Multi-resolution Genetic Algorithms and Markov Chain Monte Carlo

June 11, 2002

Christopher H. Holloman†

Duke University, Durham, USA

Herbert K. H. Lee

Duke University, Durham, USA

Dave M. Higdon

Duke University and Los Alamos National Labs, Los Alamos, USA

Summary. Due to modern advances in computing power, the use of increasingly complex models has become practical. One class of large models that often relies on numerical techniques for parameter estimation is multi-resolution models. Unfortunately, numerical maximization and sampling techniques used to estimate parameters in such complex models often explore the parameter space slowly, resulting in unreliable or unstable estimates. This paper proposes a multi-resolution genetic algorithm that incorporates elements of simulated tempering to allow efficient estimation of parameters in multi-scale models. This algorithm can also be adapted to perform Markov chain Monte Carlo sampling from a posterior distribution in a Bayesian setting, which can greatly improve mixing and exploration of the posterior. Parallel implementation is addressed. These methods are demonstrated on examples from single photon emission computed tomography and groundwater hydrology.

Keywords: Bayesian Statistics, Parallel Computing, Simulated Tempering, Evolutionary Monte Carlo

1. Introduction

Many natural phenomena that are measured over a continuous domain can be discretized and modeled on multiple scales. For instance, in applications involving the analysis of noisy images, a refinement must be

†*Address for correspondence*: Christopher Holloman, ISDS, Box 90251, Duke University, Durham, NC 27708, USA. Email: chris@stat.duke.edu

chosen for the grid at which values are to be estimated. Typically, interest lies in fitting a model at a fine scale including as much detail as possible. However, when modeling on a fine scale, model fitting can be slow since refining the scale introduces more parameters into the model. Also, for finer scale models likelihood evaluations often take longer since more information must be considered when relating data to model parameters. Modeling on a coarser scale requires the estimation of fewer parameters but yields less information. Using a coarser scale when the likelihood is highly multimodal or difficult to fully explore can often greatly increase the speed of convergence to the global maximum. An illustrative example and further discussion appear in Liu and Sabatti (1999). In order to take advantage of the beneficial qualities of both fine and coarse scale models, we introduce a multi-scale genetic algorithm that links coarse and fine scale models together.

In many situations, data naturally arise on different scales. When faced with data to be modeled at different resolutions, whether because of the actual data or for computational reasons, one useful approach is to model the data at several resolutions simultaneously. Several disciplines address the topic of multi-resolution models. In the field of statistical engineering, Lakshmanan and Derin (1993) explore multi-resolution Markov random fields, and Bouman and Liu (1991) use multiple resolutions to segment images into regions with similar statistical qualities. In the field of hydrology, Yoon et al. (1999) use multi-scale refinement techniques to find globally optimal reservoir characteristics. From a statistical standpoint, Higdon et al. (2002) use Metropolis coupling on multiple resolutions, which also serves to increase mixing in a Markov chain Monte Carlo algorithm. All of these multi-scale models involve parameter spaces of large dimension requiring complex algorithms for parameter estimation.

One highly effective technique for maximizing irregular functions defined over high-dimensional spaces is the genetic algorithm (Holland, 1975). The original form of the algorithm is suited only to solving problems for which all considered solutions have the same dimension. To overcome the fixed solution length limitation, Goldberg et al. (1989) introduces a variable string length genetic algorithm, which allows for solution representations of varying sizes in order to facilitate maximization. Although this algorithm only solves problems where the final solution is of a fixed dimension, it demonstrates the utility of allowing different solution representations within the algorithm. The algorithm we propose also extends the original genetic algorithm by making use of related solutions of differing dimensions.

In addition to maximization, genetic algorithms have been adapted for use in Markov chain Monte Carlo algorithms, sometimes referred to as Evolutionary Monte Carlo (Liang and Wong, 2001; Holmes and Mallick, 1998). Building upon these algorithms and ideas from simulated tempering (Geyer and Thompson, 1995; Marinari and Parisi, 1992), we introduce a multi-resolution Markov chain Monte Carlo algorithm for sampling from a posterior distribution in a Bayesian setting. Our work is also related to multigrid Monte Carlo methods and simulated sintering (Goodman and Sokal, 1989; Liu and Sabatti, 1999, 2000).

