Sequential Process Convolution Gaussian Process Models via Particle Learning

Waley W. J. Liang and Herbert K. H. Lee
University of California, Santa Cruz

Abstract

The process convolution approach for constructing a Gaussian Process (GP) model is a computationally efficient approach for larger datasets in lower dimensions. Bayesian inference or specifically, Markov chain Monte Carlo, is commonly used for estimating the parameters of this model. Applying this model to sequential applications where data arrives on-line requires re-running the Markov chain for each new data arrival, which can be time consuming. This paper presents an on-line inference method for the process convolution GP model based on a Sequential Monte Carlo method called Particle Learning. This model is illustrated on a synthetic example and an optimization problem in hydrology.

Keywords: Sequential Monte Carlo; Markov chain Monte Carlo; Optimization; Spatial Modeling

1 Introduction

Gaussian processes (GP) have been widely used to model the underlying process of interest in regression and classification models (Neal, 1997, 1998; Rasmussen and Williams, 2006). Some of the major applications include computer experiments (Sacks et al., 1989), and models of spatial and spatio-temporal data (Cressie, 1991; Banerjee et al., 2003). Recent developments in GP models focus on using the Bayesian approach for inference because it provides for full accounting of uncertainty. In this approach, a Gaussian process with a chosen correlation function is specified as the prior for the underlying process of interest. Combining the prior distribution with the likelihood using Bayes' rule forms the posterior distribution which can be sampled using Markov Chain Monte Carlo (MCMC). One drawback of these standard GP models is that they require a matrix

decomposition whose complexity increases at a rate of the cube of the sample size, which makes them impractical for applications with moderately large datasets.

An alternative approach which can help alleviate the problem of large sample size is the process convolution approach to constructing a GP (Higdon, 1998, 2002; Calder et al., 2002; Paciorek and Schervish, 2006). This approach generates a GP by convolving a white noise process with a smoothing kernel. Bayesian inference of the model parameters again proceeds using MCMC. Although this approach is computationally efficient for applications with a large sample size, the batch nature of MCMC makes it unsuitable for sequential problems. For example, design points in computer simulation experiments are naturally generated sequentially so that MCMC has to be repeated for each new data arrival, which renders the whole inference process computationally demanding. In this paper, sequential inference for the process convolution GP model is introduced based on a Sequential Monte Carlo (SMC) method called *Particle Learning* (Carvalho et al., 2010). A similar approach has been developed by Gramacy and Polson (2009) where a standard GP is considered. The updating time of their model is on the order of $Q(t^2)$, where t denotes the time index (or the sample size at time t assuming that one data point arrives at a time). Although the computing time is a significant improvement from $Q(t^3)$ in the setting of MCMC inference, t is not a fixed constant and still has the potential problem of being too large for the model to be computationally efficient. In contrast, the process convolution approach has running time on the order of $Q(m^3)$ for each t, where m denotes the number of basis points that is always fixed and usually on the order of hundreds so that the model remains computationally efficient. This advantage would be more obvious as the sample size becomes even moderately large, and this may occur when a batch of data points is considered at each time step. Model results show that the sequential process convolution GP provides comparable model fitting to the standard GP approach at a faster speed.

This paper is organized as follows. Section 2 gives an overview of process convolution GP model (PCGP). Section 3 introduces Sequential Monte Carlo and Particle Learning (SPCGP). Section 4 provides details of forming on-line inference for process convolution GPs. Section 5 illustrates the model on a set of 1-d synthetic sinusoidal data and a 2-d optimization problem based on expected improvement. Discussion and conclusion is given in Section 6.

2 Process Convolution GP model

A Gaussian process can be specified via discrete process convolutions (DPC) (Higdon, 1998, 2002) as follows:

$$z(\mathbf{s}) \approx \sum_{j=1}^{m} k(\mathbf{u}_j - \mathbf{s})x(\mathbf{u}_j), \quad \mathbf{s}, \mathbf{u}_j \in \mathcal{S} \subseteq \mathbb{R}^d,$$

where $\{\mathbf{u}_1, \cdots, \mathbf{u}_m\}$ is a set of basis points in $\mathcal{S}$ with even spacing, $x(\cdot)$ is a White noise process with mean zero and precision λ, and $k(\cdot)$ is a symmetric kernel (e.g, Gaussian) defined over $\mathcal{S}$. In most applications, only a finite set of spatial sites $\{\mathbf{s}_1, \cdots, \mathbf{s}_n\} \in \mathcal{S}$ are of interest. In such cases, the above representation can be written in matrix form, $\mathbf{z} = \mathbf{K}\mathbf{x}$, where $\mathbf{z} = (z(\mathbf{s}_1), \cdots, z(\mathbf{s}_n))^\top$, $\mathbf{x} = (x(\mathbf{u}_1), \cdots, x(\mathbf{u}_m))^\top$, and $\mathbf{K}$ is a $(n \times m)$ matrix with elements $\mathbf{K}_{ij} = k(\mathbf{u}_j - \mathbf{s}_i)$.

In practice, the observation $y(\mathbf{s})$ recorded at location $\mathbf{s}$ is often decomposed in the following form,

$$y(\mathbf{s}) = \mu(\mathbf{s}) + z(\mathbf{s}) + \epsilon(\mathbf{s}), \tag{1}$$

where the mean function $\mu(\mathbf{s})$ is usually taken as a constant or linear, and is represented by $\mathbf{F}\boldsymbol{\beta}$, where $\mathbf{F}$ denotes the design matrix and $\boldsymbol{\beta}$ denotes the linear coefficient vector; $z(\mathbf{s})$ denotes the value of the underlying zero-mean stochastic process at location $\mathbf{s}$; $\epsilon(\mathbf{s}) \sim N(0, \phi^{-1})$ denotes Gaussian measurement error with ϕ being the precision. In the Bayesian GP model setting, the stochastic component z is usually given a zero-mean GP prior. In the process convolution approach, this is equivalent to imposing a White noise process prior on x. Bayesian inference about the model parameters can be performed by specifying conditionally conjugate priors for the parameters,

$$\begin{aligned}
&\boldsymbol{\beta} \sim N_{p+1}(\boldsymbol{\beta}_0\,, (\phi\mathbf{C})^{-1}), \quad \phi_\nu \sim G(a_y\,, b_y),\\
&\mathbf{x} \sim N_m(\mathbf{0}\,, (\lambda\mathbf{I}_m)^{-1}), \quad \lambda \sim G(a_x\,, b_x),
\end{aligned}$$

where $\boldsymbol{\beta}_0$, $\mathbf{C}$, a_x, a_y, b_x, b_y are constants. Combining with the likelihood, the conditional posterior distributions are given by:

$$\begin{aligned}
&\mathbf{x}|\cdots \sim N_m((\phi\mathbf{K}^\top\mathbf{K} + \lambda\mathbf{I}_m)^{-1}\phi\mathbf{K}^\top\mathbf{w}\ ,\ (\phi\mathbf{K}^\top\mathbf{K} + \lambda\mathbf{I}_m)^{-1}),\\
&\lambda|\cdots \sim G(m/2 + a_x\ ,\ 0.5\mathbf{x}^\top\mathbf{x} + b_x),\\
&\boldsymbol{\beta}|\cdots \sim N_{p+1}(((\mathbf{F}^\top\mathbf{F} + \mathbf{C})^{-1}(\mathbf{F}^\top\mathbf{v} + \mathbf{C}\boldsymbol{\beta}_0))\ ,\ (\phi(\mathbf{F}^\top\mathbf{F} + \mathbf{C}))^{-1}),\\
&\phi|\cdots \sim G(n/2 + a_y\ ,\ b_y + 0.5(s^2 + (\boldsymbol{\beta}_0 - \hat{\boldsymbol{\beta}})^\top(\mathbf{C}^{-1} + (\mathbf{F}^\top\mathbf{F})^{-1})^{-1}(\boldsymbol{\beta}_0 - \hat{\boldsymbol{\beta}}))),
\end{aligned}$$

where

$$\mathbf{v} = \mathbf{y} - \mathbf{K}\mathbf{x}, \quad \mathbf{w} = \mathbf{y} - \mathbf{F}\boldsymbol{\beta}, \quad s^2 = (\mathbf{v} - \mathbf{F}\hat{\boldsymbol{\beta}})^\top(\mathbf{v} - \mathbf{F}\hat{\boldsymbol{\beta}}), \quad \hat{\boldsymbol{\beta}} = (\mathbf{F}^\top\mathbf{F})^{-1}\mathbf{F}^\top\mathbf{v}.$$

Sampling from the posteriors distributions can be done using Gibbs sampling since all posterior distributions are in closed-form.

