Parallelizing MCMC for Bayesian Spatiotemporal Geostatistical Models

Jun Yan[1], **Mary Kathryn Cowles**[1], **Shaowen Wang**[2,3], **and Marc P. Armstrong**[3,4]

[1]*Department of Statistics and Actuarial Science, The University of Iowa*

[2]*Academic Technologies – Research Services, The University of Iowa*

[3]*Department of Geography, The University of Iowa*

[4]*Program in Applied Mathematical and Computational Science, The University of Iowa*

October 24, 2006

Abstract

When MCMC methods for Bayesian spatiotemporal modeling are applied to large geostatistical problems, challenges arise as a consequence of storage requirements, computing costs, and convergence monitoring. This article describes the parallelization of a reparametrized and marginalized posterior sampling (RAMPS) algorithm, which is carefully designed to generate posterior samples efficiently. The algorithm is implemented using the Parallel Linear Algebra Package (PLAPACK). The scalability of the algorithm is investigated via simulation experiments that are implemented using a cluster with 25 processors. The usefulness of the method is illustrated with an application to sulfur dioxide concentration data from the Air Quality System database of the U.S. Environmental Protection Agency.

Keywords: Bayesian inference, Markov chain Monte Carlo, parallel algorithm, spatial modeling

1 Introduction

Markov chain Monte Carlo (MCMC) is a very useful but computing intensive statistical method. MCMC samples from the commonly used Gibbs sampler or Metropolis-Hastings algorithm, however, are usually autocorrelated. As the autocorrelation increases, larger MCMC sample sizes are needed to make credible Bayesian inferences about the quantities of interest. The liability is worsened when large problems are addressed, since both the storage and computation requirements for each single MCMC iteration can make the use of a single processor machine infeasible. A geostatistical model with a dense spatiotemporal correlation matrix demands order n^2 in storage for n observations. Cholesky decomposition of such a matrix is of order n^3 in computation. This article addresses the feasibility of parallelizing a reparametrized and marginalized posterior sampling (RAMPS) algorithm, which is carefully designed to lead to lower autocorrelation in MCMC samples, in spatiotemporal modeling of large datasets.

As a motivating example, consider air pollution data from the Air Quality System (AQS) database of the U.S. Environmental Protection Agency (EPA). Monitoring stations forward hourly or daily measurements of pollutant concentration to EPA's database. EPA computes a yearly summary for each monitoring station. In this paper, we focus on an analysis of the annual mean of hourly measured concentration of sulfur dioxide (SO_2), one of the "criteria air pollutants" regulated by EPA. There are $n = 4711$ observations from 870 sites in the most recent 8 years (1998–2005). Figure 1 shows all the measurement sites in the coterminous U.S. The data are not balanced since not all sites have all 8 measurements (see Table 1). For such a large dataset, fitting a spatiotemporal model and monitoring the convergence of MCMC algorithms would be impractical for a single processor machine due to the order n^3 Cholesky decomposotions of covariance matrices. One way to avoid Cholesky decompositions of large matrices is to divide the whole area into subregions and assume independence across subregions conditional on random effects at the subregion level. This method has been used, for example, by Cowles et al. (2002) in the analysis of point referenced snow water equivalent data. The approach reported in this paper, however, does not require the assumption of conditional independence across subregions since it provides a direct solution by parallelizing an efficient MCMC algorithm.

[Figure 1 about here.]

[Table 1 about here.]

The possibility of parallel processing has long been recognized by statisticians in carrying out computing intensive methods (Sylwestrowicz, 1982; Schervish, 1988; Adams et al., 1996). For applications such as bootstrapping and kriging, algorithms can be "embarrassingly parallel" (Rossini et al., 2003), since no inter-processor communication is needed. For notoriously computing intensive MCMC methods, however, parallelization is complicated by dependence between successive steps of the Markov chains. The simplest scheme, which runs multiple chains in parallel, has a serious limitation in that each processor must spend a significant amount of burn-in time (Rosenthal, 2000). A non-trivial scheme parallelizes a single MCMC chain (Wilkinson, 2005). As Wilkinson points out, there are different issues related to the different schemes, and each is appropriate in different settings. The focus of this article is the latter scheme, parallelization of a single MCMC chain.

In the context of spatial statistical modeling, Whiley and Wilson (2004) propose various parallel algorithms for models with Gaussian spatial random effects. Single chain parallelization is done by partitioning the spatial domain and the implementation is done with a parallel linear algebra library ScaLAPACK (Blackford et al., 1997). The RAMPS algorithm parallelized in this article is more efficient than those discussed by Whiley and Wilson (2004). It is a two-block updating algorithm: the first block of parameters is updated from their marginal distributions using slice sampling (Neal, 2003); the second block is then updated by conditioning on the current values of the parameters in the first block. The parallelization is implemented using Parallel Linear Algebra package PLAPACK (van de Geijn, 1997). This methodology has advantages over that of Whiley and Wilson (2004) in that the posterior sample has lower autocorrelation and the implementation requires less programming effort.

The rest of this article is organized as follows. In Section 2, we start with a Gaussian geostatistical model which consists of three components: trend, spatial process, and measurement error. This model is generalized to incorporate the temporal dimension, forming a spatiotemporal model. We propose a reparametrization of the model to facilitate the

MCMC algorithm. In Section 3, we describe in detail the RAMPS algorithm and discuss the computational bottleneck of the algorithm for large datasets. We address this bottleneck by parallelizing the RAMPS algorithm with PLAPACK in Section 4. Timing performance results under various sample sizes and processor grid sizes are presented in Section 5. In Section 6, we illustrate the method with a spatiotemporal model for the sulfur dioxide concentration data. A brief discussion concludes in Section 7.

2 A Bayesian Spatiotemporal Geostatistical Model

Consider a subset D of the r-dimensional Euclidean space $\mathcal{R}^r$, with positive Euclidean volume. The dimension r in most spatial data is 2. Let s be a spatial location and $Y(s)$ be a random variable measured at s. A random field (or stochastic process) $\{Y(s) : s \in D\}$, is formed when the spatial index s varies continuously over D. In this article, we focus on the case of a Gaussian random field. Geostatistical data are a realization of the random field observed at fixed locations:

$$\{Y(s_i) : s_i \in D, \quad i = 1, \ldots, n\}. \tag{1}$$

Geostatistical models provide a natural and interpretable way to model spatial data measured at irregularly-spaced point sites. Suppose that for $Y(s_i)$, a $p \times 1$ covariate vector $X(s_i)$ can be used to model the large-scale variation, or spatial trend, of $Y(s_i)$. Let $Y = \{Y(s_1), \ldots, Y(s_n)\}^\top$ and $X = \{X^\top(s_1), \ldots, X^\top(s_n)\}^\top$. A Gaussian geostatistical model for Y consists of spatial trend, spatial correlation, and measurement error:

$$\begin{aligned} &Y = X\beta + Z + \varepsilon, \\ &Z \sim N\left(0, \sigma_z^2 \Omega(\phi)\right), \\ &\varepsilon \sim N(0, \sigma_e^2 I). \end{aligned} \tag{2}$$

where β is a $p \times 1$ vector of covariate coefficients, Z is a $n \times 1$ vector capturing the spatial correlation, and ε is a $n \times 1$ vector of independent and identically distributed measurement errors. The distribution of Z is multivariate normal with mean zero and covariance matrix $\sigma_z^2\Omega(\phi)$, where $\Omega(\phi)$ is the correlation matrix as a function of parameter ϕ. In general, the correlation of $Z(s_i)$ and $Z(s_j)$ is modeled as a function of the distance, and possibly orientation, between sites s_i and s_j. When the correlation only depends on distance d between sites, the correlation is isotropic. Examples of parametric isotropic correlation functions are the exponential, spherical, and Matérn classes (see, for example, Banerjee et al., 2004, Table 2.1).

