions $\{b_{st} : \forall t, \Gamma_{st} \neq \emptyset\}$ are known. Consider the following local GP problem on $\Omega_s$

$$\begin{aligned} &\underset{\boldsymbol{u}_s}{\text{Minimize}} \quad J(\boldsymbol{u}_s) = \int_{\Omega_s} \left\{ \frac{1}{2} \boldsymbol{u}_s(x)' \boldsymbol{A}_s \boldsymbol{u}_s(x) - \boldsymbol{f}_s(x)' \boldsymbol{u}_s(x) \right\} dx \\ &\text{subject to } \boldsymbol{y}_s' \boldsymbol{u}_s(x) = b_{st}(x) \text{ on } x \in \Gamma_{st}, \ \forall t, \ \Gamma_{st} \neq \emptyset \end{aligned} \tag{19}$$

where $\boldsymbol{A}_s$, $\boldsymbol{f}_s(x)$ and $\boldsymbol{y}_s$ are the localized versions of $\boldsymbol{A}$, $\boldsymbol{f}(x)$ and $\boldsymbol{y}$, which are all computed using the data in $\Omega_s$. The constraints used in (19) restrict two local GP prediction functions $f_s$ and $f_t$ to have the same prediction $b_{st}$ on the shared boundary $\Gamma_{st}$.

As in Section 3, we replace the boundary constraints by the weak form, approximate $\boldsymbol{u}_s$ by its finite element approximation $\boldsymbol{u}_{s,h} = \boldsymbol{U}_{s,0}' \boldsymbol{\phi}_{s,0} + \boldsymbol{U}_{s,b}' \boldsymbol{\phi}_{s,b}$, where $\boldsymbol{\phi}_{s,0}$ is a vector of local basis functions $\phi_{mj}$'s satisfying $\phi_{mj}(x)_{|\Gamma_{st}} = 0$ for all $t$'s, and $\boldsymbol{\phi}_{s,b}$ is a vector of local basis functions $\phi_{mj}$'s satisfying $\phi_{mj}(x)_{|\Gamma_{st}} \neq 0$ for all $t$'s, and $\boldsymbol{\phi}_s = (\boldsymbol{\phi}_{s,0}', \boldsymbol{\phi}_{s,b}')'$. Let $N_s$ be the length of $\boldsymbol{y}_s$ and $\boldsymbol{O}_{\boldsymbol{y}_s}$ be $N_s \times (N_s - 1)$ matrix of $N_s - 1$ column vectors orthogonal to $\boldsymbol{y}_s$. We can decompose $\boldsymbol{U}_{s,b}$ into $\boldsymbol{U}_{s,b} = \boldsymbol{Z}_s \boldsymbol{O}_{\boldsymbol{y}_s}' + \boldsymbol{z}_s \boldsymbol{y}_s'$. As we derived in Section 3, the finite element solution of the local problem (19) is obtained by solving the following linear

system of equations for $\boldsymbol{U}_{s,0}$, $\boldsymbol{Z}_s$ and $\boldsymbol{z}_s$,

$$\begin{aligned} \boldsymbol{\Phi}_{s,0}\boldsymbol{U}_{s,0} + \boldsymbol{\Phi}_{s,b}\boldsymbol{Z}_s\boldsymbol{O}'_{\boldsymbol{y}_s} + \boldsymbol{\Phi}_{s,b}\boldsymbol{z}_s\boldsymbol{y}'_s &= \boldsymbol{F}_s\boldsymbol{A}_s^{-1}, \\ (\boldsymbol{y}'_s\boldsymbol{y}_s)\boldsymbol{z}_s &= \boldsymbol{B}_s^{-1}\boldsymbol{b}_s, \end{aligned} \tag{20}$$

where $\boldsymbol{\Phi}_{s,0} = \int_{\Omega_s} \boldsymbol{\phi}_s\boldsymbol{\phi}'_{s,0}$, $\boldsymbol{\Phi}_{s,b} = \int_{\Omega_s} \boldsymbol{\phi}_s\boldsymbol{\phi}'_{s,b}$, $\boldsymbol{F}_s = \int_{\Omega_s} \boldsymbol{\phi}_s\boldsymbol{f}'_s$, and

$$\begin{aligned} \boldsymbol{B}_s &= \sum_{t:\Gamma_{st}\neq\emptyset} \int_{\Gamma_{st}} \boldsymbol{\phi}_{s,b|\Gamma_{st}}\boldsymbol{\phi}_{s,b|\Gamma_{st}}, \\ \boldsymbol{b}_s &= \sum_{t:\Gamma_{st}\neq\emptyset} \int_{\Gamma_{st}} b_{st|\Gamma_{st}}\boldsymbol{\phi}_{s,b|\Gamma_{st}}. \end{aligned}$$

Since $b_{st}(x)$ is unknown, we need to estimate $\boldsymbol{b}_s$ using the procedure to be described in Section 4.2. Using the second equation of (20), we obtain $\boldsymbol{z}_s = \boldsymbol{B}_s^{-1}\boldsymbol{b}_s/\boldsymbol{y}'_s\boldsymbol{y}_s$. Substituting this expression of $\boldsymbol{z}_s$ in the first equation of (20), we obtain

$$\begin{aligned} \boldsymbol{\Phi}_{s,0}\boldsymbol{U}_{s,0} + \boldsymbol{\Phi}_{s,b}\boldsymbol{Z}_s\boldsymbol{O}'_{\boldsymbol{y}_s} &= \boldsymbol{F}_s\boldsymbol{A}_s^{-1} - \boldsymbol{\Phi}_{s,b}\boldsymbol{z}_s\boldsymbol{y}'_s, \\ &= \boldsymbol{F}_s\boldsymbol{A}_s^{-1} - \boldsymbol{\Phi}_{s,b}\frac{\boldsymbol{B}_s^{-1}\boldsymbol{b}_s\boldsymbol{y}'_s}{\boldsymbol{y}'_s\boldsymbol{y}_s}. \end{aligned} \tag{21}$$

We then solve this equation for $\boldsymbol{U}_{s,0}$ and $\boldsymbol{Z}_s$, and compute $\boldsymbol{U}_{s,b} = \boldsymbol{Z}_s\boldsymbol{O}'_{\boldsymbol{y}_s} + \boldsymbol{z}_s\boldsymbol{y}'_s$. Finally the finite element solution of the constrained local GP problem (19) is $\boldsymbol{u}_{s,h} = \boldsymbol{U}'_{s,0}\boldsymbol{\phi}_{s,0} + \boldsymbol{U}'_{s,b}\boldsymbol{\phi}_{s,b}$.

When the number of mesh cells in each local region is $M$ on average and the number of training data in each domain is $N_S$, the computation complexity of solving the constrained local GP regression (21) for one local region is $O(N_S^3 + M^2)$. The first part $N_S^3$ is for inverting $\boldsymbol{A}_j$, and the second part is for inverting $\boldsymbol{\Phi}_{s,0}$ and $\boldsymbol{\Phi}_{s,b}$, which is proportional to $M^2$ because $\boldsymbol{\Phi}_{s,0}$ and $\boldsymbol{\Phi}_{s,b}$ are sparse banded matrices; note that the complexity of solving a linear system with a banded coefficient matrix is proportional to the square of the size of the linear system (Mahmood et al., 1991). The cost per local region is mostly bounded by the cubic term $O(N_S^3)$. The total computational cost for $S$ local regions is thus $O(SN_S^3)$, which also equals to $O(NN_S^2)$.

#### 4.1.1 Illustrative Output of the Constrained GP Formulation

Our constrained local GP regression provides a good approximate to a full GP regression when the value of boundary function $b_{st}$ is close to the mean prediction of the full GP regression at $\Gamma$. To show this, we performed a simple simulation study. In the study, we generated a data set of 6,000 noisy observations from a zero-mean Gaussian process with an exponential covariance function,

$$y_i = f(x_i) + \epsilon_i \quad \text{for } i = 1, \ldots, 6000,$$

where $x_i \sim \text{Uniform}(0, 10)$ and $\epsilon_i \sim \mathcal{N}(0, 1)$ are independently sampled, and $f(x_i)$ is simulated by the R package `RandomField`. Three hundreds of the observations were randomly sampled to learn a Gaussian process regression (full GP) and the covariance hyperparameters, while the remaining 5,700 were kept for test data. The domain $[0, 10]$ was partitioned into ten local regions of size 1 delineated by boundary points $\{0.0, 0.1, 0.2, 0.3, \ldots 1.0\}$,