3 Sequential Monte Carlo and Particle Learning

In the state space models literature, there are two main statistical inference problems: 1) *sequential state filtering and parameter learning* which is characterized by the joint posterior distribution of states and parameters at each point in time, and 2) *state smoothing* which is characterized by the distribution of the states conditional on all data and marginalizing out all unknown parameters. That is, filtering and learning is about inferring the hidden states and parameters given only the currently available data at each time point, and smoothing is about inferring the states based on the full dataset. In the setting of linear Gaussian models, the Kalman filter (Kalman, 1960) provides analytical recursion equations for both filtering and smoothing assuming knowledge of parameters. For example, in the case of filtering, updating of the model at time $t+1$ is done by treating the model fitting at time t as a prior, which is then combined (using Bayes' rule) with the likelihood of new data arriving at time $t+1$. If the model has unknown static (independent of t) parameters, the full sequence of updating equations with all data up to the current time defines a likelihood which can be combined with a prior for Bayesian inference (West and Harrison, 1997). Depending on the prior, the resulting inferential complexity can go from being analytically tractable to intractable. For more general model specifications, it is common to apply Sequential Monte Carlo (SMC) methods which are also known as particle filters. SMC provides a numerical alternative to the inference problem of non-linear and/or non-Gaussian dynamical process, or when the parameters and their priors do not lead to tractable posteriors. In this paper, the emphasis is on the filtering/parameter learning problem. SMC uses a set of particles $\{Z_t^{(i)}\}_{i=1}^N$ to approximate the posterior distribution of the state information Z_t about the dynamic process, conditional on the data up to time t. The main task is to update the particle approximation from time t to $t+1$. Pure filtering of the state information (assuming knowledge of parameters) can be done using the *bootstrap filter* of Gordon et al. (1993) which upon arrival of new data $\mathbf{y}_{t+1}$ *propagates*

the particles via the state evolution equation $P(Z_{t+1}|Z_t)$, then *resamples* the propagated particles with weights proportional to the likelihood $P(\mathbf{y}_{t+1}|Z_{t+1})$. Another method for the same problem would be the Auxiliary Particle Filter (APF) of Pitt and Shephard (1999) which is based on a *resample-propagate* approach. Filtering with learning of unknown static parameters can be done using the filter of Liu and West (2001) which extends the APF by using a kernel approximation to the posterior of the parameters, or the filter of Storvik (2002) which assumes that the posterior of the parameters depends on a low-dimensional set of sufficient statistics that can be recursively updated. Although filtering algorithms can be used for learning of parameters, they are not efficient without modifications. A new class of SMC algorithms called particle learning (PL) (Carvalho et al., 2010) focuses on the parameter learning part and hence is more suitable for the sequential learning of static models (as opposed to dynamic models). PL is based on a resample-propagate approach as follows

$$\begin{aligned}
&\textbf{Resampling:} \quad P(Z_t|\mathbf{y}^{t+1}) \propto P(\mathbf{y}_{t+1}|Z_t)P(Z_t|\mathbf{y}^t), \\
&\textbf{Propogation:} \quad P(Z_{t+1}|\mathbf{y}^{t+1}) = \int P(Z_{t+1}|Z_t, \mathbf{y}_{t+1}) dP(Z_t|\mathbf{y}^{t+1}),
\end{aligned}$$

where Z_t denotes a particle that contains the *sufficient information* and $\mathbf{y}_t$ denotes the observation vector at time t. Sufficient information at time t may include the hidden states, sufficient statistics of the parameters, or even the parameters themselves. The above algorithm starts by resampling the sufficient information Z_t with probability weights proportional to the predictive distribution of the new data $P(\mathbf{y}_{t+1}|Z_t)$. Then, the new set of sufficient information is propagated based on a state transition distribution $P(Z_{t+1}|Z_t, \mathbf{y}_{t+1})$. Let $\{Z_t^{(i)}\}_{i=1}^N$ denote the set of particles that approximates $P(Z_t|\mathbf{y}^t)$, the actual implementation procedure of particle learning can be summarized as follows:

1. Sample indices $\{\zeta(j) : j = 1, \cdots, N\}$ with replacement from a Multinomial distribution with weights proportional to the predictive distribution, i.e., $P(\zeta(j) = i) \propto P(\mathbf{y}_{t+1} \mid Z_t^{(i)})$ for $i = 1, \cdots, N$. Set $\{Z_t^{(j)}\}_{j=1}^N = \{Z_t^{\zeta(j)}\}_{j=1}^N$.

2. Draw $Z_{t+1}^{(j)}$ from $P(Z_{t+1}|Z_t^{(j)}, \mathbf{y}_{t+1})$ to obtain a new particle set $\{Z_{t+1}^{(j)}\}_{j=1}^N$ which approximates $P(Z_{t+1}|\mathbf{y}^{t+1})$.

The following section shows how to perform on-line inference for a process convolution GP model based on the particle learning procedure.

4 Particle learning for process convolution GPs

The default construction of the process convolution GP model is static in the sense that there is no time component involved. To allow sequential inference, the variable t is used to denote the sequential ordering of the data and sufficient information. The data are assumed to be sequentially independent. As shown in the previous sections, definitions of the sufficient information Z_t and predictive distribution $P(\mathbf{y}^{t+1}|Z_t)$ are required for the application of particle learning. The predictive distribution of model (1) can be obtained in closed form:

$$\begin{aligned} & P(\mathbf{y}_{t+1}|a_{y,t}, b_{y,t}, \boldsymbol{\beta}_t, \mathbf{C}_t, \mathbf{x}) \\ \equiv \; & \mathscr{T}_{n_{t+1}}\Big(2a_{y,t}\,,\, \mathbf{F}_{t+1}\boldsymbol{\beta}_t + \mathbf{K}_{t+1}\mathbf{x}\,,\, \frac{b_{y,t}}{a_{y,t}}\Big(\mathbf{I}_{n_t} + \mathbf{F}_{t+1}{\mathbf{C}_t}^{-1}\mathbf{F}_{t+1}^{\top}\Big)\Big), \end{aligned} \tag{2}$$

where $\mathscr{T}$ denotes the Multivariate Student's t distribution, and $\mathbf{y}_{t+1}$ denotes the $(n_{t+1} \times 1)$ data vector at time $t+1$. Note that all parameters except $\mathbf{x}$ have been integrated out and only the sufficient statistics $\{a_{y,t}, b_{y,t}, \boldsymbol{\beta}_t, \mathbf{C}_t\}$ are needed to compute the predictive probability given a new data point/set $\{\mathbf{y}_{t+1}, \mathbf{F}_{t+1}, \mathbf{K}_{t+1}\}$. These sufficient statistics are based on the complete conditional distributions. Therefore, the sufficient information Z_t would contain $\{a_{y,t}, b_{y,t}, \boldsymbol{\beta}_t, \mathbf{C}_t\}$. Moreover, $\mathbf{x}$ is needed to evaluate the predictive density, thus it is also stored into the sufficient information Z_t. If one is interested in $\{\lambda, \phi, \boldsymbol{\beta}\}$, they can also be kept, namely, $Z_t = \{a_{y,t}, b_{y,t}, \boldsymbol{\beta}_t, \mathbf{C}_t, \mathbf{x}, \lambda, \phi, \boldsymbol{\beta}\}$. Assuming that the initial priors are given by