To facilitate the MCMC algorithm in the next section, we reparametrize the variance components of model (2). Note that model (2) can be written as

$$Y \sim N\left(X\beta, \sigma_z^2 \Omega(\phi) + \sigma_e^2 I\right). \tag{3}$$

Let $\sigma^2 = \sigma_s^2 + \sigma_e^2$ and $\kappa = \sigma_e^2/\sigma^2$. The reparametrized model is then

$$Y \sim N\left(X\beta, \sigma^2[(1-\kappa)\Omega(\phi) + \kappa I]\right). \tag{4}$$

The ratio κ is interpreted as the fraction in the total variation of Y contributed by the measurement error. It is the "shrinkage factor" used in uniform shrinkage priors on variances in hierarchical models (Christiansen and Morris, 1997; Daniels, 1999). This reparametrization leads to a bounded support $(0, 1)$ for κ, in contrast to the form used by Diggle and Ribeiro (2002)

$$Y \sim N\left(X\beta, \sigma_z^2[\Omega(\phi) + \lambda I]\right), \tag{5}$$

where $\lambda = \sigma_e^2/\sigma_z^2$ has unbounded support.

Model (4) can be generalized to incorporate a temporal component. Let $Y = \{Y(s_i, t_i) : s_i \in D,\ t_i \in E, i = 1,\ \ldots, n\}$, where E is a collection of time points. An extension of model (4) is simply

$$Y \sim N\left(X\beta, \sigma^2[(1-\kappa)\Omega(\phi, \rho) + \kappa I]\right), \tag{6}$$

where $\Omega(\phi, \rho)$ is a function of spatial correlation parameter ϕ and temporal correlation parameter ρ. A simple form for $\Omega(\phi, \rho)$ is a separable structure of spatial correlation and temporal correlation. When the data are balanced — that is, all sites have the same number of temporal points — the correlation matrix $\Omega(\phi, \rho)$ is $\Omega_S(\phi) \otimes \Omega_T(\rho)$, where Ω_S and Ω_T are, respectively, the spatial and the temporal correlation matrix, and $\otimes$ represents the Kronecker product. A widely used temporal correlation matrix uses the AR(1) structure. When a site has missing observations for some temporal points, which is often the case in practice, we need to extract the right rows and columns of $\Omega_S(\phi) \otimes \Omega_T(\rho)$ to form the correlation matrix $\Omega(\phi, \rho)$.

Prior distributions on all unknown model parameters $\theta = (\phi, \rho, \kappa, \sigma^2, \beta)$ are required to complete the Bayesian model. We first consider the priors of (σ^2, κ), which are reparametrized from variance parameters (σ_z^2, σ_e^2). It is common practice in Bayesian geostatistical modeling to place semi-conjugate inverse gamma priors on both σ_z^2 and σ_e^2. If independent inverse gamma priors are placed on σ_e^2 and σ_z^2 — that is, if

$$\begin{aligned} \sigma_z^2 &\sim \mathrm{IG}(a_z,\ b_z), \\ \sigma_e^2 &\sim \mathrm{IG}(a_e,\ b_e), \end{aligned}$$

then standard multivariate change-of-variable methods can be used to show that the joint prior distribution induced on (σ^2, κ) in our model parameterization is characterized by a marginal density f of κ,

$$f(\kappa; a_z, b_z, a_e, b_e) = \frac{\Gamma(a_e + a_z) b_e^{a_e} b_z^{a_z}}{\Gamma(a_e)\Gamma(a_z)} \frac{\kappa^{a_z - 1}(1-\kappa)^{a_e - 1}}{[b_z \kappa + b_e(1-\kappa)]^{a_e + a_z}}, \qquad \kappa \in (0,1), \tag{7}$$

and the conditional density of σ^2 given κ,

$$\sigma^2 | \kappa \sim \mathrm{IG}\left(\alpha_e + \alpha_z,\ \frac{b_z}{1-\kappa} + \frac{b_e}{\kappa}\right). \tag{8}$$

When $b_z = b_e$, the density of κ in (7) is a Beta density with shape parameters a_z and a_e.

The semi-conjugate prior for the vector of regression parameters β is multivariate normal. If a noninformative prior is desired, variances approaching infinity may be specified, yielding a prior proportional to a constant on the whole real line. This improper prior is widely used in hierarchical regression modeling and leads to proper posterior distributions (Diggle and Ribeiro, 2002).

The semi-conjugate prior for the temporal autocorrelation parameter ρ is normal, truncated to the interval $(-1, 1)$, or $(0, 1)$ if restriction to positive autocorrelation is justified in the application. The variance may be allowed to go to infinity, yielding a uniform prior over the appropriate interval. The analysis presented in this article simply uses $U(0, 1)$.

No semi-conjugate prior family exists for the spatial correlation parameter ϕ. A plausible, and minimally informative, choice is uniform or log-uniform such that the induced support interval (l, r) for ϕ have endpoints specifying a safe range. For example, in the case of spherical correlation where the spatial correlation between two sites with distance d is

$$\left(1 - \frac{3d\phi}{2} - \frac{d^3\phi^3}{2}\right) I\{d \leq 1/\phi\},$$

(l, r) can be chosen such that $1/r$ and $1/l$ represent the smallest and largest distances at which spatial correlation could conceivably decay to 0 in the application.

3 The RAMPS Algorithm

Our RAMPS algorithm is designed to approach independent sampling from the joint posterior distribution of all unknown model parameters θ. The posterior density of θ given Y may be factored as

$$p(\theta|Y) = p(\phi, \rho, \kappa|Y)p(\sigma^2|\phi, \rho, \kappa, Y)p(\beta|\phi, \rho, \sigma^2, \kappa, Y) \tag{9}$$

The RAMPS algorithm to draw posterior samples of θ given Y at the kth iteration is specified as follows:

1. Draw $(\phi^{(k)}, \rho^{(k)}, \kappa^{(k)})$ from their joint posterior marginal distribution $p(\phi, \rho, \kappa|Y)$ using slice sampling (Neal, 2003);

2. Draw $\sigma^{2,(k)}$ from $p(\sigma^2|\phi^{(k)}, \rho^{(k)}, \kappa^{(k)}, Y)$, which is an inverse gamma density; and

3. Draw $\beta^{(k)}$ from $p(\beta|\phi^{(k)}, \rho^{(k)}, \sigma^{2,(k)}, \kappa^{(k)}, Y)$, which is a multivariate normal density.