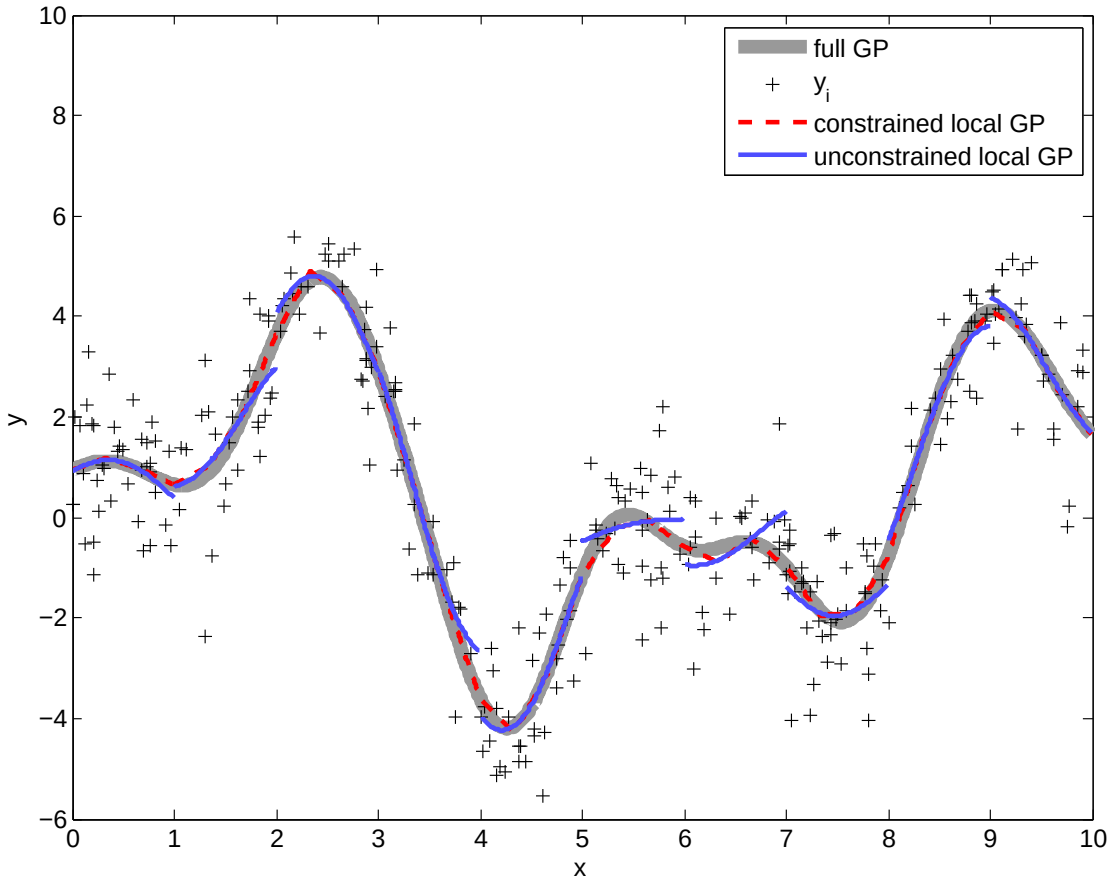


Figure 1: Effect of the boundary constraint on Gaussian process regression.

and the 300 observations were distributed into the local regions accordingly. For each domain, an unconstrained GP regression (unconstrained local GP) and a constrained GP regression (constrained local GP) were learned. When the constrained local GP was learned, the values of the regression outcome at the boundary points were constrained to be equal to the mean prediction of a full GP regression at the points. Note this is not a fair comparison since the full GP prediction was used. This example was just used to show the room for improvement if constraints are used.

Figure 1 shows the comparison of the mean predictions from a full GP, the constrained local GP and the unconstrained local GP regressions. Compared to the unconstrained local GP regression, the constrained model is much closer to a full GP regression especially at the boundary points. The maximum difference in the mean predictions of the constrained model and a full GP over the test data was 0.2803, while that of the unconstrained version was 0.6411. The advantage of placing the boundary constraints on the local GP in improving the prediction accuracy is clear. However, placing the boundary constraints requires knowing the value of the ground truth $f$ at boundary points, i.e., boundary function $b_{st}$. Estimating the values of $f$ at boundary points is the subject of the next section.

### 4.2 Estimation of Boundary Values

The prediction function $f$ at $\Gamma_{st}$, i.e., the boundary function $b_{st}$, is unknown and needs to be estimated before the constrained local GP regression is solved. We propose two approaches for the estimation boundary values for the local GP regressions. The first approach is to train a separate local GP regression using a subset of training data located around the boundary $\Gamma_{st}$—this together with the constrained local GP regressions, leads to a two-step procedure. The second approach is to iteratively solve the boundary value estimation and the constrained local GP problems.

#### 4.2.1 Localized Estimation

This approach is motivated by our observation that a GP regression for a local domain gives accurate prediction at the center of the domain. We propose to estimate the prediction function $f$ at $\Gamma_{st}$ by learning a local GP regression with a subset of the training data that belong to a neighborhood of $\Gamma_{st}$. When $x \in \mathcal{R}$, $\Gamma_{st}$ is a point coordinate in $\mathcal{R}$, and its neighborhood is defined by an interval $[\Gamma_{st}-r, \Gamma_{st}+r]$ around it with half width $r > 0$. When $x \in \mathcal{R}^d$ in general, $\Gamma_{st}$ is a $d-1$ dimensional hyperplane within $\mathcal{R}^d$, and its neighborhood is defined by $Nh_r(\Gamma_{st})$,

$$Nh_r(\Gamma_{st}) = \{x' \in \mathcal{R}^d; \min_{x \in \Gamma_{st}} ||x' - x||_2 \leq r\}.$$

The value of the prediction function $f$ at $x \in \Gamma_{st}$, i.e. $b_{st}(x)$, is estimated by the mean prediction of the local GP regression built from a subset of training data, $\mathbf{x}_{st} = \{x_n \in \mathcal{D}; x_n \in Nh_r(\Gamma_{st})\}$ and the corresponding observed outputs $\mathbf{y}_{st}$,

$$\hat{b}_{st}(x) = \boldsymbol{k}'_{\mathbf{x}_{st}*}(\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_{st}\mathbf{x}_{st}})^{-1}\mathbf{y}_{st}. \tag{22}$$

When the average number of observations in the local neighborhood $Nh_r(\Gamma_{st})$ is $N_B$, the complexity of this boundary estimation per boundary is $O(N_B^3)$. When the dimension of the domain $\Omega$ is $d$ and the domain is decomposed into $S$ local regions of $d$-simplices, the total number of the boundaries in between the local regions is proportional to $dS$. So, the complexity of this boundary estimation procedure is $O(dSN_B^3)$.

#### 4.2.2 Block Gauss-Seidel Iteration

The system of equations for the constrained local GP regression given by (20) is converted into the following equation for three unknown variables $\boldsymbol{U}_{s,0}$, $\boldsymbol{Z}_s$ and $\boldsymbol{b}_s$,

$$\boldsymbol{\Phi}_{s,0}\boldsymbol{U}_{s,0} + \boldsymbol{\Phi}_{s,b}\boldsymbol{Z}_s\boldsymbol{O}'_{\boldsymbol{y}_s} + \boldsymbol{\Phi}_{s,b}\boldsymbol{B}_s^{-1}\boldsymbol{b}_s\boldsymbol{y}'_s/(\boldsymbol{y}'_s\boldsymbol{y}_s) = \boldsymbol{F}_s\boldsymbol{A}_s^{-1}, \tag{23}$$

where we used $(\boldsymbol{y}'_s\boldsymbol{y}_s)\boldsymbol{z}_s = \boldsymbol{B}_s^{-1}\boldsymbol{b}_s$ to replace the $\boldsymbol{z}_s$ in the first line of (20) with $\boldsymbol{B}_s^{-1}\boldsymbol{b}_s/(\boldsymbol{y}'_s\boldsymbol{y}_s)$. Note that the above equation depends on an unknown boundary function $b_{st}$ only through $\boldsymbol{b}_s$. We will estimate the vector quantity $\boldsymbol{b}_s$ instead of estimating the boundary function $b_{st}$ directly. The equation for $\boldsymbol{U}_{s,0}$, $\boldsymbol{Z}_s$ and $\boldsymbol{b}_s$ can be solved iteratively by the block Gauss-Seidel method (Saad, 2003). The block Gauss-Seidel method is an iterative solver for a linear system that partitions a number of unknowns into multiple blocks and solves the linear system for one block at a time while keeping the other blocks fixed. In our problem,

we have two block of unknowns, one block for $\boldsymbol{U}_{s,0}$ and $\boldsymbol{Z}_s$ and the other block for $\boldsymbol{b}_s$. The corresponding block Gauss-Seidel iteration is as follows. Start with an initial guess $\boldsymbol{b}_s^{(0)}$. We used a zero vector for the initial guess. At iteration $k$, we perform the following two steps sequentially:

**Step 1.** With $\boldsymbol{b}_s^{(k-1)}$ fixed from the previous iteration, obtain $\boldsymbol{U}_{s,0}^{(k)}$ and $\boldsymbol{Z}_s^{(k)}$ by solving

$$\boldsymbol{\Phi}_{s,0}\boldsymbol{U}_{s,0}^{(k)} + \boldsymbol{\Phi}_{s,b}\boldsymbol{Z}_s^{(k)}\boldsymbol{O}'_{\boldsymbol{y}_s} = \boldsymbol{F}_s\boldsymbol{A}_s^{-1} - \boldsymbol{\Phi}_{s,b}\boldsymbol{B}_s^{-1}\boldsymbol{b}_s^{(k-1)}\boldsymbol{y}'_s/(\boldsymbol{y}'_s\boldsymbol{y}_s), \quad s = 1, \ldots, S. \quad (24)$$