$$\begin{aligned} \boldsymbol{\beta}|\phi &\sim N_{p+1}(\boldsymbol{\beta}_0, (\phi\mathbf{C}_0)^{-1}), \\ \phi &\sim G(a_{y,0}, b_{y,0}), \\ \mathbf{x}|\lambda &\sim N_m(\mathbf{0}, \lambda^{-1}\mathbf{I}_m), \\ \lambda &\sim G(a_{x,0}, b_{x,0}), \end{aligned}$$

the complete conditionals at time t can be found as

$$\begin{aligned} \boldsymbol{\beta}|\phi &\sim N_{p+1}(\boldsymbol{\beta}_t\,,\, (\phi\mathbf{C}_t)^{-1}), \\ \phi &\sim G(a_{y,t}, b_{y,t}), \\ \mathbf{x}|\lambda &\sim N_m(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \\ \lambda &\sim G(a_{x,t}, b_{x,t}), \end{aligned}$$

where

$$\boldsymbol{\beta}_t = \mathbf{C}_t^{-1}(\mathbf{F}_t{}^\top \mathbf{v}_t + \mathbf{C}_{t-1}\boldsymbol{\beta}_{t-1}), \quad \mathbf{C}_t = \mathbf{F}_t{}^\top \mathbf{F}_t + \mathbf{C}_{t-1},$$
$$a_{x,t} = m/2 + a_{x,0}, \quad b_{x,t} = 0.5\mathbf{x}^\top \mathbf{x} + b_{x,0}, \quad a_{y,t} = n_t/2 + a_{y,t-1},$$
$$b_{y,t} = b_{y,t-1} + \frac{1}{2}(s_t^2 + (\boldsymbol{\beta}_{t-1} - \hat{\boldsymbol{\beta}}_t)^\top (\mathbf{C}_{t-1}^{-1} + (\mathbf{F}_t{}^\top \mathbf{F}_t)^{-1})^{-1}(\boldsymbol{\beta}_{t-1} - \hat{\boldsymbol{\beta}}_t)),$$
$$\boldsymbol{\mu}_t = (\phi \mathbf{K}^t{}^\top \mathbf{K}^t + \lambda \mathbf{I}_m)^{-1} \phi \mathbf{K}^t{}^\top (\mathbf{y}^t - \mathbf{F}^t \boldsymbol{\beta}), \quad \boldsymbol{\Sigma}_t = (\phi \mathbf{K}^t{}^\top \mathbf{K}^t + \lambda \mathbf{I}_m)^{-1},$$
$$\hat{\boldsymbol{\beta}}_t = (\mathbf{F}_t{}^\top \mathbf{F}_t)^{-1} \mathbf{F}_t{}^\top \mathbf{v}_t, \quad \mathbf{v}_t = \mathbf{y}_t - \mathbf{K}_t \mathbf{x}, \quad s_t^2 = (\mathbf{v}_t - \mathbf{F}_t \hat{\boldsymbol{\beta}}_t)^\top (\mathbf{v}_t - \mathbf{F}_t \hat{\boldsymbol{\beta}}_t).$$

Here, n^t denotes the total number of observations up to time t, $\mathbf{y}^t = (\mathbf{y}^{t-1}{}^\top, \mathbf{y}_t^\top)^\top$, $\mathbf{F}^t = (\mathbf{F}^{t-1}{}^\top, \mathbf{F}_t^\top)^\top$, and $\mathbf{K}^t = (\mathbf{K}^{t-1}{}^\top, \mathbf{K}_t^\top)^\top$ denote the $(n^t \times 1)$ data vector, $(n^t \times (p+1))$ design matrix, and $(n^t \times m)$ kernel matrix, respectively. In the propagate step, the resampled sufficient information Z_t is updated to account for the new data $\{\mathbf{y}_{t+1}, \mathbf{F}_{t+1}\}$. Specifically, the sufficient statistics are updated deterministically based on the above equations, but for time $t+1$. Once the sufficient statistics are updated, then the parameters are sampled from their conditionals since they are needed to compute the predictive density for the next data arrival. Following is a rough sketch of the resampling and propagation procedures:

1. Upon arrival of $\mathbf{y}_{t+1}$, sample indices $\{\zeta(j) : j = 1, \cdots, N\}$ with replacement from a Multinomial distribution with weights proportional to the predictive distribution (2), i.e., $P(\zeta(j) = i) \propto P(\mathbf{y}_{t+1} \,|\, Z_t^{(i)})$ for $i = 1, \cdots, N$. Set $\{Z_t^{(j)}\}_{j=1}^N = \{Z_t^{\zeta(j)}\}_{j=1}^N$.

2. For $j = 1, \cdots, N$, update the sufficient statistics to obtain $a_{x,t+1}^{(j)}, b_{x,t+1}^{(j)}$, $a_{y,t+1}^{(j)}, b_{y,t+1}^{(j)}$, $\boldsymbol{\beta}_{t+1}^{(j)}$, $\mathbf{C}_{t+1}^{(j)}, \boldsymbol{\mu}_{t+1}^{(j)}$, and $\boldsymbol{\Sigma}_{t+1}^{(j)}$, then sample

$$\begin{aligned}
\lambda^{(j)} &\sim G(a_{x,t+1}^{(j)}, b_{x,t+1}^{(j)}), \\
\phi^{(j)} &\sim G(a_{y,t+1}^{(j)}, b_{y,t+1}^{(j)}), \\
\boldsymbol{\beta}^{(j)} &\sim N_{p+1}(\boldsymbol{\beta}_{t+1}^{(j)}, (\phi \mathbf{C}_{t+1}^{(j)})^{-1}), \\
\mathbf{x}^{(j)} &\sim N_m(\boldsymbol{\mu}_{t+1}^{(j)}, \boldsymbol{\Sigma}_{t+1}^{(j)}).
\end{aligned}$$

The following section applies our methodology to two illustrative examples. For convenience, the sequential process convolution GP model will be abbreviated by SPCGP. Comparisons with the standard process convolution GP model (PCGP), the sequential GP with standard specification (PLGP) by Gramacy and Polson (2009), and the standard GP with MCMC, are also given. PLGP

is provided by the *plgp* R package (Gramacy, 2010), and the standard GP with MCMC is provided by the *tgp* R package (Gramacy, 2007).

5 Examples

5.1 1-d Synthetic Sinusoidal Data

One hundred data points are generated by sampling from the 1-d response below,

$$z(s) = \sin\left(\frac{\pi s}{5}\right) + 0.2\cos\left(\frac{4\pi s}{5}\right), \quad 0 \le s \le 9.6,$$

and adding $N(0, sd = 0.1^2)$ noise to the sampled points (shown in Figure 1). One data point is

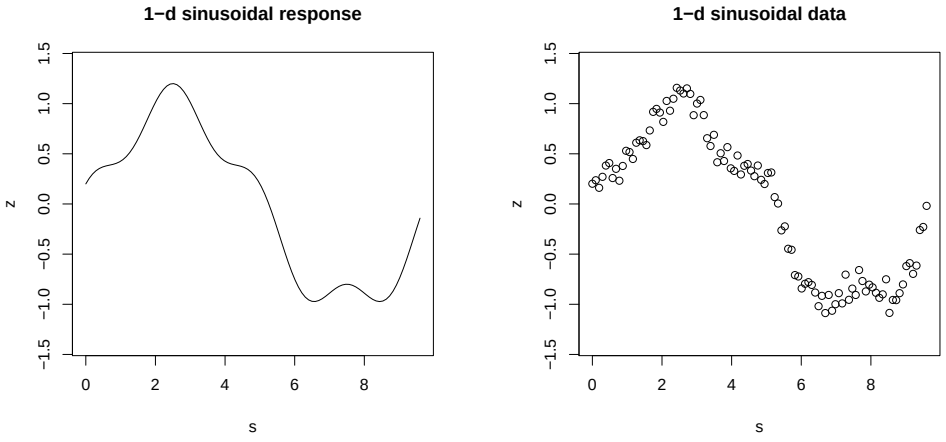

Figure 1: 1-d synthetic sinusoidal response (*left*) and data (*right*)

randomly selected without replacement at each time step as an input to the model. The initial priors are given by

$$\beta \sim N(0,\ (10\phi)^{-1}), \quad \phi \sim G(a_y = 1,\ 0.001),$$
$$\mathbf{x}|\lambda \sim N_m(\mathbf{0},\ (\lambda \mathbf{I}_m)^{-1}), \quad \lambda \sim G(a_x = 1,\ 0.001).$$

Note that β is a scalar which corresponds to the intercept, and the linear component is not included. The initial priors for β, ϕ and λ are fairly non-informative so that the modeling results would depend largely on the data. A Gaussian kernel with 30 bases is applied and the standard deviation of the kernel is set equal to the spacing between adjacent basis points. A total of 500 particles are used

in the simulation. The posterior predictive mean surfaces with the corresponding 90% interval are shown in Figure 2 for $t = \{5, 10, 20, 30, 40, 50\}$.