Of note is that none of the distributions depends on any parameter values from the previous iteration. Consequently, in cases where $p(\phi, \rho, \kappa|Y)$ is a known standard distribution, independent samples can be drawn from the joint posterior distribution $p(\theta|Y)$. For our model, however, $p(\phi, \rho, \kappa|Y)$ has a complex form for which the normalizing constant is unknown. Thus, we rely on a slice sampling method (Neal, 2003) to draw from $p(\phi, \rho, \kappa|Y)$.

Steps 2 and 3 of the algorithm are easy to implement. Let $\Omega(\phi, \rho, \kappa) = (1-\kappa)\Omega(\phi, \rho)+\kappa I$. It can be shown that, with a flat noninformative prior on β, the conditional distribution $p(\beta|\phi, \rho, \sigma^2, \kappa, Y)$ in step 3 is multivariate normal

$$N\left(\hat{\beta}, \quad \left[X^\top \sigma^2 \Omega^{-1}(\phi, \rho, \kappa)X\right]^{-1}\right), \tag{10}$$

where

$$\hat{\beta} = [X^\top \Omega^{-1}(\phi, \rho, \kappa) X]^{-1} X^\top \Omega^{-1}(\phi, \rho, \kappa) Y$$

is the generalized least squares (GLS) estimate of β given (ϕ, ρ, κ). With the priors on (σ^2, κ) in (7) and (8), the distribution $p(\sigma^2 | \phi, \rho, \kappa, Y)$ in step 2 is inverse gamma

$$\text{IG}\left(\alpha_z + \alpha_e + \frac{n-p}{2}, \quad \frac{b_z}{1-\kappa} + \frac{b_e}{\kappa} + \frac{1}{2}\hat{R}^\top \Omega^{-1}(\phi, \rho, \kappa)\hat{R}\right), \tag{11}$$

where $\hat{R} = Y - X\hat{\beta}$ is the residual from the the GLS fit. All quantities in (10) and (11) are computed in step 1 (see below) and need not be recomputed.

The only challenging step in the algorithm is step 1. The use of semi-conjugate priors on σ^2 and β, as given in Section 2, simplifies the process of integrating these parameters out of the joint posterior distribution to obtain the following analytic form of $p(\phi, \rho, \kappa | Y)$:

$$\begin{aligned} p(\phi, \rho, \kappa\ | Y) \propto & \left(\frac{b_z}{1-\kappa} + \frac{b_e}{\kappa} + \frac{1}{2}\hat{R}^\top \Omega^{-1}(\phi, \rho, \kappa)\hat{R}\right)^{-(\alpha_e + \alpha_z + \frac{n-p}{2})} \\ & \times |\Omega(\phi, \rho, \kappa)|^{-1/2} |X^\top \Omega^{-1}(\phi, \rho, \kappa) X|^{-1/2} \\ & \times \kappa^{\alpha_z - 1}(1-\kappa)^{\alpha_e - 1} \times I_{(l,r)}(\phi) \ \times \ I_{(-1,1)}(\rho), \end{aligned} \tag{12}$$

where $I_{(a,b)}(c) = 1$ if $c \in (a, b)$ and 0 otherwise. If there were a way to draw independent samples from (12), then our algorithm would produce independent draws from the joint posterior of all model parameters. This is, however, not possible, and, within each iteration of the MCMC sampler, we must turn to iterative methods to draw from (12). The method we use is slice sampling (Neal, 2003), which has been used by Agarwal and Gelfand (2002) in a different context of fitting spatial data models. At each MCMC iteration, slice sampling requires the identification of a region in the parameter space over which the density being sampled exceeds a stochastically-determined threshold value. Since this region obviously must be contained within the posterior support of the parameters being sampled, the search for this region is greatly simplified by parameterizing the model such that the posterior support of all parameters sampled by slice sampling is guaranteed to be bounded.

A commonly used slice sampler draws from a sequence of shrinking hyperrectangles placed around the values of the previous iteration. Let $g(\phi, \rho, \kappa)$ be the expression on the right side of (12). Let $g_0 = g(\phi^{(k)}, \rho^{(k)}, \kappa^{(k)})$. With bounded support, a pseudo-algorithm for our slice sampler that draws $(\phi^{(k+1)}, \rho^{(k+1)}, \kappa^{(k+1)})$ given $(\phi^{(k)}, \rho^{(k)}, \kappa^{(k)})$ is:

1. Draw U from a uniform distribution over $(0, 1)$.

2. Let $g_1 = g_0 U$ and let $(L_\phi, R_\phi) \times (L_\rho, R_\rho) \times (L_\kappa, R_\kappa)$ be the support of (ϕ, ρ, κ).

3. Draw $(\phi^*, \rho^*, \kappa^*)$ from a uniform distribution over $(L_\phi, R_\phi) \times (L_\rho, R_\rho) \times (L_\kappa, R_\kappa)$.

4. If $g(\phi^*, \rho^*, \kappa^*) > g_1$, then output $(\phi^{(k+1)}, \rho^{(k+1)}, \kappa^{(k+1)}) = (\phi^*, \rho^*, \kappa^*)$; Otherwise, shrink the hyperrectangles (see details below) and goto step 3.

The shrinking of the hyperrectangles is done in the same way for all three parameters. Taking ϕ as an example, if $\phi^* \leq \phi^{(k)}$, then $L_\phi = \phi^*$; if $\phi^* > \phi^{(k)}$, then $R_\phi = \phi^*$. With very large datasets, the variances of $(\phi, \rho, \kappa|Y)$ tend to be very small, and many loops through steps 3 and 4 may be required to find a point $(\phi^*, \rho^*, \kappa^*)$ such that $g(\phi^*, \rho^*, \kappa^*) > g_1$. This search is computationally expensive due to the repeated evaluations of g. In the SO_2 example in Section 6, the average number of evaluations per iteration computed from 1000 iterations is 9.887. The number of loops can be reduced, however, at the cost of higher autocorrelation in the samples, by replacing the initial hyperrectangle in step 2 with a smaller one randomly placed around the current point $(\phi^{(k)}, \rho^{(k)}, \kappa^{(k)})$. For a small dataset, the reduction in the number of loops may not worth the resulting stronger autocorrelation.

Autocorrelations are introduced into the RAMPS output through the shrinking hyperrectangle. Nonetheless, we found that slice sampling introduced lower levels of autocorrelation than the random-walk Metropolis-Hastings algorithm with a multivariate normal proposal density. Thus, all of our reported results are based on MCMC samplers using slice sampling to draw (ϕ, ρ, κ) from (12) in step 1, from a hyperrectangle centered at $(\phi^{(k-1)}, \rho^{(k-1)}, \kappa^{(k-1)})$. There is no dependence on previous values of σ^2 and β.