**Step 2.** Obtain $\boldsymbol{b}_s^{(k)}$ by solving

$$\boldsymbol{\Phi}_{s,b}\boldsymbol{B}_s^{-1}\boldsymbol{b}_s^{(k)}\boldsymbol{y}'_s/(\boldsymbol{y}'_s\boldsymbol{y}_s) = \boldsymbol{F}_s\boldsymbol{A}_s^{-1} - \boldsymbol{\Phi}_{s,0}\boldsymbol{U}_{s,0}^{(k)} - \boldsymbol{\Phi}_{s,b}\boldsymbol{Z}_s^{(k)}\boldsymbol{O}'_{\boldsymbol{y}_s}, \quad s = 1, \ldots, S. \quad (25)$$

Note that the equations in the system (24) appeared in Step 1 can be solved in parallel for $s = 1, \ldots, S$. But the equations in the system (25) appeared in Step 2 should be solved collectively for all $s = 1, \ldots, S$, since $\boldsymbol{b}_s$ is shared by multiple local regions and thus appears in multiple equations. The block Gauss-Seidel method converges very fast. When the dimension of the domain $\Omega$ is $d$ and the domain is decomposed into $S$ local regions of $d$-simplices, the total number of boundaries in between the local regions is proportional to $dS$. On the other hand, the size of the linear system to be solved in Step 2 is proportional to the number of boundaries. Since the coefficient matrix of the linear system is a banded matrix, the complexity of solving such a linear system is proportional to the square of the size of the linear system (Mahmood et al., 1991), that is, $O(d^2S^2)$.

#### 4.2.3 Numerical Comparison

This section numerically compares the two aforementioned solutions for boundary value estimation. We used the same data set used in Section 4.1.1 and applied the same partitioning scheme for splitting the entire domain into 10 local regions, in between which there are nine boundary locations. We applied the localized estimation method and the iterative block Gauss-Seidel approach for estimating $f(x)$ at the nine locations, and compared them with the estimated values from a full GP regression. Figure 2-(a) shows the comparison results. The root mean squared difference of the localized estimation to a full GP regression was 0.0775, while that of the iterative approach was 0.1369. Both of the errors are far below the noise parameter $\sigma = 1$. The computation time for the estimation was comparable, 0.041328 seconds for the localized method and 0.038071 seconds for the iterative approach.

For another comparison, we generated a synthetic data set in 2-d of 8,000 noisy observations from a zero-mean Gaussian process with an exponential covariance function of scale one and variance 10,

$$y_i = f(x_i) + \epsilon_i \quad \text{for } i = 1, \ldots, 8000,$$

where $x_i \sim \text{Uniform}([0,6] \times [0,6])$ and $\epsilon_i \sim \mathcal{N}(0,1)$ were independently sampled, and the Gaussian process realization $f(x_i)$ was simulated by the R package `RandomField`. We split the input domain $[0,6] \times [0,6]$ into sixteen local regions of equal size, and 1,881 test locations were chosen uniformly surrounding the local region boundaries. For each test location, we obtained the prediction based on a full GP regression and computed the differences of the

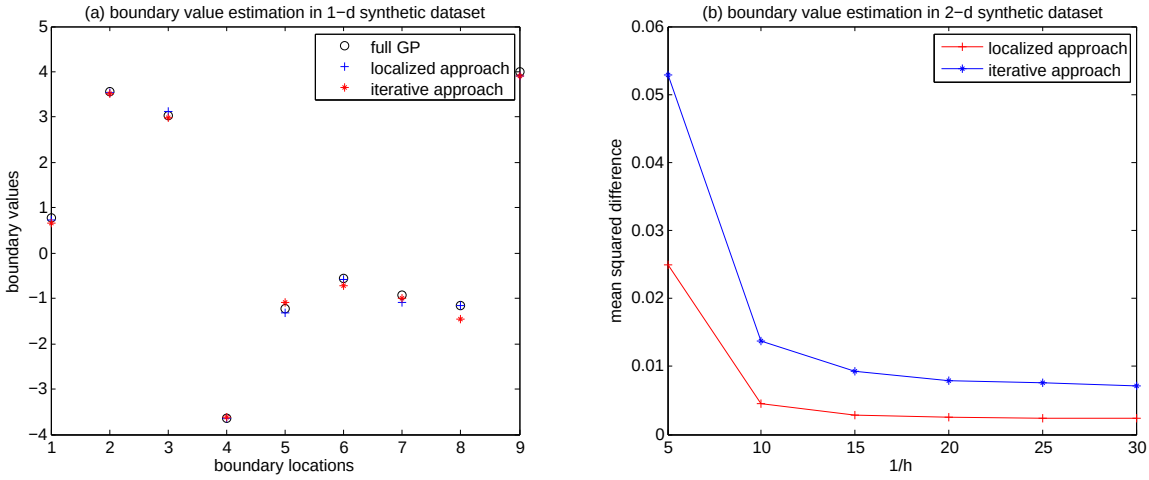


Figure 2: Comparison of the two proposed methods of boundary value estimation with a full GP regression.

boundary estimation by the localized approach or the iterative approach and the full GP regression prediction. The mean squared differences versus mesh size $h$ are plotted in Figure 2-(b). Again, the localized approach works better.

### 4.3 Computation Complexity and Implementation Details

The total computation complexity of our proposed approach is the summation of two complexities, one for the constrained GP regressions and the other for the boundary value estimation. It is $O(SN_S^3 + dSN_B^3)$ when the localized estimation approach is used for boundary value estimation, and it is $O(SN_S^3 + d^2S^2)$ when the block Gauss-Seidel iteration is used. Since $SN_S = N$ and $N_B$ is a constant, the complexity is $O(NN_S^2 + dS)$ or $O(NN_S^2 + d^2S^2)$ respectively.

#### 4.3.1 Evaluation of Integrals for quantities in Equation (20)

The integrals for defining several quantities in equation (20) can be computed effectively using well-established finite element computations. Suppose that $\Omega \subset \mathbb{R}^d$ and we use the Lagrange finite elements of polynomial degree $k$, where the $s$th local region $\Omega_s$ is partitioned into $M$ mesh cells $\{\mathcal{K}_m; m = 1, \ldots, M\}$ for the finite element approximation. The $\boldsymbol{\phi}_s$ is a column vector of the Lagrange basis functions for the mesh cells,

$$\{\phi_{m,j}; m = 1, \ldots, M, j = 0, 1, \ldots, J\}, \text{ and } J = \begin{pmatrix} d+k \\ k \end{pmatrix}.$$

It is well known that each $\phi_{m,j}$ is a polynomial function of barycentric coordinates $\lambda_j$'s with respect to the $d$-simplex $\mathcal{K}_m$ (Ern and Guermond, 2004, pages 22-23). One can use the integral formula for barycentric coordinates (Voitovich and Vandewalle, 2008) to compute

$\boldsymbol{\Phi}_{s,0}$ and $\boldsymbol{\Phi}_{s,b}$. For example, when $d = 2$,

$$\int_{\mathcal{K}_m} \lambda_{j_1}^a \lambda_{j_2}^b \lambda_{j_3}^c = 2|\mathcal{K}_m| \frac{a!b!c!}{(2+a+b+c)!},$$

and when $d = 3$,

$$\int_{\mathcal{K}_m} \lambda_{j_1}^a \lambda_{j_2}^b \lambda_{j_3}^c \lambda_{j_4}^d = 6|\mathcal{K}_m| \frac{a!b!c!d!}{(3+a+b+c+d)!},$$

where $|\mathcal{K}_m|$ is the volume of $\mathcal{K}_m$. Since $\phi_{m,j_1}\phi_{m,j_2}$ is also a polynomial functions of $\lambda_j$'s, one can use the previous integration formulas to evaluate $\int_{\mathcal{K}_m} \phi_{m,j_1}\phi_{m,j_2}$ and

$$\int_{\Omega_s} \phi_{m,j_1}\phi_{m,j_2} = \sum_m \int_{\mathcal{K}_m} \phi_{m,j_1}\phi_{m,j_2}. \tag{26}$$

For the values of $\boldsymbol{F}_s$, one can take the finite element approximation of $\boldsymbol{f}_s$, where each function $f_i$ in $\boldsymbol{f}_s$ is approximated by

$$\sum_j \alpha_{m,j}^{(i)} \phi_{m,j} \text{ on } \mathcal{K}_m.$$

With this approximation, $\boldsymbol{F}_s$ becomes

$$\begin{aligned}\int_{\Omega_s} \boldsymbol{\phi}_s f_i &= \sum_m \int_{\mathcal{K}_m} \boldsymbol{\phi}_s \sum_j \alpha_{m,j}^{(i)} \phi_{m,j} \\ &= \sum_m \sum_j \alpha_{m,j}^{(i)} \int_{\mathcal{K}_m} \boldsymbol{\phi}_s \phi_{m,j}.\end{aligned}$$

The last integral can be computed using (26).