When only a few data points are available, uncertainty at unobserved locations is reflected through having a larger 90% posterior predictive interval, as opposed to a relatively smaller interval at the observed locations. As more data points arrive and spread over the entire domain, the model quickly improves and is able to obtain a mean surface at $t = 50$ that has most of the features of the true response. The posterior predictive summary for $t = 100$ is shown in the top left panel of Figure 3, along with the corresponding 500 particles on the right. The mean surface resembles the true response, and the 90% interval well captures the data variability. Sensitivity of the model against the number of particles is illustrated in the middle and bottom panels, which are based on 100 and 20 particles, respectively. Note that SPCGP is fairly robust in the sense that even with very few particles such as 20, the result is comparable to that of using 500 particles. Figure 4 displays results from a SPCGP (*top left*), PLGP (*bottom left*), PCGP (*top right*), and a standard GP with MCMC (*bottom right*). Results from both sequential approaches are displayed for $t = 100$ with 500 particles. The MCMC approaches are based on a total of 600 iterations with a burn-in of 100 so that the number of samples is 500. Note that the posterior predictive summaries from all models are quite similar except for PLGP. It produces a mean surface that over smoothes the data and fails to capture the local features in the true response. The resulting 90% posterior predictive interval is an over estimate of the true data variability. The misfitting might be due to the default prior parameters in the *plgp* R package. In contrast, SPCGP is able to capture the true response, both globally and locally, and also has a more reasonable prediction for the data variability.

5.2 The Pump-and-Treat problem

The Pump-and-Treat problem (Matott et al., 2011) involves a groundwater contamination scenario based on the Lockwood Solvent Groundwater Plume Site located near Billings, Montana. Two plumes (A and B) containing chlorinated solvents were developed due to industrial practices near the Yellowstone river as shown in Figure 5. Of interest is plume A located in the southern section of the site. The primary concern is to prevent the plume from migrating to and contaminating the Yellowstone river. The proposed remediation involves drilling two pump-and-treat wells. This problem has been modeled using a computer simulator where the inputs are pumping rates for the

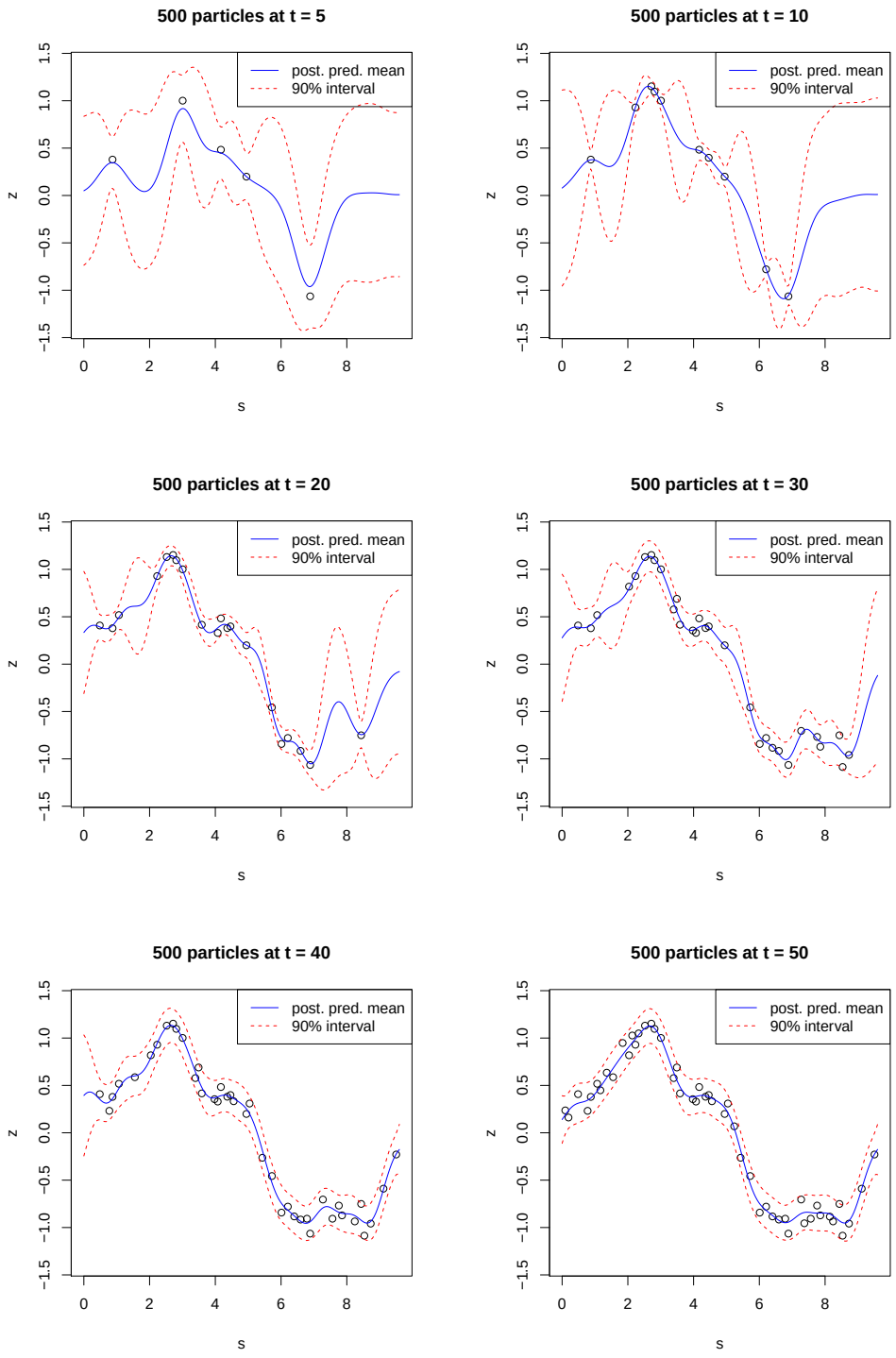

Figure 2: Posterior predictive summary from SPCGP for $t = \{5, 10, 20, 30, 40, 50\}$