The computations needed in slice sampling from $p(\phi, \rho, \kappa|Y)$ require solving linear equation systems and calculating the determinant of $\Omega(\phi, \rho, \kappa)$. These quantities are computed efficiently after the Cholesky decomposition of $\Omega(\phi, \rho, \kappa)$. The Cholesky factorization of an $n \times n$ matrix is of order n^2 in terms of storage and n^3 in terms of computation. Thus, as sample size n becomes large, this step becomes very expensive in both storage and computation. To minimize this computational burden, the RAMPS algorithm is implemented using PLAPACK, a parallel linear algebra library described in the next section (van de Geijn, 1997).

The efficiency of the RAMPS algorithm can be appreciated by a comparison with the traditional Gibbs sampling algorithm, which updates through full conditional distributions, possibly blocked. We compared our serial algorithm with the Gibbs sampler algorithm as implemented in the R package spBayes (Finley et al., 2006). Both algorithms are implemented in C and interfaced to R. For a spatial dataset with 437 spatial observations that comes with the spBayes package, we ran 10,000 iterations using each algorithm on a 2.40GHz CPU Linux machine. The RAMPS algorithm took 9390s and the Gibbs sampler algorithm took 6228s. We discarded 1000 burn-in iterations from each run. The efficiency comparison results are summarized in Table 2, including ARL (loss of information due to autocorrelation), ESS (effective sample size), and ESS/s (ESS per second). ARL is an estimate of the autocorrelation time $\tau = 1 + 2\sum_{k=1}^{\infty} r(k)$ (Kass et al., 1998), where $r(k)$ is the autocorrelation at lag k for the parameter of interest. ESS is the number of points in the chain divided by τ. Though there are 5 regression coefficients in the model, only the first is reported because they have the same efficiency. Both algorithms draw the regression coefficients efficiently. For the other parameters $(\sigma_z^2, \sigma_e^2, \phi)$, the RAMPS algorithm generates samples with much lower autocorrelation, and, hence, many more effective samples. For example, in the most difficult case of ϕ, the Gibbs sampler yields 226.5 effective samples, while the RAMPS algorithm produces 1725.7 effective samples. When time is taken into consideration, the RAMPS algorithm generates 0.184 effective samples per second, while the Gibbs sampler generates 0.036 effective samples per second. This comparison confirms that the RAMPS algorithm is a superior candidate for parallelization.

[Table 2 about here.]

4 Parallelizing with PLAPACK

The literature on parallel algorithms for matrix operations is voluminous (see, for example, Duff and van der Vorst, 1999, for a review). There are two main open-source parallel direct solvers for dense matrices: ScaLAPACK (Blackford et al., 1997) and PLAPACK (van de Geijn, 1997). Both packages partition the matrix into submatrices and assign them to a grid of processors for parallel calculations. In this context, the word "grid" is used not as in general parallel computing for its specific connotation but as in the literature on parallel linear algebra algorithms – to refer to viewing the processors in a single cluster as being laid out in rows and columns. An important difference between ScaLAPACK and PLAPACK, from an application's point of view, is the strategy used to partition and distribute tasks. For ScaLAPACK, global matrices are distributed on the processor grid prior to the invocation of a ScaLAPACK routine. It is the application's responsibility to perform this data distribution. For PLAPACK, on the other hand, an application only needs to specify a scheme that defines a matrix distribution strategy. The real data distribution process is done by the package, significantly reducing application programming efforts. Although ScaLAPACK does provide more control and functionality, PLAPACK is used to parallelize the MCMC algorithm in this research.

5 Performance Study

We conducted a numerical study to gain insights into the timing performance of the parallel algorithm at various configurations of sample size and number of processors. The quantity of interest is the speedup of the parallel algorithm. Let $T(N)$ be the time required to complete a computation task on N processors. The speedup $S(N)$ is the ratio $S(N) = T(1)/T(N)$. The performance of the parallel algorithm was tested on both a local Beowulf Linux cluster and a reserved cluster on the TeraGrid (Reed, 2003), an element of the U.S. national cyber-infrastructure with integrated, persistent computational resources. Because the speedup results are virtually the same, we only report the results from our local Beowulf cluster since similar resources may be available at many institutions. Our Beowulf cluster runs RedHat Linux and has 14 nodes, each with dual 1.4GHz Xeon CPUs and 1GB memory. The entire cluster was dedicated to our use during the test. Since PLAPACK works best when the process grid is square, we tested for grid sizes 1, 4, 9, 16, and 25.

The model used in the performance test is the spatiotemporal model (6). The covariate matrix X has four columns: intercept, longitude, latitude, and time. The true value of β is $(1, 2, -1, 1)^\top$. The spatial correlation function is spherical with $\phi = 1.5$. The temporal correlation function is AR(1) with $\rho = 0.5$. The two variance components are $\sigma_z^2 = 1$ and

$\sigma_e^2 = 0.25$. The sample sizes n considered are 1,000, 2,000, 4,000, 6,000, 8,000, and 10,000. We could not go beyond 10,000 due to the memory limit of each single node. The number of spatial points are, respectively, 60, 120, 240, 360, 480, and 600. For all datasets, 20 time points were used but not all sites were sampled at all times.

[Table 3 about here.]

[Figure 2 about here.]

Table 3 and Figure 2 summarize the speedup obtained for all combinations of the 5 processor grid sizes and 6 sample sizes. It is clear that for smaller sample sizes, the benefit of a larger processor grid is reduced by inter-process communication overhead. As the sample size increases, however, the speedup for 25 processors also increases. For a dataset with 10,000 observations, the grid of 25 processors leads to a speedup factor of about 9. In summary, for larger problems, the parallel algorithm with 16–25 processors offers reasonable speedup and can go beyond the memory limit of single nodes.

6 Sulfur Dioxide Data Analysis

To illustrate the parallel MCMC algorithm, we analyze the aforementioned sulfur dioxide data obtained from 870 monitoring sites in the continental US from the year 1998 to the year 2005. The response variable is the mean of hourly measures of SO_2 concentration in 10^{-3} ppm. Table 1 summarizes the SO_2 concentrations for groups of year, latitude, and longitude. A decreasing trend is observed over the six years, with mean 5.12 in 1998 and 4.01 in 2000. Moving northward, as latitude increases by 3.9 degrees from group to group, the SO_2 concentration starts with mean 2.94 in the band of $(25.9, 29.8]$, increases to 6.02 in the band of $(37.5, 41.4]$, then decreases to 2.82 in the band of $(45.3, 49.2]$. Moving eastward, as longitude increases by about 10 degrees, the mean concentration starts with 2.21 in the band of $(-124, -115]$, increases to 5.84 in the band of $(-86.6, -77.2]$, and ends with 5.18 in the band of $(-77.2, -67.8]$. These descriptive statistics suggest the inclusion of a linear term of time and quadratic terms of latitude and longitude in the spatial trend.