Since $\phi_{m,j|\Gamma_{st}}$ is a polynomial function of barycentric coordinates with respect to $\Gamma_{st}$,

$$\int_{\Gamma_{st}} \phi_{m,j_1|\Gamma_{st}} \phi_{m,j_2|\Gamma_{jk}}$$

can be computed using the integral formulas in barycentric coordinates, facilitating the evaluation of $\boldsymbol{B}_s$ and $\boldsymbol{b}_s$.

#### 4.3.2 Learning Covariance Parameters

By far, our discussions have been made when using fixed parameters (often referred to as hyperparameters in the literature) for the covariance function $k(\cdot,\cdot)$. In this subsection, we discuss how to choose the hyperparameters. Basically, we follow the approach in Park et al. (2011), which has two options, namely choosing different hyperparameters for each local region or choosing the same hyperparameters for all local regions. When different hyperparameters are chosen for each local region, the hyperparameters are estimated by maximizing the local marginal likelihood functions. Specifically, the local hyperparameters, denoted by $\theta_s$ associated with each $\Omega_s$, are selected such that they minimize the negative log marginal likelihood:

$$ML_s(\theta_s) := -\log p(\boldsymbol{y}_s;\theta_s) = \frac{n_s}{2}\log(2\pi) + \frac{1}{2}\log|\boldsymbol{A}_s| + \frac{1}{2}\boldsymbol{y}_s'\boldsymbol{A}_s^{-1}\boldsymbol{y}_s, \tag{27}$$

where $\boldsymbol{A}_s$ depends on $\theta_s$. Note that (27) is the marginal likelihood of the standard local kriging model typically seen in geostatistics.

When we want to choose the same hyperparameters applied for all local regions, we choose the hyperparameter $\theta$ such that it minimizes

$$ML(\theta) = \sum_{s=1}^{S} ML_s(\theta), \tag{28}$$

where the summation of the negative log local marginal likelihoods is over all local regions. The above treatment implicitly assumes that the data from each local region are mutually independent. We used the criterion (28) for all numerical comparisons presented below.

## 5. Numerical Study of Patched GP and Comparison with DDM

In this section, we present the numerical performance of our patched GP method for different tuning parameters, compared to the full GP regression. We also compare our patched GP method with its precursor, the DDM (Park et al., 2011).

### 5.1 Data Sets and Evaluation Criteria

We considered four data sets: one synthetic data set in 1-d, one synthetic data set in 2-d, and three real spatial data sets both in 2-d. The two synthetic data sets were generated by the R package `RandomField`. The first data set in 1-d (hereafter denoted by `synthetic-1d`) consists of 6,000 noisy observations from a zero-mean Gaussian process with an exponential covariance function of scale one and variance 10,

$$y_i = f(x_i) + \epsilon_i \quad \text{for } i = 1, \ldots, 6000,$$

where $x_i \sim \text{Uniform}(0, 10)$ and $\epsilon_i \sim \mathcal{N}(0, 1)$ were independently sampled, and the Gaussian process realization $f(x_i)$ was simulated by the R package. The synthetic data set in 2-d (hereafter denoted by `synthetic-2d`) consists of 8,000 noisy observations from a zero-mean Gaussian process with an exponential covariance function of scale one and variance 10,

$$y_i = f(x_i) + \epsilon_i \quad \text{for } i = 1, \ldots, 8000,$$

where $x_i \sim \text{Uniform}([0, 6] \times [0, 6])$ and $\epsilon_i \sim \mathcal{N}(0, 1)$ were independently sampled, and the Gaussian process realization $f(x_i)$ was simulated by the R package `RandomField`. The two synthetic data sets were used to show how our proposed method performs, compared to the full GP regression.

The first real data set, `TCO`, contains data collected by NIMBUS-7/TOMS satellite to measure the total column of ozone over the globe on Oct 1 1988. This set consists of 48,331 measurements. The second real data set, `TCO.L2`, also contains the total column of ozone measured by the same satellite on the same date at much more locations (182,591 locations). The third real data set, `ICETHICK`, is the ice thickness profile for a portion of the western Antartic ice sheet, which is available at `http://nsidc.org/`. The data set has 32,481 measurements. As shown in Figure 3, the `ICETHICK` data set has some sparse regions with very few training points, while the `TCO` data set has a very dense distribution of the training points; `TCO.L2` data set has even denser distribution.

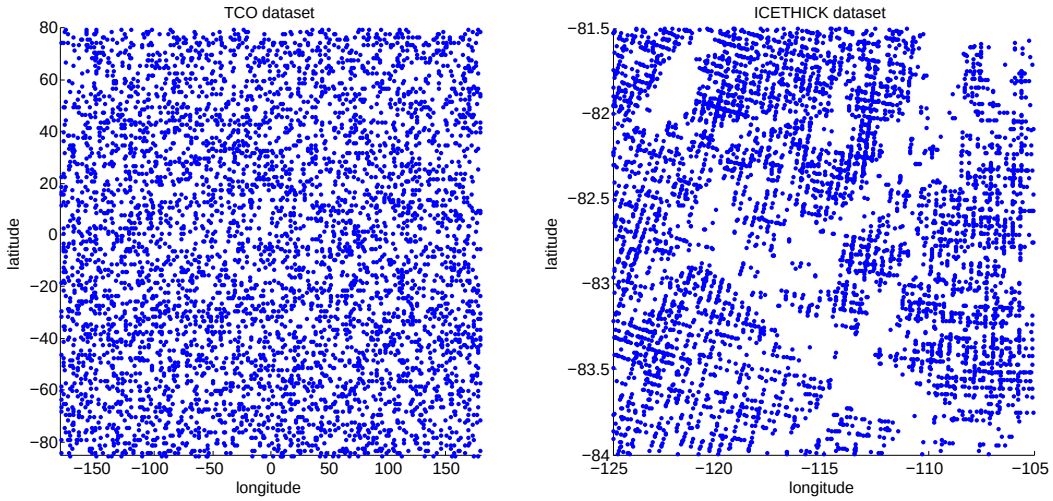


Figure 3: Spatial distribution of the measurements for two real data sets. A dot represent one measurement.

Using the three real spatial data sets, we can compare the computation time and prediction accuracy of the patched GP with other methods. We randomly split each data set into a training set containing 90% of the total observations and a test set containing the remaining 10% of the observations. To compare the computational efficiency of methods, we measure two computation times, the training time (including the time for hyperparameter learning) and the prediction (or test) time. For comparison of accuracy, we use two measures on the set of the test data, denoted as $\{(x_t, y_t); t = 1, \ldots, T\}$, where $T$ is the size of the test set. The first measure is the mean squared error (MSE)

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \mu_t)^2, \tag{29}$$

which measures the accuracy of the mean prediction $\mu_t$ at location $x_t$. The second one is the negative log predictive density (NLPD)

$$\text{NLPD} = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{(y_t - \mu_t)^2}{2\sigma_t^2} + \frac{1}{2}\log(2\pi\sigma_t^2)\right], \tag{30}$$

which considers the accuracy of the predictive variance $\sigma_t$ as well as the mean prediction $\mu_t$. These two criteria were used broadly in the GP regression literature. A smaller value of MSE or NLPD indicates a better performance.

When applying the patched GP, one issue is how to partition the whole domain into local regions, also known as *meshing* in the finite element analysis literature (Ern and Guermond, 2004). The patched GP works with any shapes of meshing. For this paper, we used a uniform triangular mesh, where each local region is a triangular shaped region of the same size. The implementation of the meshing was performed using the `DistMesh` MATLAB

software (Persson and Strang, 2004). We used the localized estimation presented in Section 4.2.1 for boundary value estimation, and the hyperparameters of a covariance function was obtained by minimizing (28) and was applied for all local regions. All numerical studies were performed on a computer with Intel Xeon Processor W3520 and 6GB memory.

### 5.2 Performance of Patched GP with Different Tuning Parameters

The patched GP has two tuning parameters, number of local regions $S$ and mesh size $h$ in finite element approximation. If the number of local regions ($S$) is one (i.e. there is no split to local regions), the patched GP should converge to a full GP as the mesh size of the patched GP's finite element approximation goes to zero. In this section we illustrate how the patched GP works for $S > 1$ and different mesh sizes using the data sets described in the previous section.

For `synthetic-1d`, we uniformly partitioned the domain $[0, 10]$ into $S$ local regions of equal size where $S$ varies over $\{2, 4, 6, 8, 10\}$. The number of meshes per local region is denoted by $M$, and it is related to mesh size $h$, which is the length of an interval mesh. We randomly split 6,000 observations in `synthetic-1d` into a training data set of 4,500 observations and a test data set of 1,500 observations. For each $S$ and $h$, we used the training data set to learn the patched GP and a full GP, and compared the mean squared difference of the patched GP and a full GP over the test data set. Figure 4-(a) shows the mean squared difference versus $S$ and $h$. Regardless of $S$, the difference converges to almost zero (about $e^{-6}$) as $h$ goes to zero, which implies that the mean prediction of the patched GP becomes very close to that of a full GP even with a large $S$; this can be qualitatively seen in Figure 4-(b), -(c) and -(d). In other words, the performance of the patched GP does not vary much with the choice of $S$ although the method with a larger $S$ typically converges faster.