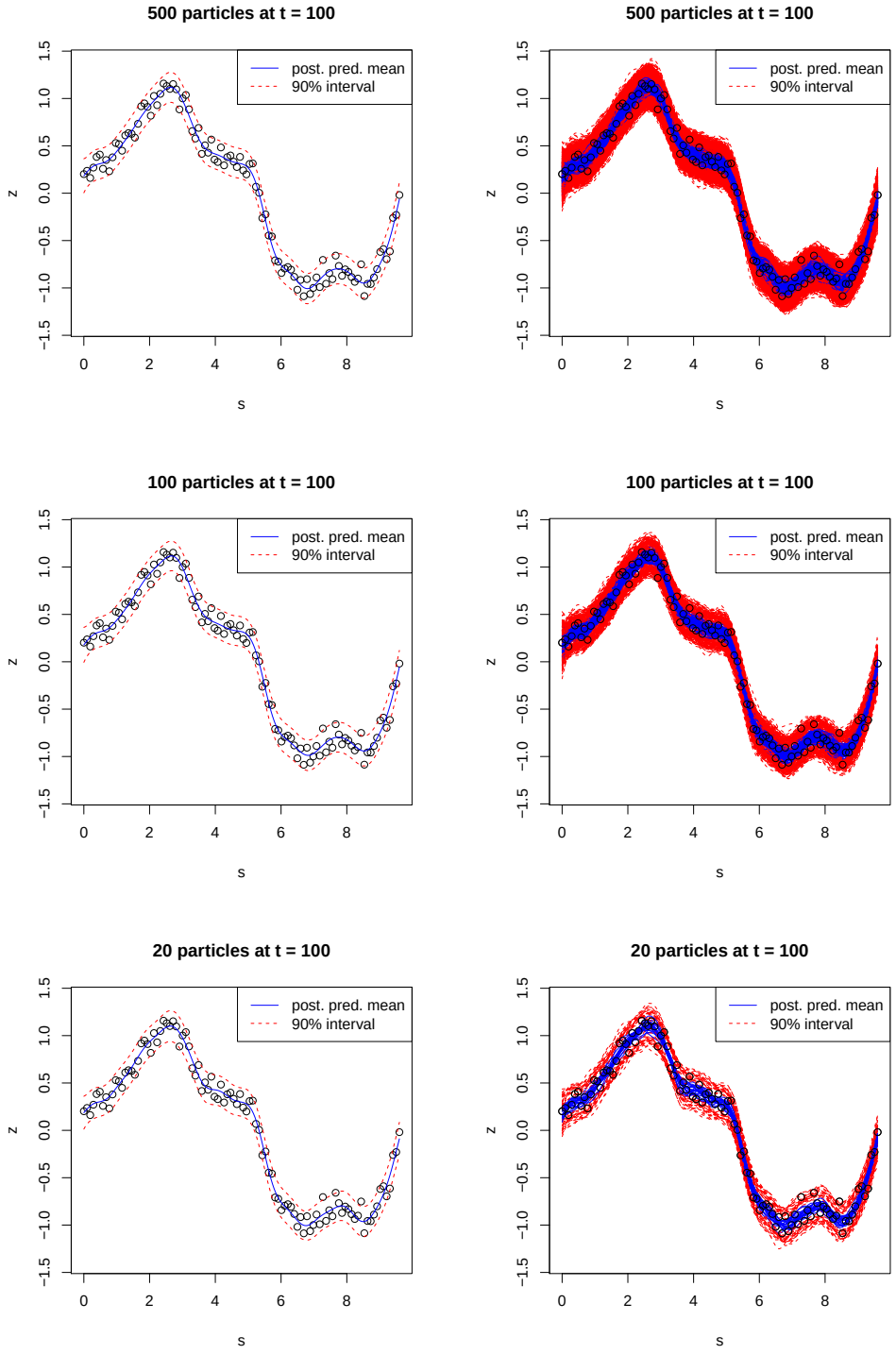

Figure 3: Posterior predictive summary from SPCGP for $t = 100$ with $500, 100$, and 20 particles

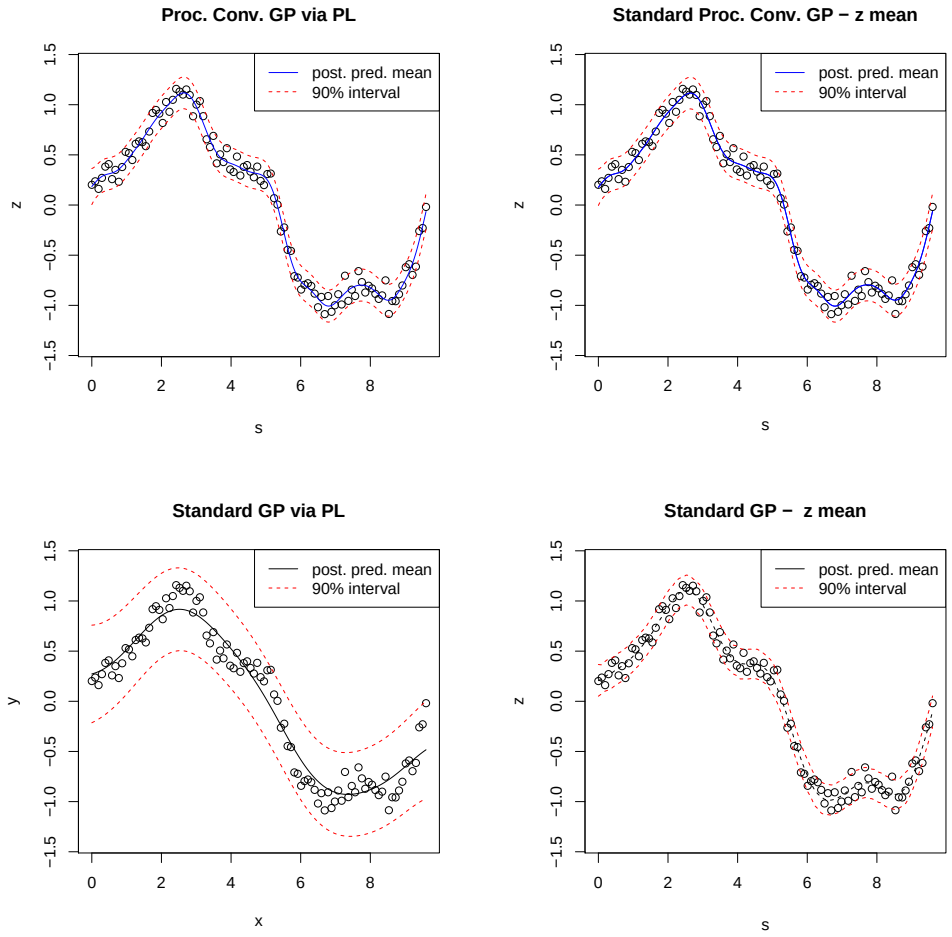

Figure 4: Posterior predictive summary based on SPCGP (*top left*), PCGP (*top right*), PLGP (*bottom left*), and standard GP (*bottom right*)

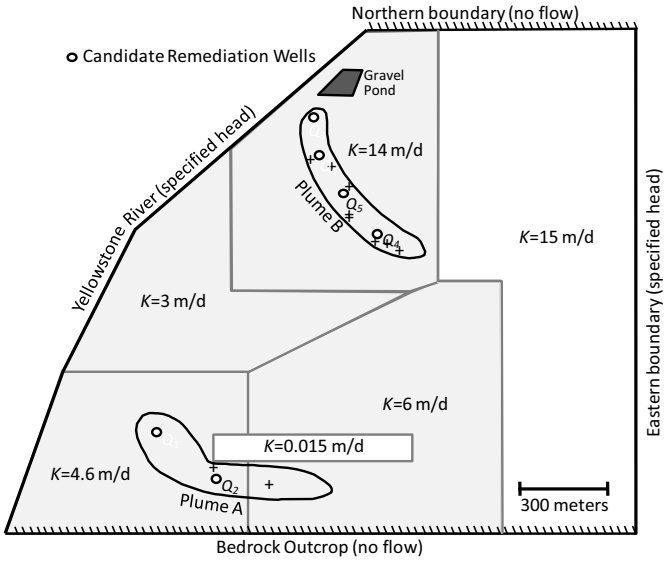

Figure 5: Lockwood Solvent Groundwater Plume Site located near Billings, Montana

two pump-and-treat wells, and the output is a cost function which combines the financial cost of running the wells with a large penalty for any contamination of the river (the penalty ensures that any optimal solution will not allow any contamination of the river). The two pumping rates can be set between 0 and 20,000. The objective is to minimize the cost function, i.e., the expense of running the wells. Because of the non-trivial time for each simulator run, it is not possible to run the simulator at every possible combination of inputs and find the one that has the minimum cost. Instead, a computer simulation experiment approach (Sacks et al., 1989) is taken to sequentially build a surrogate model while searching for the minimum of the surface (Jones et al., 1998; Taddy et al., 2009; Gramacy and Lee, 2011). This method proceeds sequentially by adding new design points (a pair of pump rates and the associated cost) one-by-one based on some criterion and update the model fit conditional on the new design point. Updating of the model fit could be done with MCMC, however, it could be computationally demanding since the MCMC has to be repeated for every new design point. Instead, SPCGP is applied to this problem. The model is setup by specifying a (25×25) basis grid, and a Bézier kernel (Lemos and Sansó, 2009) whose circular support has a radius of three times the spacing of any two adjacent bases. A Bézier kernel has a compact support such that it makes the kernel matrix $\mathbf{K}$ sparse by assuming that spatial