More insight can be gained by looking at maps of SO_2 concentration. Figure 3 presents a year by year quilt plot (Nychka, 2005) of the mean concentration on the same scale, with monitoring sites discretized to a 128×64 grid. The high concentration values are scattered in a subregion of states around Ohio and Pennsylvania. This "bump" is consistent with the descriptive statistics in Table 1. Figure 4 presents an enlargement of this subregion discretized to a 128×64 grid. A similar pattern is observed across all six years. Within each year, higher values around the border of Ohio and Pennsylvania are mingled with lower values. These plots suggest that the spatial correlation is weak and that the temporal correlation is strong. In the analysis, a spherical structure is used for spatial correlation and an AR(1) structure is used for the temporal correlation.

[Figure 3 about here.]

[Figure 4 about here.]

The priors of the model parameters θ are specified as follows. The priors of the regression coefficients β are flat non-informative. The priors of the variance parameters (σ^2, κ) are induced by $\sigma_z^2 \sim \text{IG}(1, 1)$ and $\sigma_e^2 \sim \text{IG}(1, 1)$. The prior of the temporal correlation parameter ρ is $U(0, 1)$. The prior of the spherical spatial correlation parameter ϕ is $U(0.005, 1)$. This implies that the smallest and largest distances at which spatial correlation could conceivably decay to 0 are 1 mile and 200 miles, with distances computed as great circle distances.

After fitting several candidate regression models, we report the one with a quadratic term for latitude, a linear term for longitude, and a linear term for year. The latitude and longitude have been centered by their means. The year is coded such that year 0 means year 2000. Therefore, the intercept β_0 has the interpretation of mean SO_2 concentration at longitude -89.38 and latitude 38.63 in year 2000. The RAMPS algorithm converges quickly. We ran the MCMC sampler for 1000 iterations and discarded the first 50. The advantage of the algorithm is clearly seen from the autocorrelation plot of 950 posterior samples presented in Figure 5. The autocorrelation of ϕ, κ, ρ, and σ^2 drops rapidly. Even in the worst case of ϕ, it drops to zero after 10 lags. The autocorrelation for each regression coefficient drops immediately to zero.

[Figure 5 about here.]

The summary statistics for the model parameters from 950 posterior samples are reported in Table 4. These results are consistent with the descriptive statistics in Table 1 and Figures 3–4. The estimated spatial correlation drops to zero when two sites are about 7 miles apart. The measurement error accounts for about 10% of the total variation. It is interesting to make inference about the variance parameters on the scale of σ_z^2 and σ_e^2. The 95% credible intervals for σ_z^2 and σ_e^2 are, respectively, $(4.370, 5.212)$ and $(0.463, 0.561)$. The posterior mean of temporal correlation is 0.95 with a very tight 95% credible interval. The estimated regression coefficients of latitude and longitude produce a trend surface that captures the "bump" around Ohio and Pennsylvania. Most interestingly, the linear decrease of SO_2 concentration over the years is highly significant. On average, there is a reduction of 0.158 (10^{-3} ppm) each year.

[Table 4 about here.]

We ran the analysis using the TeraGrid with different numbers of processors. The TeraGrid cluster we used consists of 262 nodes, each with dual 1.5 GHz Intel Itanium 2 processors, for a peak performance of 3.1 teraflops. Each node is equipped with 4GB of physical memory. The cluster runs SuSE Linux and employs Myricom's Myrinet cluster interconnect network. For 1000 iterations, it took 19.5 hours for 9 processors, 16.2 hours for 16 processors, and 14.5 hours for 25 processors. These timing results are to be expected given the performance summary in Table 3, which provides a rough guideline for choosing the number of processors given a problem size.

7 Discussion

This article has discussed an efficient MCMC algorithm named RAMPS for Bayesian spatiotemporal geostatistical modeling. Chains generated with this algorithm converge rapidly. When the marginalized posterior density only involves one or two variables, the algorithm is very efficient if we use an adaptive rejection or adaptive rejection Metropolis sampling (ARMS) algorithm (Gilks et al., 1995) in place of the slice sampling algorithm. Even in examples with 3 marginalized variables, similar to our analysis with the $S0_2$ data, convergence is usually obtained within the first 50 iterations. As a result, the autocorrelations in the posterior samples are much lower in Figure 5 compared to other algorithms (for example, Whiley and Wilson, 2004, Figure 3). The cost of the algorithm is that, when drawing (ϕ, ρ, κ) with slice sampling, the joint posterior marginal density $p(\phi, \rho, \kappa|Y)$ must be evaluated repeatedly. For example, in the SO_2 example, the average number of evaluations per iteration is 9.887 based on 1,000 iterations. The efficiency of the RAMPS algorithm may be improved if we adaptively choose the size of the initial hyperrectangle used by the slice sampling procedure. This is an important topic for future work.

For large datasets, the evaluation of $p(\phi, \rho, \kappa|Y)$ is computationally very expensive, if feasible at all. The approach of this article is to parallelize it with the Parallel Linear Algebra Package PLAPACK (van de Geijn, 1997). The data distribution strategy used in PLAPACK made it an attractive option since we were concerned with the provision of interfaces between applications and libraries. Since users do not need to attend to data distribution details, programming efforts are significantly reduced. From our performance study, this approach has reasonable scalability as shown in Figure 2, which makes it a good choice for spatiotemporal geostatistical modeling of the large datasets frequently encountered in environmental research.

Acknowledgments

This research was supported in part by the National Science Foundation through TeraGrid resources grant TG-SES060003T. The authors thank Brian Smith for detailed comments on an earlier version of the manuscript and Wenli He for TeraGrid computing assistance.

References

Adams, N. M., Kirby, S. P. J., Harris, P., and Clegg, D. B. 1996. A review of parallel processing for statistical computation. Statistics and Computing 6: 37–49.

Agarwal, D. K. and Gelfand, A. E. 2002. Slice sampling for simulation based fitting of spatial data models. Statistics and Computing 15: 61–69.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. 2004. Hierarchical modeling and analysis for spatial data. Chapman & Hall / CRC.

Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. C. 1997. ScaLAPACK Users' Guide. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Christiansen, C. L. and Morris, C. N. 1997. Hierarchical Poisson regression modeling. Journal of the American Statistical Association 92: 618–632.

Cowles, M. K., Zimmerman, D. L., Christ, A., and McGinnis, D. L. 2002. Combining snow water equivalent data from multiple sources to estimate spatio-temporal trends and compare measurement systems. Journal of Agricultural, Biological, and Environmental Statistics 7: 536–557.

Daniels, M. J. 1999. A prior for the variance in hierarchical models. The Canadian Journal of Statistics / La Revue Canadienne de Statistique 27: 567–578.

Diggle, P. and Ribeiro, P. J. 2002. Bayesian inference in Gaussian model-based geostatistics. Geographical and Environmental Modelling 6: 129–146.