For `synthetic-2d`, we uniformly partitioned the domain into $S$ local regions of equal size where $S$ varies over $\{68, 47, 32, 17, 10, 4\}$. The number of meshes per local region is denoted by $M$, and it is related to mesh size $h$, which is the side length of a triangular mesh for `synthetic-2d`. We randomly split 8,000 observations in `synthetic-2d` into a training data of size 6,500 and a test data set of size 1,500. For each $S$ and $h$, we used the training data set to learn the patched GP and a full GP, and compared the mean squared difference between the patched GP and the full GP over the test data set. Figure 5 shows the mean squared difference versus $S$ and $h$. Similar to the 1-d case, the performance of patched GP does not vary much with different $S$ values for the two synthetic data sets.

We also evaluated the performance of the patched GP on the real data sets `TCO` and `ICETHICK` for different values of $S$ and $M$. As described in Section 5.1, 90% of each data set was randomly chosen and used as a training data set, and the remaining 10% was used for computing the MSE. Figure 6 summarizes the results. The performance of the patched GP did not vary much for different $S$. If $S$ is large, the overall computation complexity decreases significantly as $M$ decreases. Therefore, in general, a larger $S$ is preferred.

### 5.3 Comparison with DDM

Our proposed patched GP method is the direct enhancement of DDM. In this section, we compare the numerical performance of DDM and patched GP.

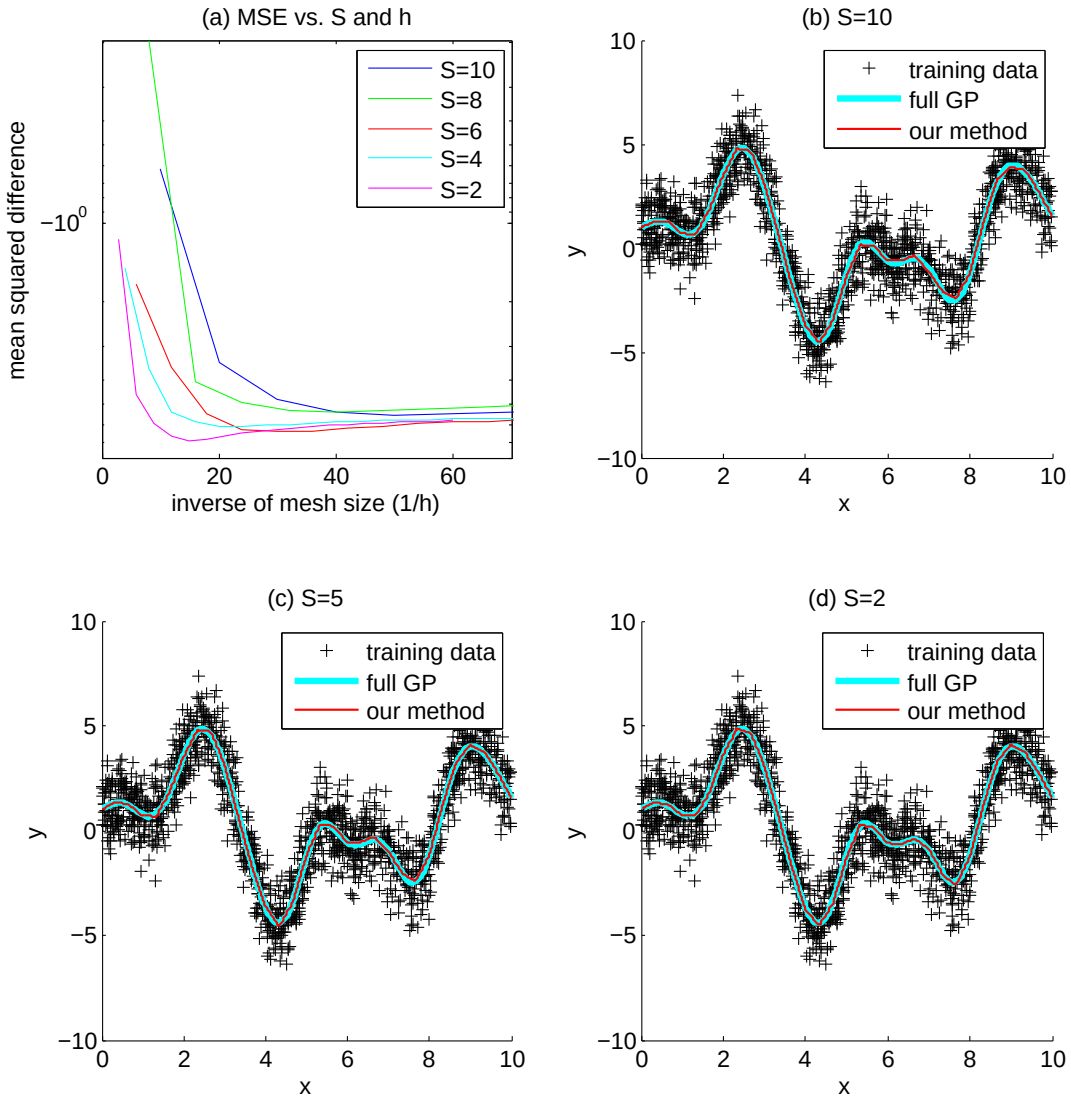


Figure 4: Mean squared difference of the patched GP and a full GP over the test data set for `synthetic-1d`; (a) shows the mean squared differences for different $S$ and $h$ parameter values, and (b)-(d) illustrate the mean predictions of the patched GP and a full GP for different $S$'s with fixed $1/h = 3$.

### 5.3.1 Overall Performance

We used three real data sets in 2-d to compare the MSEs and NLPDs of patched GP and DDM versus the total computation time (training and test time), which includes the time for hyperparameter learning, model learning, and prediction.

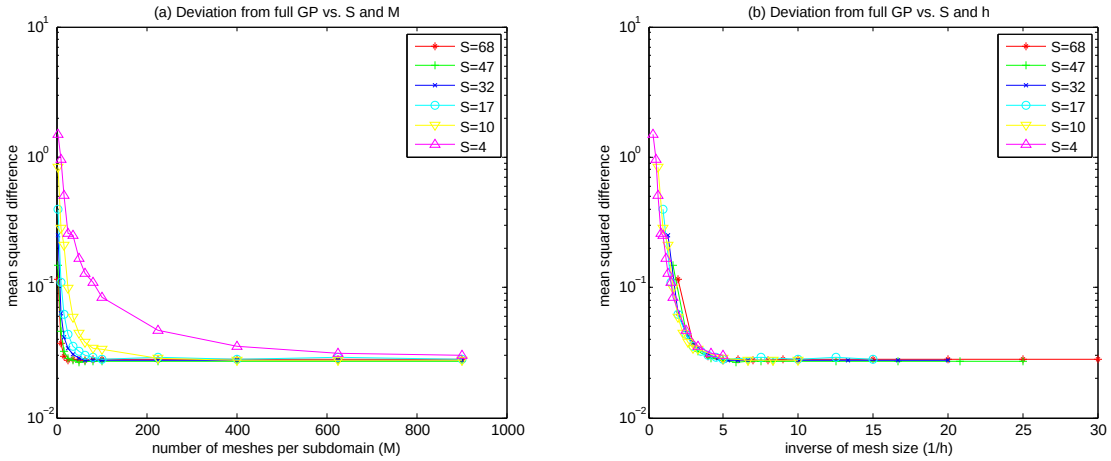


Figure 5: Mean squared difference of the patched GP and a full GP versus $S$ and $h$ for `synthetic-2d`

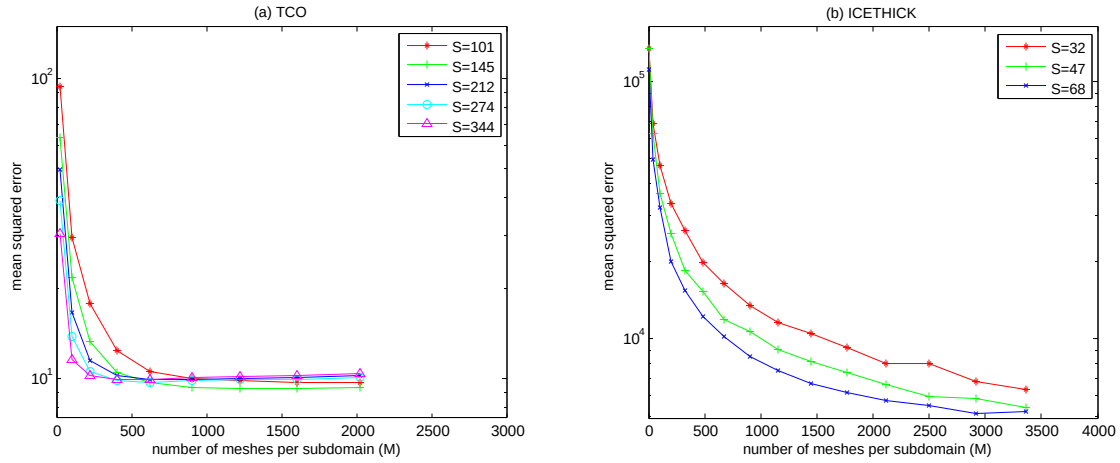


Figure 6: MSE of the patched GP versus $S$ and $h$ for `TCO` and `ICETHICK`.