points far apart have negligible correlation. This allows dedicated matrix decomposition routines to speed up the computation. Setting the kernel radius to be three times the spacing of adjacent bases aligns with the rule of thumb given by Higdon (2005) for Gaussian kernels since the shape of the employed Bézier kernel resembles a Gaussian. To choose the new data point (simulator run), the expected improvement (EI) approach (Jones et al., 1998) is employed by choosing the point $\mathbf{s}$ that maximizes

$$E[I^g(\mathbf{s})] = E[max(f_{best} - f(\mathbf{s}))^g, 0],$$

where f_{best} denotes the current best point (inputs with minimum response) and $f(\mathbf{s})$ denotes the predicted output response (from the current state of SPCGP) at input $\mathbf{s}$. The power g can be specified to tune the local versus global character of the optimization. For example, $g = 1$ yields the standard expected improvement statistic, while for $g = 2$, $E[I^2(\mathbf{s})] = var[I(\mathbf{s})] + E[I(\mathbf{s})]^2$ explicitly rewards the improvement in variance and thus gives relatively more weight toward global exploration of the response surface. Since finding the maximizing $\mathbf{s}$ exactly would be another difficult problem, optimization is approximated by considering 200 candidate points in the input space generated using Latin hypercube sampling (McKay et al., 1979) and then choosing the candidate point with largest expected improvement (Gramacy and Lee, 2009). The initial priors are specified as

$$\begin{aligned}
&\beta \sim N(0,\ \phi^{-1}), \quad \phi \sim G(1,\ 0.0001), \\
&\mathbf{x}|\lambda \sim N_m(\mathbf{0},\ (\lambda \mathbf{I}_m)^{-1}), \quad \lambda \sim G(1,\ 0.001),
\end{aligned}$$

where β denotes the mean level (intercept). A narrow $G(1,\ 0.0001)$ prior is imposed on ϕ because the simulator is approximating a deterministic process, however, it makes sense to leave some room for any error that might result, and this error is expected to be relatively small. A total of 60 input points are considered, where the first 30 are generated from a Latin hypercube to build an initial model surface, and the remaining 30 are chosen sequentially one-by-one using the EI approach described above. A total of 500 particles are used for the simulation.

Fixing $g = 1$, the posterior predictive mean surfaces from SPCGP are shown in Figure 6. For $t = \{6, 9, 15\}$, the mean surfaces illustrate the intermediate results during the process of generating the initial model surface for $t = 30$. Starting from $t = 30$, the EI algorithm is deployed and it is apparent that most of the design points considered are near the minimum of the mean surface.

This is expected because the goal of this approach is to explore the input space in areas that are likely to provide the minimum response. The posterior predictive summary from SPCGP and PLGP at $t = 60$ are shown in the top and bottom panels of Figure 7, respectively. The middle panel displays results from repeatedly applying PCGP with 600 MCMC iterations (100 burn-in and 500 samples) to each newly generated design point. The mean surface from these models are resembling in general, with SPCGP and PCGP predicting more local features than PLGP. The 90% interval width of both SPCGP and PCGP have low uncertainty at the data locations and high uncertainty at the unobserved locations, whereas that of PLGP seem to overestimate/underestimate the uncertainty at the observed/unobserved locations. Nonetheless, the locations of the predicted minimums found by these models are comparable to one another. Figure 8 displays the posterior predictive mean surfaces under 500 (*top*), 100 (*middle*), and 20 (*bottom*) particles with $g = 1$ (*left*) and $g = 2$ (*right*). Results that have the same g value are closely resembling. For different g values, the difference comes from the fact that the $g = 2$ case tends to fit a better overall surface than the $g = 1$ case by exploring places with high uncertainty. Nonetheless, the locations of minimums are similar for all, even with very few particles such as 20. Since finding the minimum location is the main purpose of this problem, all models and settings considered here have comparable performance in this regard. All simulations are run on an Intel Core 2 duo CPU at 2.4 Ghz with 4 Gb of RAM. Table 1 and 2 display the running times of SPCGP, PCGP, and PLGP v.s. the number of LHS candidates (200, 500, and 1000) with 500 particles (or MCMC samples) at $g = 1$. The left panel displays the average updating time (model update and prediction on LHS candidates, but not including simulator time) for each design point, and the right panel shows the total running time for the whole process. For 200 LHS candidates, the average updating time for SPCGP is 15.78 seconds, and that of PCGP is 21.09 seconds. The difference is not too big and mostly due to the extra 100 burn-in iterations in PCGP. However, this difference could be exacerbated in more complicated datasets, where MCMC might need a larger number of burn-in steps to reach equilibrium, and also require thinning in order to obtain less correlated samples. These mechanisms can greatly increase the computational time of MCMC. In contrast, such difficulties generally do not exist in PL. In addition, PL (or generally, SMC) is completely parallelizable in the propagation step since the particles can be updated independently of one another up to having a unique computing node for each particle. In the case of PLGP, the average updating time and total running time are not

too much larger than those of SPCGP for 200 LHS candidates. However, as the number of LHS candidates increases, PLGP significantly slows down, while SPCGP maintains roughly the same speed. This suggests that for more complicated problems where a higher number of LHS candidates are needed, SPCGP is computationally more efficient than PLGP (for low dimensional problems) in its current implementation.

Table 1: Average updating time (seconds)

	Number of LHS candidates		
Model	200	500	1000
SPCGP	15.78	16.09	16.10
PCGP	21.16	22.15	21.72
PLGP	19.28	96.96	263.02

Table 2: Total running time (seconds)

	Number of LHS candidates		
Model	200	500	1000
SPCGP	595.73	610.96	618.69
PCGP	759.27	792.964	789.89
PLGP	700.90	3030.97	8012.80

6 Discussion and Conclusion

On-line inference for a GP model based on MCMC is inefficient because it requires re-running the MCMC for every new data arrival, which can be computationally demanding due to slow convergence. In this paper, a sequential inference approach for the process convolution GP model is developed, which is based on a method called Particle Learning. It allows parameter inference to be performed on-line for each new batch of data without having to use MCMC. This convolution approach allows for the handling of much larger datasets than would not be computationally feasible under the standard GP approach. This is because the computational expense is tied to the number of basis points instead of the number of datapoints, although the convolution approach is only practical for lower-dimensional problems because of the need to create a grid of basis points. Another advantage of SMC methods is that they are completely parallelizable in the propagation step - the particles can be updated independently of one another. In contrast, MCMC has to be done, to a large extent, in serial. Illustrations of SPCGP on a 1-d synthetic dataset and a 2-d optimization problem show promising results in terms of model fitting and computational speed, robustness in optimization, as well as robustness with respect to the number of particles required.