1.1. Genetic Algorithms and MCMC

Holland (1975) pioneered the genetic algorithm, a maximization technique for functions defined on multidimensional spaces. Chatterjee et al. (1996) provides a good overview of the concepts in genetic algorithms including basic applications for the field of statistics. Our perspective in this paper derives from interest in real-valued parameters, but we note that the methodology herein is equally applicable to parameters from a finite alphabet. In the rest of this section, we describe a simple genetic algorithm, and then we explain how it may be adapted to sample from a distribution instead of performing a maximization.

Suppose we wish to create a genetic algorithm to maximize some function $g(\mathbf{v})$, where $\mathbf{v}$ is a vector $(v_1, \ldots, v_L)$. To initialize the algorithm we must create a population of solutions each of which is represented as a vector of length equal to the number of parameters, L. For example, a population of solutions $\mathbf{v}^1_{(0)}, \ldots, \mathbf{v}^M_{(0)}$ could be created by randomly selecting M vectors of length L from some distribution. Here, the subscript in parentheses denotes the iteration number within the algorithm. Ordinarily L is fixed, but M, the number of solutions, may be adjusted. One cycle of the algorithm is comprised of three steps: selection, crossover and mutation.

Selection. In the selection step, the population is altered by allowing better solutions to remain while poorer solutions are potentially removed. Recall that the goal of the genetic algorithm is to maximize $g(\mathbf{v})$, typically called the fitness function. The selection step proceeds by drawing M vectors with replacement from the current population of solutions to form a new population. If $g(\cdot)$ is non-negative, solution vectors are selected with probability proportional to their fitness, $g(\mathbf{v})$. Thus, for one solution $\mathbf{v}^i_{(t)}$, its probability of being selected is $g(\mathbf{v}^i_{(t)}) / \sum_{j=1}^{M} g(\mathbf{v}^j_{(t)})$. In this way, solutions with a higher fitness have a larger probability of being selected and moving through to the next cycle. If $g(\cdot)$ is not a non-negative function then it must

be related to some other non-negative fitness function that can be used to select "better" solutions (*e.g.*, $\exp(g)$).

Crossover. To perform a crossover, several pairs of solutions (up to $M/2$ pairs) are selected and some of their values are traded. It has been recommended that for each pair, the probability of performing a trade should be tuned to be between 0.6 and 0.95 (see Bäck (1993) and the references therein). If a pair is selected to perform a trade, then one of several types of crossovers may be performed. In a one-point crossover, a single element is chosen randomly from $\{1, \ldots, L-1\}$ and all elements after the chosen element are swapped. The following diagram depicts a one point crossover between $\mathbf{v}^1_{(t)}$ and $\mathbf{v}^2_{(t)}$ when the element z^* is chosen. The subscript denoting the iteration, (t), is suppressed within the expanded vectors for clarity.

$$\begin{aligned} \mathbf{v}^1_{(t)} &= \{v^1_1, \ldots, v^1_{z^*}, v^1_{z^*+1}, \ldots, v^1_L\} \xrightarrow{\text{Swap Elements}} \{v^1_1, \ldots, v^1_{z^*}, v^2_{z^*+1}, \ldots, v^2_L\} \\ \mathbf{v}^2_{(t)} &= \{v^2_1, \ldots, v^2_{z^*}, v^2_{z^*+1}, \ldots, v^2_L\} \qquad\qquad \{v^2_1, \ldots, v^2_{z^*}, v^1_{z^*+1}, \ldots, v^1_L\} \end{aligned}$$

In a k-point crossover, k elements are chosen and segments between the chosen elements alternate between swapping and not swapping. In a uniform crossover, every element has some equal probability of being swapped. In addition, more complicated forms of crossover have been proposed, for example, the snooker crossover (Liang and Wong, 2001).

Mutation. In the mutation step, each element of each vector in the population of solutions is perturbed with some small mutation probability. For instance, using a probability of mutation $p^* = .01$, in each cycle approximately $.01 \times M \times L$ elements would be altered. When working with real valued vectors, mutation of a single element can be performed by adding random noise to the selected element.