Duff, I. S. and van der Vorst, H. A. 1999. Developments and trends in the parallel solution of linear systems. Parallel Computing 25: 1931–1970.

Finley, A. O., Banerjee, S., and Carlin, B. P. 2006. spBayes: spBayes fits Gaussian models with potentially complex hierarchical error structures. R package version 0.0-1.
URL `http://blue.fr.umn.edu/spatialBayes`

Gilks, W. R., Best, N. G., and Tan, K. K. C. 1995. Adaptive rejection Metropolis sampling within Gibbs sampling (Corr: 97V46 p541-542 with R. M. Neal). Applied Statistics 44: 455–472.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. 1998. Markov chain Monte Carlo in practice: A roundtable discussion. The American Statistician 52: 93–100.

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

Nychka, D. 2005. fields: Tools for spatial data. R package version 3.04.
URL `http://www.image.ucar.edu/GSP/Software/Fields`

Reed, D. A. 2003. Grids, the teragrid, and beyond. Computer 36: 62–68.

Rosenthal, J. S. 2000. Parallel computing and Monte Carlo algorithms. Far East Journal of Theoretical Statistics 4: 207–236.

Rossini, A., Tierney, L., and Li, N. 2003. Simple parallel statistical computing in R. Working paper, UW Biostatistics.

Schervish, M. J. 1988. Applications of parallel computation to statistical inference. Journal of the American Statistical Association 83: 976–983.

Sylwestrowicz, J. D. 1982. Parallel processing in statistics. In H. Caussinus, P. Ettinger, and R. Tomassone (eds.), COMPSTAT 1982, Proceedings in Computational Statistics, pp. 131–136, Physica-Verlag Ges.m.b.H.

van de Geijn, R. A. 1997. Using PLAPACK. The MIT Press.

Whiley, M. and Wilson, S. P. 2004. Parallel algorithms for Markov chain Monte Carlo methods in latent spatial Gaussian models. Statistics and Computing 14: 171–179.

Wilkinson, D. J. 2005. Parallel Bayesian computation. In E. J. Kontoghiorghes (ed.), Handbook of Parallel Computing and Statistics, pp. 481–512, Marcel Dekker/CRC Press.

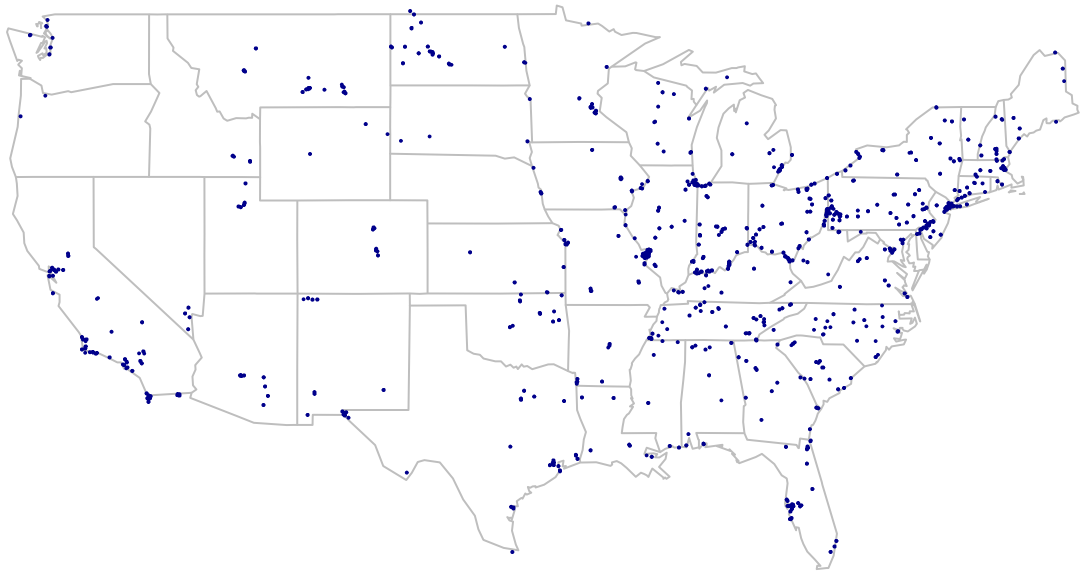

Figure 1: Sulfur dioxide monitoring sites.

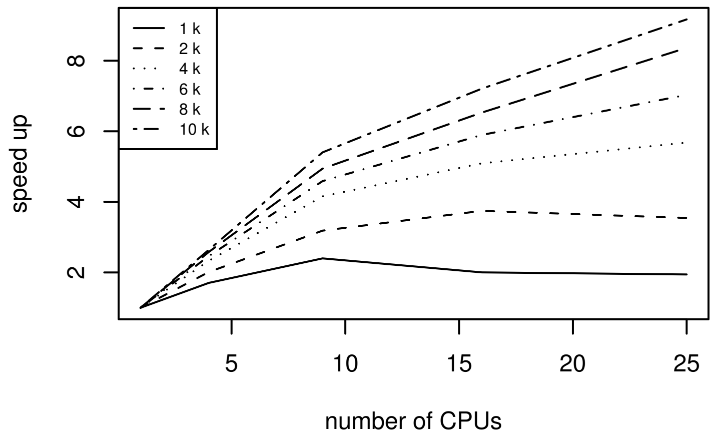

Figure 2: Speedup comparison for the spatiotemporal model as sample size increases from 1000 (1k) to 10,000 (10k).