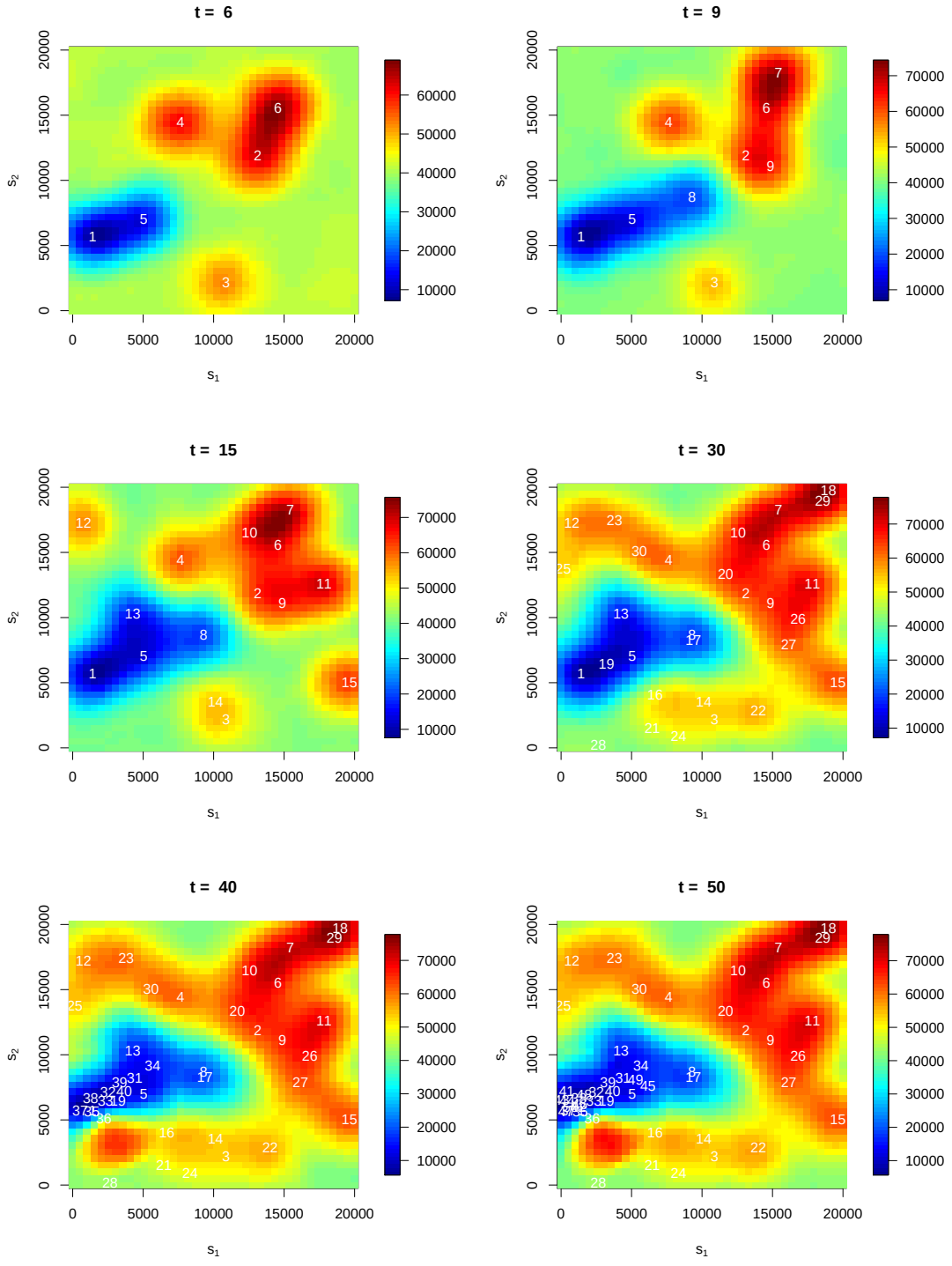

Figure 6: Posterior predictive mean from SPCGP with $g = 1$ for $t = \{6, 9, 15, 30, 40, 50\}$ for the Pump-and-Treat problem

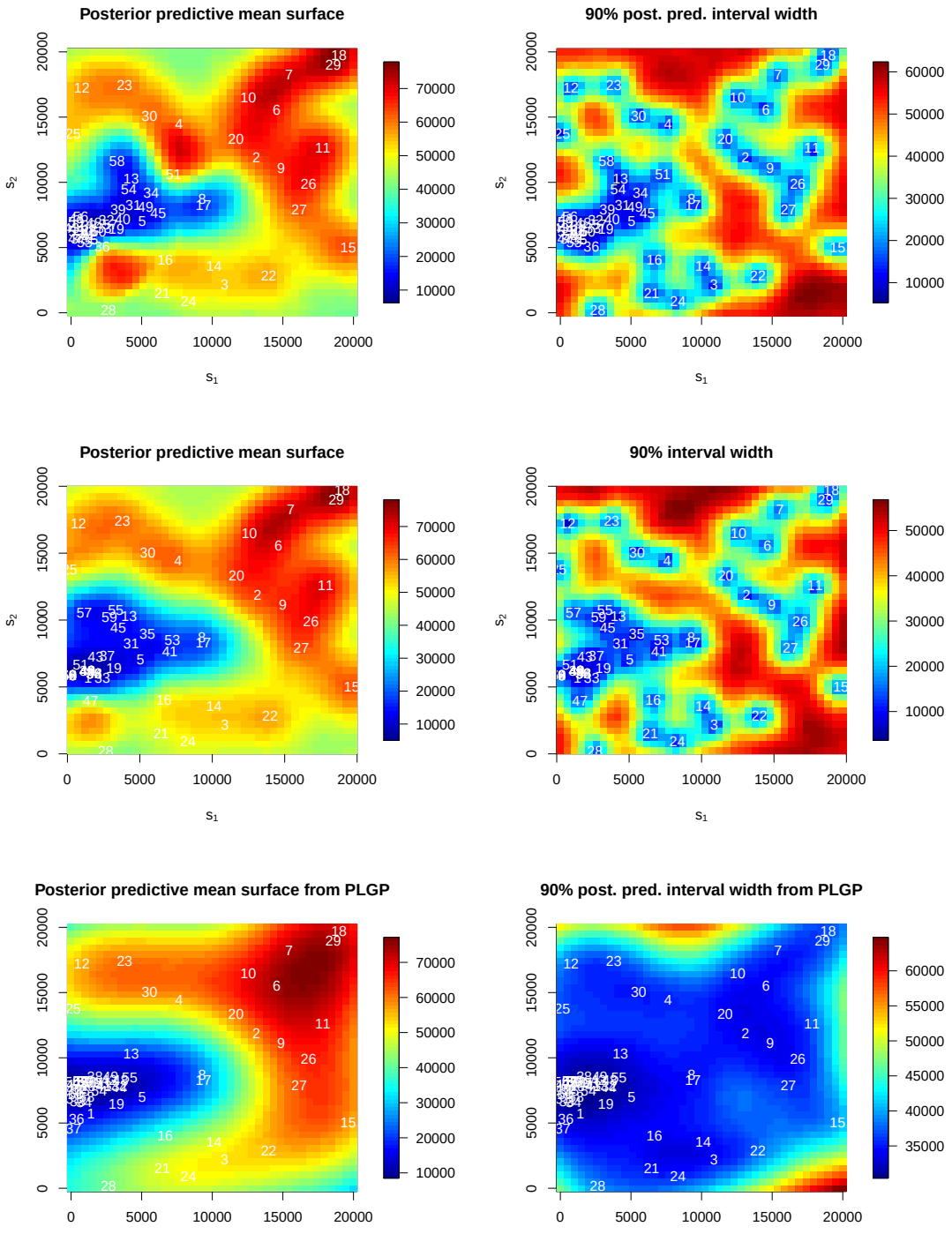

Figure 7: Posterior predictive summary from SPCGP (*top*), PCGP (*middle*), and PLGP (*bottom*) with $g = 1$ at $t = 60$ for the Pump-and-Treat problem