At the end of the mutation step, the next iteration of the algorithm begins. These three steps are repeated until some measure of convergence is satisfied. Several modifications to the basic genetic algorithm have been proposed to speed up convergence, for example, the elitist strategy (DeJong, 1975), where the the solution with the highest fitness value is automatically retained unless a better one is found through mutation and crossover.

Genetic algorithms have been adapted to facilitate MCMC analyses. In order to implement these algorithms several MCMC chains are run simultaneously. For each chain, parameter updates (mutations) are analogous to mutation steps in a genetic algorithm. However, instead of always accepting the mutations

as in the genetic algorithm setting, an acceptance probability must be calculated if Metropolis-Hastings updates are used. Selection and crossover are used in addition to within-chain parameter updates to develop intelligent proposals for a pair of chains. These proposals are accepted or rejected according to a Metropolis-Hastings rule. Holmes and Mallick (1998) combine selection and crossover steps by selecting pairs according to their fitness for uniform crossovers. Liang and Wong (2001) use a similar selection/crossover combination and incorporate elements of simulated tempering (discussed in the next section) into the MCMC.

1.2. Simulated Tempering and Simulated Sintering

Simulated tempering is an MCMC scheme proposed independently by Marinari and Parisi (1992) and Geyer and Thompson (1995) to help explore complex distributions by using a series of other distributions. If $h_1(\boldsymbol{\theta})$ is the distribution from which we want to draw samples, we create $M-1$ other distributions, $h_2(\boldsymbol{\theta}), \ldots, h_M(\boldsymbol{\theta})$ to facilitate sampling. Here, $h_1(\boldsymbol{\theta})$ is called the "cold" distribution and $h_2(\boldsymbol{\theta}), \ldots, h_M(\boldsymbol{\theta})$ are called the "heated" distributions. Usually, the heated distributions are successively easier to sample; however, they are also successively less similar to the distribution of interest. When creating heated distributions, one common method derived from simulated annealing sets $h_j(\boldsymbol{\theta}) \propto h_1(\boldsymbol{\theta})^{(1/\beta_j)}, 1 < \beta_2 < \ldots < \beta_M$. Once the distributions are defined, simulated tempering algorithms perform inference over the space of all of the distributions using a single MCMC chain. In this MCMC chain, the state at any iteration t is denoted $(\boldsymbol{\theta}_{(t)}, j_{(t)})$ where $\boldsymbol{\theta}$ is a parameter vector and j denotes the current distribution h_j. MCMC updates made along the parameter vector $\boldsymbol{\theta}$ while holding the distribution parameter j constant produce movements within one distribution. MCMC updates of the distribution parameter j made while holding $\boldsymbol{\theta}$ constant produce movements across distributions. Typically, movements across distributions increase or decrease j by one, restricting such movements to adjacent distributions along the heating scale. Because the chain moves between distributions, it does not always sample from the distribution of interest; however, it moves around the space more freely in the heated distributions allowing more efficient exploration of the target distribution. Due to this ease of movement, simulated tempering is well suited to problems where the distribution of interest is multimodal or not connected.

Simulated sintering (Liu and Sabatti, 1999) can be seen as a generalization of Gibbs sampling and of simulated tempering. As with this paper, coarser versions of the parameter space can be used as the heated

distributions for tempering. Sintering, however, like tempering, uses a single chain that moves between dimensions. In this paper, we explore methodology that not only allows a chain to move between scales, but also allows information to be shared between multiple chains at different scales.

2. Multi-resolution Models

2.1. Statement of the Model

We start with a general formulation of our model. In many cases, the form simplifies considerably, but we state the model to be as widely applicable as possible. The key element is a subset of parameters that is of the same dimension and comparable interpretation for each scale of the model. It is this subset that serves as the basis for the genetic algorithm and tempering types of moves between scales.

Denote the data for scale i, $i \in \{1, \ldots, I\}$, by $\mathbf{y}^{(i)}$. We use superscripts in parentheses to denote scale-specific data and parameters throughout this paper. Note that in some problems different data are observed at each scale, while in other problems the same data apply to all scales, i.e., $\mathbf{y}^{(i)} = \mathbf{y}^{(j)}$ for all i and j. Denote the parameters at scale i by $\boldsymbol{\psi}^{(i)}$. The dimension of $\boldsymbol{\psi}$ usually varies by scale. The full likelihood is taken as the product of the likelihoods for each scale:

$$L(\boldsymbol{\psi} \mid \mathbf{y}) = \prod_{i=1}^{I} L^{(i)}(\boldsymbol{\psi}^{(i)} \mid \mathbf{y}^{(i)}).$$

We note that in this formulation, the connections between scales are induced only through the model-fitting algorithm, and are not modeled explicitly.