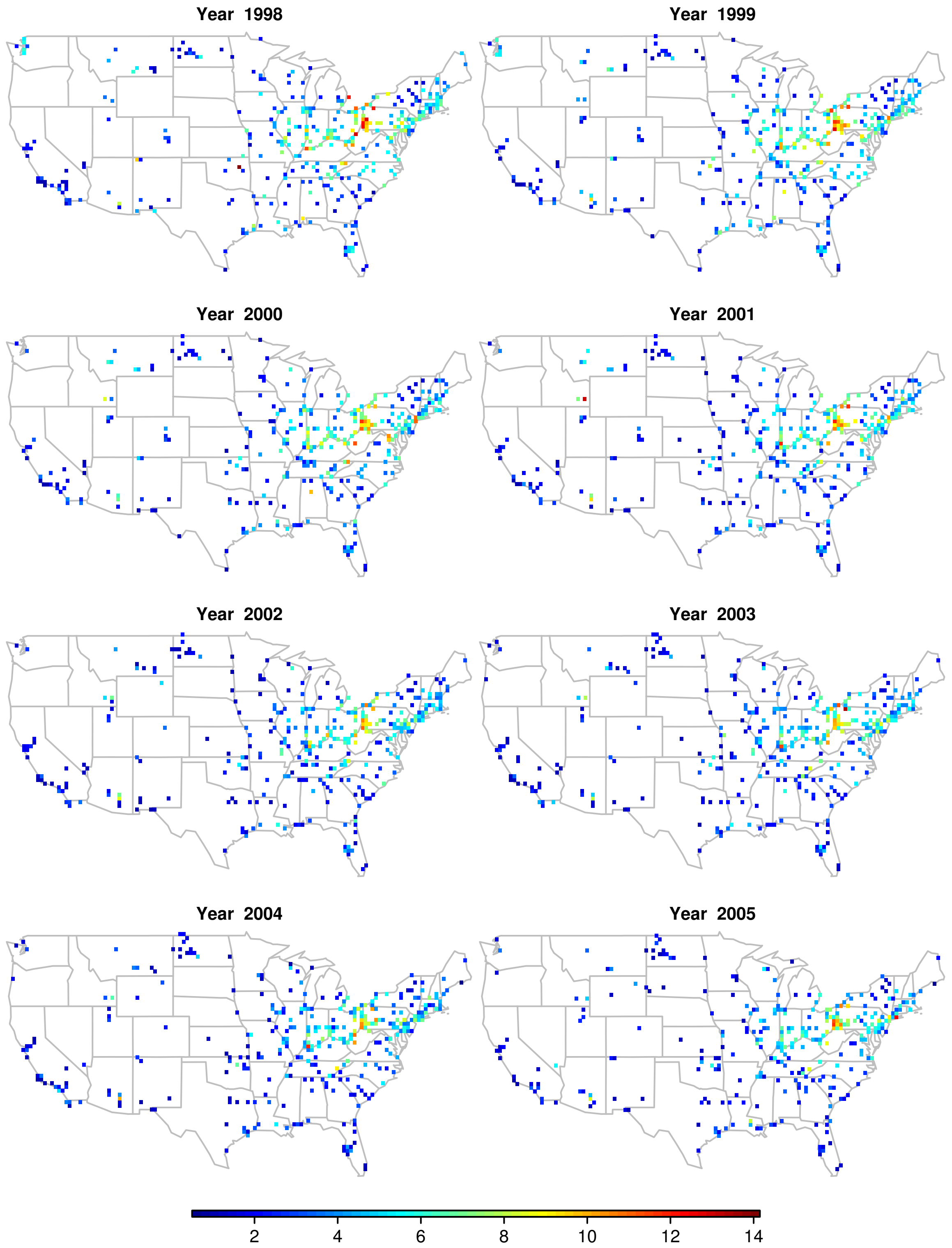

Figure 3: Quilt plot of SO_2 concentration (10^{-3} ppm) over years in the US.

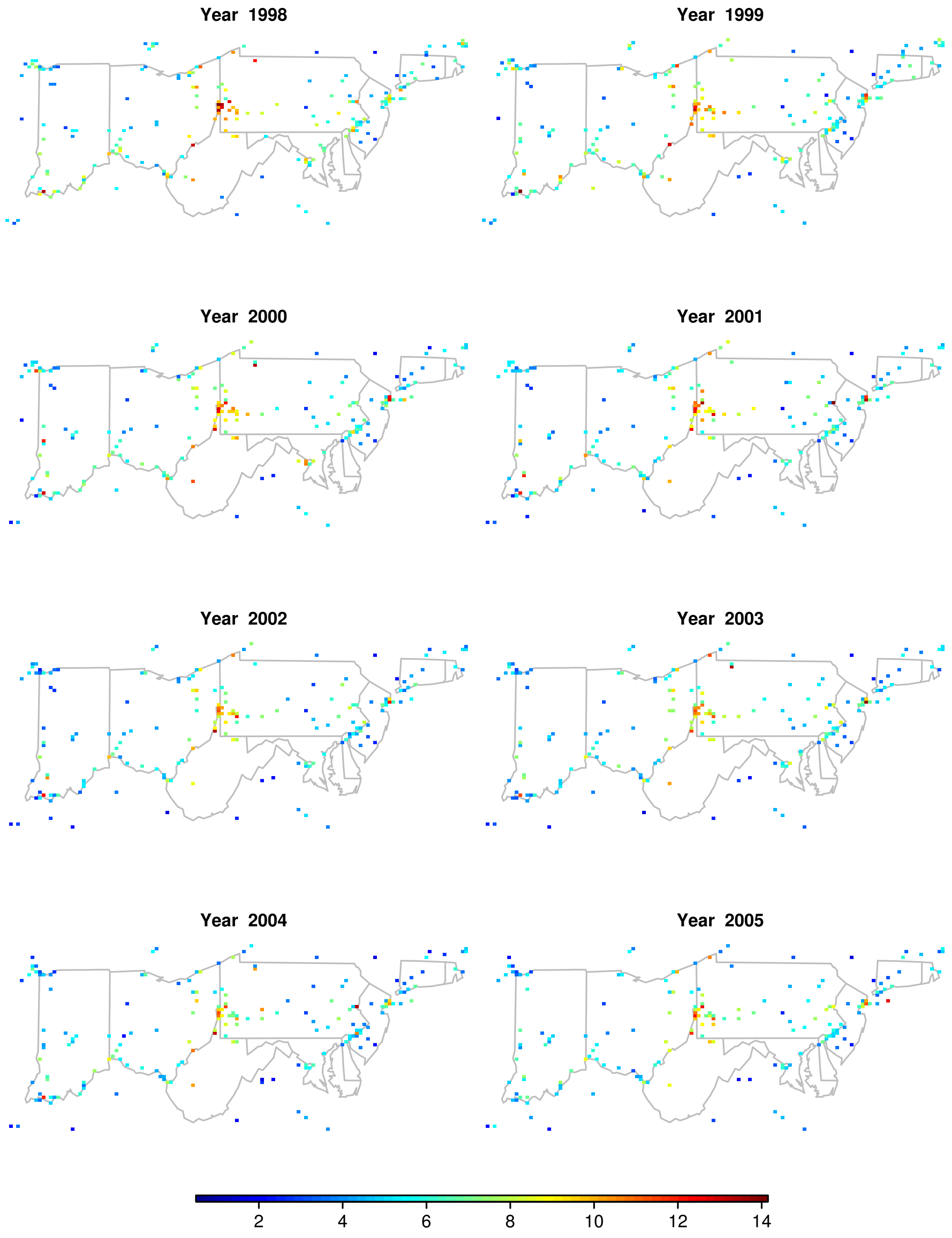

Figure 4: Quilt plot of SO_2 concentration (10^{-3} ppm) over years in a subregion in north-central U.S.