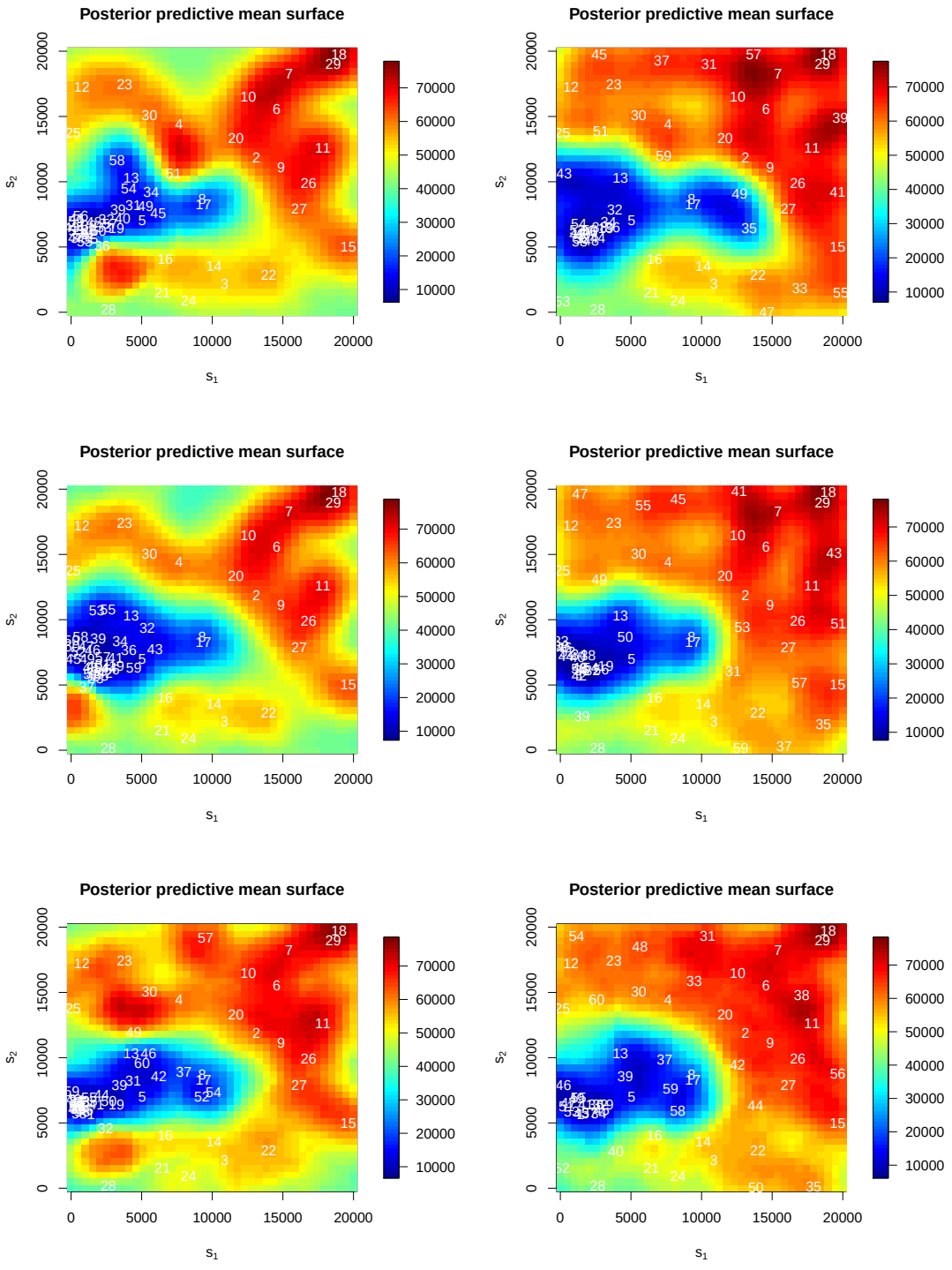

Figure 8: Posterior predictive mean from SPCGP based on 500 (*top*), 100 (*middle*), and 20 (*bottom*) particles for $g = 1$ (*left*) and $g = 2$ (*right*) at $t = 60$ for the Pump-and-Treat problem

References

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2003), *Hierarchical Modeling and Analysis for Spatial Data*, Chapman & Hall, Boca Raton, FL.

Calder, C. A., Holloman, C., and Higdon, D. (2002), "Exploring Space-Time Structure in Ozone Concentration Using a Dynamic Process Convolution Model," *Case Studies in Bayesian Statistics*, 6, 165–176.

Carvalho, C. M., Johannes, M., Lopes, H. F., and Polson, N. G. (2010), "Particle Learning and Smoothing," *Statistical Science*, 25(1), 88–106.

Cressie, N. (1991), *Statistics for Spatial Data*, John Wiley and Sons, Inc.

Gordon, N., Salmond, D., and Smith, A. (1993), "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEEE Proceedings*, F140, 107–113.

Gramacy, R. B. (2007), "tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models," *Journal of Statistical Software*, 19, 1–46.

Gramacy, R. B. (2010), "R Package plgp: Particle Learning of Gaussian Processes," .

Gramacy, R. B. and Lee, H. K. (2009), "Adaptive Design and Analyis of Supercomputer Experiments," *Technometrics*, 51, 130–145.

Gramacy, R. B. and Lee, H. K. H. (2011), "Optimization under unknown constraints," in *Bayesian Statistics 9*, eds. J. Bernardo, S. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, pp. 229–256, Oxford University Press.

Gramacy, R. B. and Polson, N. G. (2009), "Particle Learning of Gaussian Process Models for Sequential Design and Optimization," *Journal of Computational and Graphical Statistics*, 20, 102–118.

Higdon, D. (1998), "A process-convolution approach to modeling temperatures in the North Atlantic Ocean," *Journal of Environmental and Ecological Statistics*, 5(2), 173–190.

Higdon, D. (2002), "Space and space-time modeling using process convolutions," in *Quantitative Methods for Current Environmental Issues*, eds. C. Anderson, V. Barnett, P. Chatwin, and A. El-Shaarawi, pp. 37–54, London, Springer.

Higdon, D. (2005), "A Primer on Space-time Modeling from a Bayesian Perspective," Tech. Rep. LA-UR-05-3097, Statistical Sciences Group, Los Alamos National Laboratory.

Jones, D., Schonlau, M., and Welch, W. (1998), "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, 13, 455–492.

Kalman, R. E. (1960), "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, 82, 35–45.

Lemos, R. T. and Sansó, B. (2009), "Spatio-Temporal Model for Mean, Anomaly and Trend Fields of North Atlantic Sea Surface Temperature," *Journal of the American Statistical Association*, 104, 5–18.

Liu, J. and West, M. (2001), "Combined parameters and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice (Eds. A. Doucet, N. de Freitas and N. Gordon)*, pp. 197–223, Springer-Verlag, New York.

Matott, L. S., Leung, K., and Sim, J. (2011), "Application of Matlab and Python Optimizers to Two Case-Studies Involving Groundwater Flow and Contaminant Transport Modeling," *Geospatial Cyberinfrastructure for Polar Research*, 37 (11), 1894–1899.

McKay, M. D., Conover, W. J., and Beckman, R. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239–245.

Neal, R. M. (1997), "Monte Carlo implementation of Gaussian process for Bayesian regression and classification," Tech. rep., Dept. of statistics, University of Toronto.

Neal, R. M. (1998), "Regression and classification using Gaussian process priors (with discussion)," *Bayesian statistics*, 6, 476–501.

Paciorek, C. and Schervish, M. J. (2006), "Spatial Modelling Using a New Class of Nonstationary Covariance Functions," *Environmetrics*, 17, 483–506.

Pitt, M. and Shephard, N. (1999), "Filtering via simulation: auxiliary particle lters," *Journal of the American Statistical Association*, 94, 590–599.

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

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–435.

Storvik, G. (2002), "Particle lters in state space models with the presence of unknown static parameters," *IEEE Transactions of Signal Processing*, 50, 281–289.

Taddy, M. A., Lee, H. K. H., Gray, G. A., and Griffin, J. D. (2009), "Bayesian Guided Pattern Search for Robust Local Optimization," *Technometrics*, 51 (4), 389–401.

West, M. and Harrison, J. (1997), *Bayesian Forecasting and Dynamic Models*, Springer.