For both methods, we fixed $S$, $S = 145$ for `TCO`, $S = 623$ for `TCO.L2`, and $S = 47$ for `ICETHICK`, because the performance did not vary much with the choice of $S$. We applied the squared exponential covariance function and the same covariance hyperparameter learning method for both DDM and patched GP, which is described in Park et al. (2011, Section 5). In the hyperparameter learning, we used a subset of the training data. The fractions of the training data set used for the hyperparameter learning varied over $\{0.3, 0.5, 0.8, 1.0\}$. As the fraction increases, we expect the training time increases, and the accuracy of the hyperparameter estimation and the final prediction improves. For patched GP, the number of meshes per local region for the finite element approximation ($h$) varied from 5 to 30 with step size 5. As seen in Section 5.2, the increase of $h$ implies the increase of the

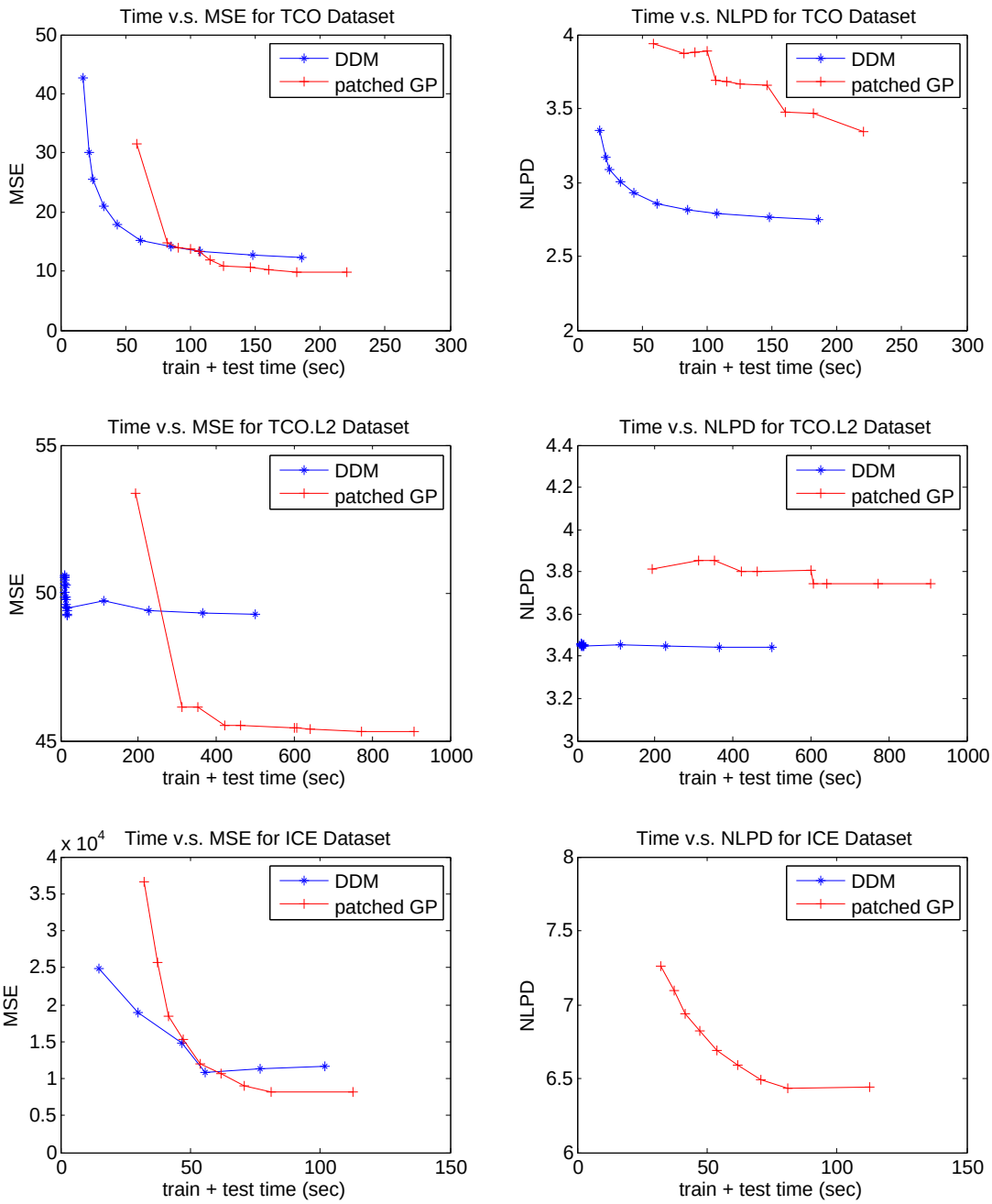


Figure 7: Prediction accuracy versus total computation time. For ICETHICK (ICE) data set, the NLPDs of DDM are imaginary numbers since the predictive variance estimates were all negative.

overall prediction accuracy and the total computation time. For DDM, we varied the number of degree of freedoms on a local region boundary from 5 to 30 with step size 5. For each experimental setting, we performed 20 replicated experiments with new random splits of training and test data sets. We randomly split each data set into a training set containing 90% of the total observations and a test set containing the remaining 10% of the observations. The MSEs, NLPDs and the total computation times were averaged to reduce the variation caused by random splits.

Figure 7 shows the MSE and NLPD versus the total computation time for both DDM and patched GP. For shorter computation time, DDM performed better in terms of MSE but patched GP obtained lower MSEs with longer computation time. In terms of NLPD, the DDM was better for `TCO` and `TCO.L2`. The NLPD is roughly the squared bias of the predictive mean divided by the predictive variance. Since the patched GP is better than the DDM in MSE, the better NLPD performance of the DDM can be attributed to difference in variance estimation. For `TCO` and `TCO.L2`, the DDM's variance estimation is sufficiently large to cover most observations. However, the DDM's variance estimation is sometimes too small, being negative. For example, the DDM produced the negative predictive variances for `ICETHICK`, so the resulting NLPDs are imaginary numbers. The issue with DDM regarding possible negative predictive variances has been reported in Pourhabib et al. (2014). The same problem occurred in this numerical example.

#### 5.3.2 Comparison in Boundary Value Estimation

We compared DDM and patched GP for boundary value estimation. For this comparison, we used the localized estimation approach described in Section 4.2.1 with $N_B = 50$. The comparison was primarily focused on (1) how the boundary estimation of each method on a boundary location is close to the mean prediction of a full GP regression on the same location, and (2) how the boundary estimations of two neighboring local regions on a boundary point are closed to each other. We fixed $S = 16$ and tried different mesh sizes $h$ for patched GP and different numbers of the control points placed on each boundary ($p$) for DDM, which are directly relevant to the performance of boundary estimation. The $h$ varied over one fifth of a local region size through one thirtieth of a local region size, while $p$ comparably varied over five through thirty.

We used the whole `synthetic-2d` data set as a training data set to train both of the methods, and 1,881 test locations were chosen uniformly from local region boundaries. For each test location, we obtained the mean prediction of a full GP regression. The squared differences of the mean prediction of DDM or patched GP to that of a full GP at the test locations are averaged to obtain the mean squared difference. This difference versus $h$ or $p$ is plotted in Figure 8-(a). As $h$ decreases, the boundary estimation of patched GP at the test locations converges to the mean prediction of a full GP regression at the same locations, while the boundary estimation of DDM keeps deviating from a full GP result.