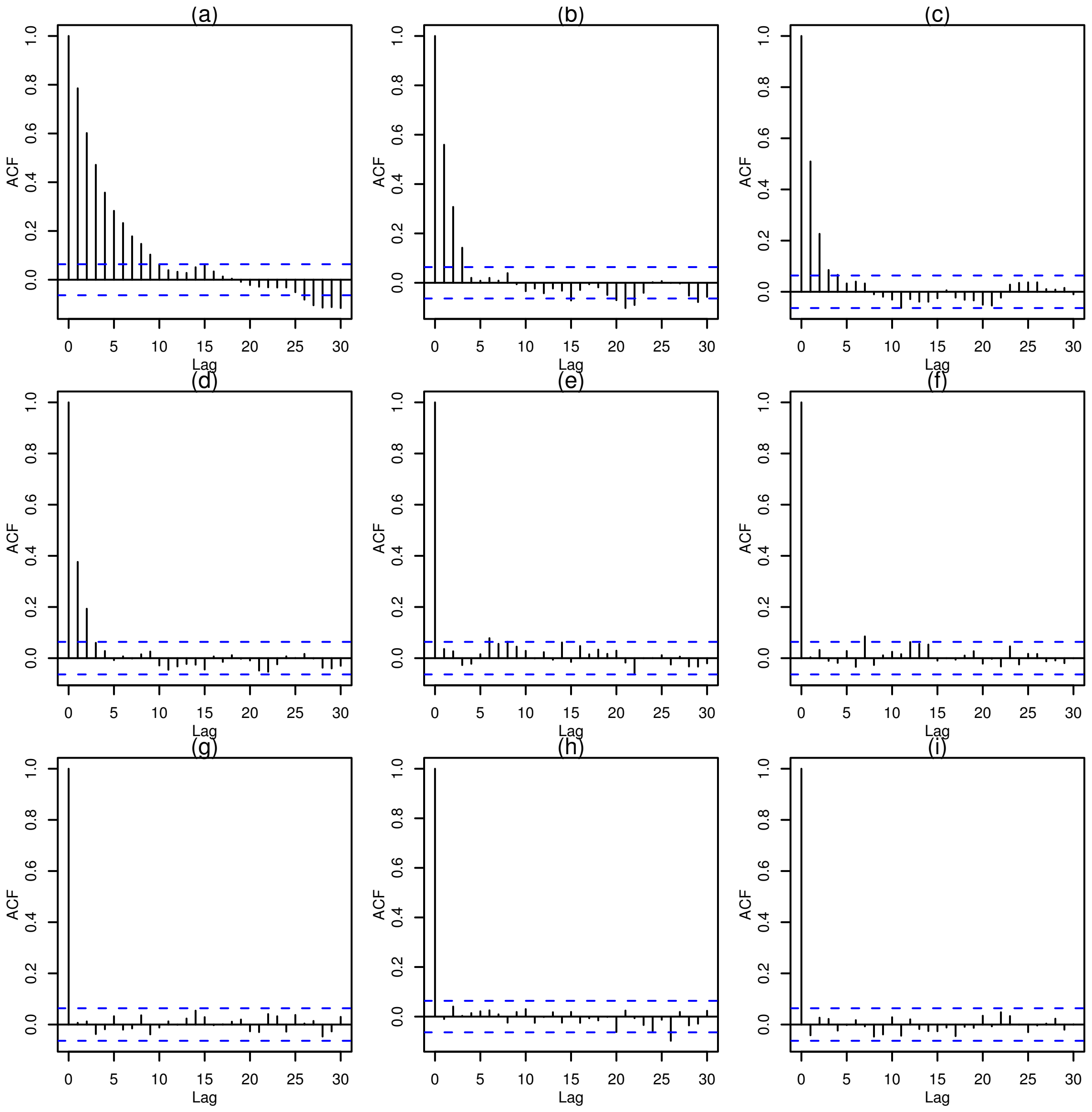

Figure 5: Autocorrelation plot from 950 posterior samples: (a) ϕ; (b) ρ; (c) κ; (d) σ^2; (e) intercept β_0; (f) coefficient of latitude β_1; (g) coefficient of latitude squared β_2; (h) coefficient of longitude β_3; and (i) coefficient of year β_4.

Table 1: Summary statistics of SO_2 concentration (10^{-3}ppm) for years 1998–2005 and groups of latitude and longitude. Nobs is the number of observations.

	Nobs	Mean	SD
Years			
1998	667	5.12	3.03
1999	643	5.06	2.90
2000	606	4.80	2.96
2001	596	4.50	2.91
2002	578	4.09	2.58
2003	561	4.12	2.60
2004	542	3.97	2.52
2005	518	4.01	2.56
Grouping of Latitude			
$(25.9, 29.8]$	233	2.94	1.52
$(29.8, 33.6]$	521	2.91	1.66
$(33.6, 37.5]$	805	3.39	2.20
$(37.5, 41.4]$	1836	6.02	3.03
$(41.4, 45.3]$	955	4.35	2.37
$(45.3, 49.2]$	361	2.82	1.69
Grouping of longitude			
$(-124, -115]$	426	2.21	1.28
$(-115, -105]$	329	3.30	2.36
$(-105, -95.9]$	347	2.38	1.75
$(-95.9, -86.5]$	1218	4.01	2.17
$(-86.5, -77.1]$	1570	5.84	3.17
$(-77.1, -67.8]$	821	5.18	2.34

Table 2: Efficiency comparison of the RAMPS algorithm and the Gibbs sampler for a sample dataset of 437 spatial observations. ARL is the loss of information due to autocorrelation measured by the expected lag at which autocorrelation drops to zero. ESS is the effetive sample size. ESS/s is ESS per second. Both algorithms were run with 10,000 iterations on a 2.40GHz CPU Linux machine with the first 1,000 discarded. The RAMPS algorithm took 9390s and the Gibbs sampler took 6228s.

Parameter	Gibbs Sampler			RAMPS Algorithm			ESS/s Ratio
	ARL	ESS	ESS/s	ARL	ESS	ESS/s	RAMPS/Gibbs
β_0	1.00	9000.0	1.445	1.00	9000.0	0.958	0.663
σ_z^2	37.28	241.4	0.039	1.30	6913.9	0.736	18.872
σ_e^2	23.27	386.8	0.062	1.74	5172.1	0.551	8.887
ϕ	39.74	226.5	0.036	5.22	1725.7	0.184	5.111

Table 3: Speedup comparison for the spatiotemporal model as samples sizes increases.

Number of	Sample Size					
CPUs	1000	2000	4000	6000	8000	10000
1	1.00	1.00	1.00	1.00	1.00	1.00
4	1.70	2.01	2.33	2.46	2.58	2.64
9	2.40	3.18	4.16	4.59	4.94	5.40
16	2.00	3.75	5.09	5.90	6.53	7.21
25	1.94	3.54	5.68	7.03	8.36	9.17

Table 4: Summary statistics of the model parameters from 950 posterior samples.

Parameter	Mean	SD	Percentiles		
			2.5%	50%	97.5%
ϕ	0.141	0.018	0.109	0.139	0.179
ρ	0.950	0.004	0.940	0.950	0.956
κ	0.097	0.006	0.085	0.096	0.109
σ^2	5.288	0.216	4.877	5.277	5.722
β_0	4.799	0.111	4.587	4.795	5.023
β_1	0.032	0.018	-0.003	0.032	0.067
β_2	-0.021	0.003	-0.027	-0.021	-0.016
β_3	0.065	0.006	0.052	0.065	0.078
β_4	-0.158	0.013	-0.184	-0.158	-0.134