We also compared how consistent the mean predictions from two neighboring local regions at their shared boundary are. We simply took the two mean predictions from two neighboring local regions at some of the 1,881 test locations on their shared boundary. The squared differences were taken and averaged over all shared boundaries. The mean squared differences versus $h$ or $p$ are plotted in Figure 8-(b). The mean squared differences for

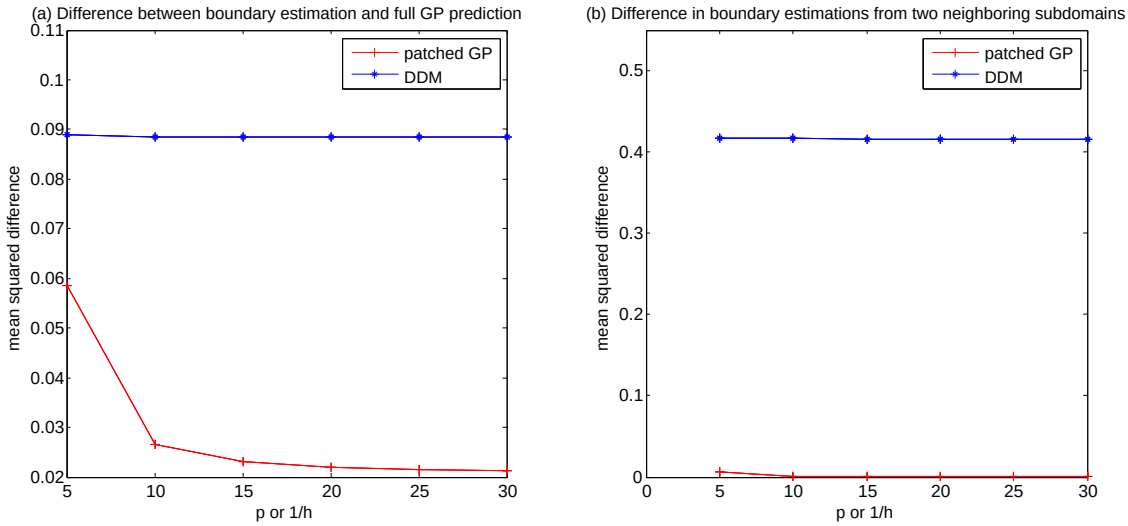


Figure 8: Accuracy of Boundary Estimation.

patched GP are almost zero, while those for DDM are significantly non-zero relative to those of patched GP.

## 6. Numerical Comparison with Other State-of-The-Art methods

This section compares patched GP with other state-of-the-art methods. Section 6.1 contains the comparison to several localized approaches for GP regression, while Section 6.2 contains the comparison to the Gaussian Markov random field approach to the GP regression (Lindgren et al., 2011).

### 6.1 Comparison with Other Local GP Methods

In this section, we compare patched GP with other localized approaches for GP regression, including BCM (Tresp, 2000), PIC (Snelson and Ghahramani, 2007), and RBCM (Deisenroth and Ng, 2015); we used the author's implementation of BCM and implemented RBCM and PIC with matlab by ourselves. We used three real data sets `TCO`, `TCO.L2` and `ICETHICK` to compare the MSEs and NLPDs of the approaches versus the total computation time, which includes the time for hyperparameter learning, model learning and prediction. We used the squared exponential covariance function and used the whole training data set to choose hyperparameters for all of the compared methods. For patched GP, we fixed $S = 145$ for `TCO`, $S = 623$ for `TCO.L2`, and $S = 47$ for `ICETHICK`, and the number of meshes per local region was varied from 5 to 40 with step size 5. For BCM and RBCM, we varied the number of local experts $M \in \{100, 150, 200, 250, 300, 600\}$ for `TCO`, $M \in \{50, 100, 150, 200, 250, 300\}$ for `TCO.L2`, and $M \in \{50, 100, 150, 200, 250, 300\}$ for `ICETHICK`. For PIC, we varied the total number of local regions $m \in \{100, 150, 200, 250, 350\}$ for `TCO`, $m \in \{100, 200, 300, 400, 600\}$ for `TCO.L2`, and $m \in \{50, 100, 150, 200\}$ for `ICETHICK`, and also varied the number of in-

ducing inputs ($M \in \{100, 150, 200, 250, 300, 400, 500, 600, 700\}$) for all of the data sets. We used the k-means clustering for splitting training data for both of BCM, RBCM, and PIC.

Figure 9 shows the logarithms of MSEs and NLPDs versus total computation times for the three data sets. For `TCO` data set, the BCM, RBCM, and patched GP obtained more accurate prediction than the PIC, and the patched GP was computationally more efficient than the BCM and RBCM. For `TCO.L2` data set, the patched GP uniformly outperformed other competing methods, scaling better than BCM and RBCM and achieving better MSE than the PIC. It is interesting to see that BCM is almost identically performing as the RBCM when $N$ is so large like in `TCO.L2` data set. For `ICETHICK` data set, the PIC and patched GP obtained more accurate prediction than the BCM and the RBCM. Please note that the training data are quite densely spread over the whole domain for `TCO` while the training data are sparse for some local regions in `ICETHICK`. The patched GP worked well for both of the cases, while the PIC worked better for the sparse case and the BCM worked better for the dense case. The RBCM has shown much better results than the BCM for the sparse case but it is not better than the patched GP and PIC. The PIC combines a global model with local models, which may help to improve the performance for the sparse case.

### 6.2 Comparison with GMRF

In this section, we compare patched GP with the Gaussian Markov random field approach to the GP regression (Lindgren et al., 2011, GMRF), which was reported to scale great with massive data set; we implemented the GMRF with matlab. The major checkpoints of this comparison are the scalability and prediction accuracy. We used three real data sets of different sizes, `ICETHICK` ($N = 32,813$), `TCO` ($N = 48,311$) and `TCO.L2` ($N = 182,591$) to compare the MSEs, NLPDS, and computation time of the approaches. In this comparison, we used the exponential covariance function, since the GMRF does not work with the squared exponential covariance function used for the other comparisons; at least, the construction of the precision matrix for Gaussian Markov random field is not straightforward. The GMRF does not have any tuning parameters, and the hyperparameter learning of the GMRF was performed using 5% of the training data; the MSE performance did not change much as the percentage increases, so we chose the smallest percentage to obtain the smallest computation time. For patched GP, we presented the results with the combinations of tuning parameters that obtain the best RMSE and the worst RMSE. To be specific, we fixed $S = 145$ for `TCO`, $S = 47$ for `ICETHICK`, and $S = 623$ for `TCO.L2`, while the number of meshes per local region was varied from 5 to 40 with step size 5.

Table 1 summarizes the comparison results. The computation time of patched GP increases linearly in data size $N$, while the GMRF's computation time increases in $\mathrm{O}(N^2)$. This is not surprising because the GMRF's computation depends on $n_z$, the number of nonzero elements in the precision matrix, proportionally in $n_z^2$ or $n_z^{3/2}$, and the $n_z$ increases at least linearly in $N$. For prediction performance, the best RMSE of the patched GP was at least comparable or better than that of GMRF. The patched GP uniformly outperformed the GMRF in terms of NLPD, which means that the posterior distribution of the patched GP was better fitted to test data sets.

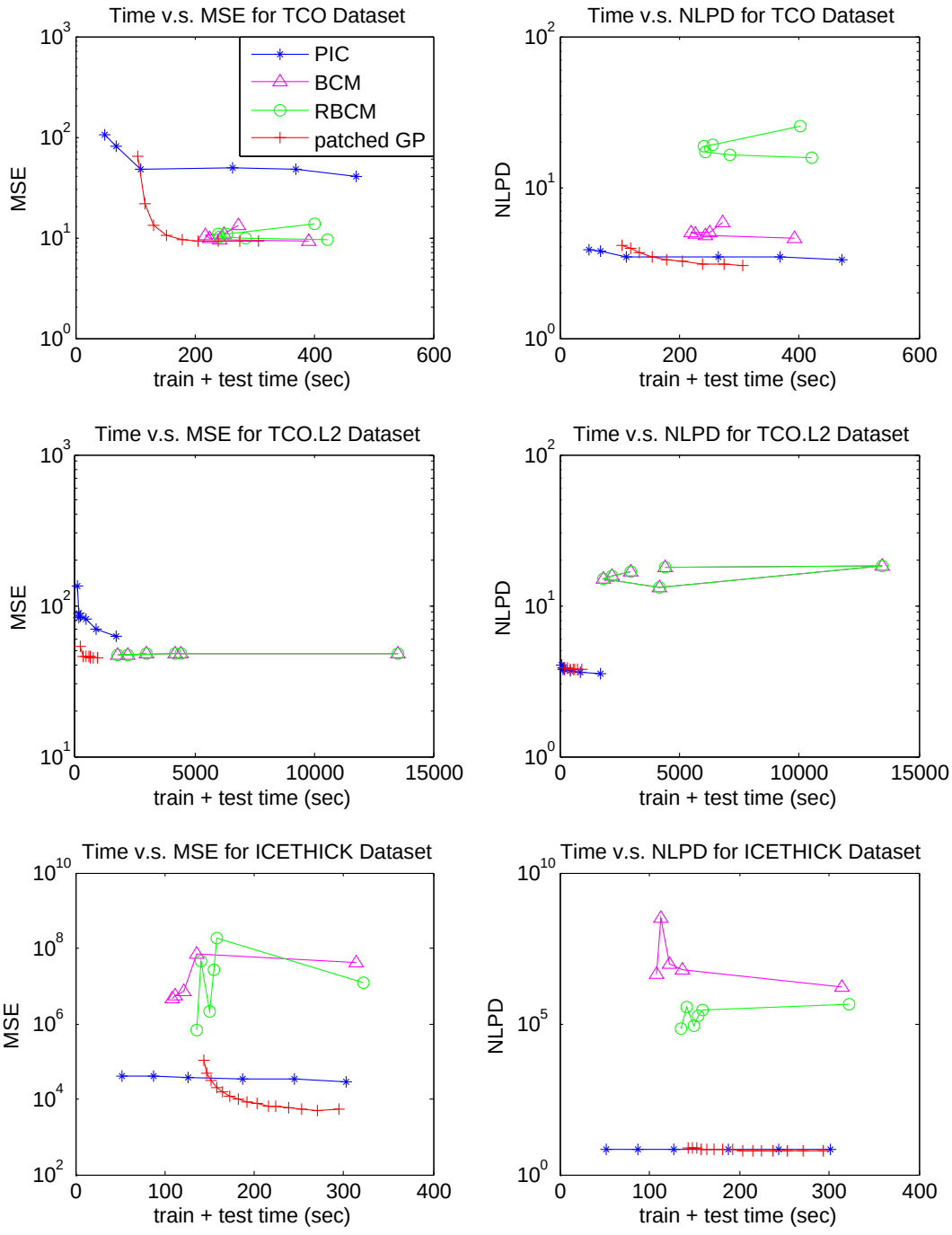


Figure 9: Prediction accuracy versus total computation time. The legend in the upper left panel applies to all other panels.

| Datasets | patched GP (best, worst) | | | GMRF | | |
|---|---|---|---|---|---|---|
| | Time | RMSE | NLPD | Time | RMSE | NLPD |
| TCO | (250.4, 48.4) | (2.85, 5.75) | (3.20, 3.91) | 651.4 | 2.78 | 5.01 |
| TCO.L2 | (807.4, 192.2) | (6.74, 7.31) | (3.70, 3.82) | 5702.7 | 9.24 | 5.26 |
| ICETHICK | (212.1, 31.2) | (89.30, 199.33) | (6.12, 6.80) | 385.0 | 89.80 | 7.08 |

Table 1: Performance comparison of the patched GP with GMRF: the time unit used is second.

## 7. Conclusion

We developed a method for solving a Gaussian process regression with constraints on a domain boundary and also developed a solution approach based on a finite element method. The method is then applied to local GP regressions as a building block to develop the patched GP method as a computationally efficient solver of a large-scale Gaussian process regression or spatial kriging problem. The patched GP solves two issues of the simple local GP approaches, namely the inaccuracy and inconsistency of prediction on the boundaries of neighboring local regions. Comparing with its precursor DDM, the patched GP has an improved way of considering the constraints related to the boundary regions. Both methods reformulate the GP regression as an optimization problem, the patched GP method improves DDM by rewriting the optimization problem in a function space and using the finite element methods to solve the required integrals arising from the solution of the minimization problem. The patched GP method is mathematically more elegant and its competitiveness to existing methods is demonstrated through numerical studies.

## Acknowledgments

The authors thank the reviewers for constructive comments. The authors also thank Anton Schwaighofer at Microsoft for sharing the BCM implementation. Chiwoo Park was supported by grants from the National Science Foundation (CMMI-1334012) and the Air Force Office of Scientific Research (FA9550-13-1-0075). Jianhua Z. Huang was supported by grants from the National Science Foundation (DMS-1208952) and the Air Force Office of Scientific Research (FA9550-13-1-0075).

## Appendix A. Proof of Proposition 1

Note that $[L^2(\Omega)]^N$ is a Hilbert space with the following inner product

$$(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{u}'\boldsymbol{v}.$$

We define a bi-linear form on $[L^2(\Omega)]^N$ by $a : [L^2(\Omega)]^N \times [L^2(\Omega)]^N \to \mathbb{R}$,

$$a(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{A}\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{u}'\boldsymbol{A}\boldsymbol{v} \qquad \text{for } \boldsymbol{u}, \boldsymbol{v} \in [L^2(\Omega)]^N,$$

a linear functional $c : [L^2(\Omega)]^N \to \mathbb{R}$ as

$$c(\boldsymbol{u}) = \int_\Omega \boldsymbol{f}'\boldsymbol{u},$$

and define $J(\boldsymbol{u}) = \frac{1}{2}a(\boldsymbol{u}, \boldsymbol{u}) - c(\boldsymbol{u})$. Since $\boldsymbol{A}$ is a $N \times N$ (real) positive definite matrix, the bi-linear form $a(\boldsymbol{u}, \boldsymbol{v})$ is symmetric and positive. Let $\alpha$ be the smallest eigenvalue of $\boldsymbol{A}$. We have that

$$a(\boldsymbol{u}, \boldsymbol{u}) \geq \alpha||\boldsymbol{u}||^2, \qquad \forall \boldsymbol{u} \in [L^2(\Omega)]^N.$$

Therefore, the bi-linear form $a$ is coercive. It follows from Ern and Guermond (2004, Proposition 2.4) that, $\boldsymbol{u}$ satisfies $a(\boldsymbol{u}, \boldsymbol{v}) - c(\boldsymbol{v}) = 0$ for every $\boldsymbol{v} \in [L^2(\Omega)]^N$ if and only if it minimizes $J(\boldsymbol{u})$ over $\boldsymbol{u} \in [L^2(\Omega)]^N$. Note that the coercivity of the bi-linear form $a$ can be interpreted as a strong convexity property of the functional $J(\boldsymbol{u})$, which makes the problem have a unique optimal solution (Ern and Guermond, 2004, Lemma 2.2). ∎

## Appendix B. Proof of Proposition 2

We have already proven that $a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{u}'\boldsymbol{A}\boldsymbol{v}$ is coercive, symmetric and positive for $\boldsymbol{u}, \boldsymbol{v} \in [L^2(\Omega)]^N$ in the proof of Proposition 1. The same result holds for $\boldsymbol{u}, \boldsymbol{v} \in H_b$ because $H_b \subset [L^2(\Omega)]^N$. Since $H_b$ is a Hilbert space, solving the minimization problem is equivalent to solving the integral equation (8) by Ern and Guermond (2004, Proposition 2.4). ∎

## References

Sudipto Banerjee, Alan E Gelfand, Andrew O Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):825–848, 2008.

Tao Chen and Jianghong Ren. Bagging for Gaussian process regression. *Neurocomputing*, 72(7):1605–1610, 2009.

Noel Cressie and Gardar Johannesson. Fiexed rank kriging for very large spatial data sets. *Journal of Royal Statistical Society, Series B*, 70:209–226, 2008.

Paul J Curran and Peter M Atkinson. Geostatistics and remote sensing. *Progress in Physical Geography*, 22(1):61–78, 1998.

Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *Proceedings of the 32 nd International Conference on Machine Learning*, pages 1–10, 2015.

A. Ern and J.L. Guermond. *Theory and practice of finite elements*. Springer Verlag, 2004. ISBN 0387205748.

Reinhard Furrer, Marc G Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3), 2006.

Thore Graepel. Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations. In *International Workshop on Machine Learning*, volume 20, page 234, 2003.

Robert B Gramacy and Daniel W Apley. Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, pages 561–578, 2015.

Robert B Gramacy and Herbert KH Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103 (483), 2008.

Cari G Kaufman, Mark J Schervish, and Douglas W Nychka. Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555, 2008.

Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73 (4):423–498, 2011.

David JC MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural networks and machine learning*, volume 168 of *NATO ASI Series F Computer and Systems Sciences*, pages 133–166. Springer Verlag, 1998.

A Mahmood, DJ Lynch, and LD Philipp. A fast banded matrix inversion using connectivity of schur's complements. In *IEEE International Conference on Systems Engineering*, pages 303–306. IEEE, 1991.

Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local Gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

Chiwoo Park, Jianhua Z. Huang, and Yu Ding. Domain decomposition approach for fast Gaussian process regression of large spatial datasets. *Journal of Machine Learning Research*, 12:1697–1728, May 2011.

Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM review*, 46(2):329–345, 2004.

Arash Pourhabib, Faming Liang, and Yu Ding. Bayesian site selection for fast Gaussian process regression. *IIE Transactions*, 46(5):543–555, 2014.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*, pages 881–888. MIT Press, 2002.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Huiyan Sang and Jianhua Z. Huang. A full-scale approximation of covariance functions for large spatial data sets. *Journal of Royal Statistical Society, Series B*, 74:111–132, 2012.

Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics 9*. Society for Artificial Intelligence and Statistics, 2003.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *International Conference on Artifical Intelligence and Statistics 11*, pages 524–531. Society for Artificial Intelligence and Statistics, 2007.

Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

Florian Steinke and Bernhard Schölkopf. Kernels, regularization and differential equations. *Pattern Recognition*, 41(11):3271–3286, 2008.

Volker Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

Jarno Vanhatalo and Aki Vehtari. Modelling local and global phenomena with sparse Gaussian processes. In *the 24th Conference on Uncertainty in Artificial Intelligence*, page 571578. Association for Uncertainty in Artificial Intelligence, 2008.

Tatiana V Voitovich and Stefan Vandewalle. Barycentric interpolation and exact integration formulas for the finite volume element method. In *International Conference on Numerical Analysis and Applied Mathematics*, volume 1048, pages 575–579. AIP Publishing, 2008.