The particular form of the likelihood is problem-dependent. The only restriction of our algorithm is that the multi-scale parameter $\boldsymbol{\psi}^{(i)}$ must be written in terms of two other variables, $\boldsymbol{\phi}^{(i)}$ and $\boldsymbol{\lambda}^{(i)}$, where $\boldsymbol{\phi}^{(i)}$ is a fixed-dimensional part that is used to move between scales (and thus requires common interpretability), and $\boldsymbol{\lambda}^{(i)}$ denotes the rest of the necessary parameters. $\boldsymbol{\psi}^{(i)}$ relates to these parameters by a deterministic function $g^{(i)}$ (although in the Bayesian context, it no longer needs to be deterministic):

$$\boldsymbol{\psi}^{(i)} = g^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)}).$$

Later in this paper we will be interested in Bayesian versions, and we note here that there will be a prior $\pi^{(i)}$ for the parameters $(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)})$ at each scale.

2.1.1. Example: Using the mean to move between scales

Suppose that our observed data $\mathbf{y}$ arise from a noisy, known, possibly non-linear transformation f of an unobserved spatial process $\boldsymbol{\psi}$:

$$y^{(j)} = f(\boldsymbol{\psi}^{(j)}) + \varepsilon^{(j)} , \qquad \varepsilon^{(j)} \overset{iid}{\sim} N(0, \sigma^2) .$$

Suppose our interest lies in inferring the latent process on a grid at a particular scale, $\boldsymbol{\psi}^{(1)}$. For computational efficiency, it is often helpful to introduce coarser versions, $\boldsymbol{\psi}^{(2)}, \ldots, \boldsymbol{\psi}^{(I)}$, as the likelihood tends to become smoother and less multimodal as the resolution decreases. For simplicity, consider modeling a two-dimensional process on just two scales (the procedure has a straightforward extension to additional scales), a fine scale and a coarse scale with half the resolution (and thus one-fourth the number of cells). One way to get a fixed-dimensional common parameter vector, $\boldsymbol{\phi}^{(i)}$, is to let $\boldsymbol{\phi}^{(2)}$ equal the coarse parameters $\boldsymbol{\psi}^{(2)}$, let $\boldsymbol{\phi}^{(1)}$ be the mean of each set of four fine cells that correspond to a single coarse cell, and let $\boldsymbol{\lambda}^{(1)}$ contain the deviations from the mean of the upper left, upper right, and lower left cells of each group of four (since the fourth cell will be determined by the other three cells and the mean level). Thus $\boldsymbol{\lambda}^{(2)}$ is empty, $g^{(2)}$ is the identity, and $g^{(1)}$ calculates the values of the fine cells from the mean for each group of four and the differences from the mean for the individual cells. Note that the parameter for the noise at each resolution, $\sigma^{(i)}$, would typically be included as an element of $\boldsymbol{\lambda}^{(i)}$. We detail a more complex version of this example in Section 3.

2.1.2. Example: Discretization of a continuous process with a common basis representation

As with the previous example, suppose the data come from a known transformation of a latent process. Here we use a basis representation so that the parameters are directly comparable across scales. We think of the process on each scale as the discretization of a continuous process, where the continuous process is represented as a combination of a fixed number of bases. For example, consider the two-dimensional problem illustrated by the diagram in Figure 1. The dark points are the bases, $\phi_j^{(i)}$, which can be combined by convolving with a smoothing kernel, k, to produce a continuous process z, *e.g.*,

$$z^{(i)}(s) = \sum_j k(d(s, \phi_j^{(i)}))\ \phi_j^{(i)}.$$

where k is a smoothing kernel (such as a Gaussian kernel) and $d(s, \phi_j^{(i)})$ represents the distance between the point s at which the process is being evaluated and the location of the basis point $\phi_j^{(i)}$ (whose location does not depend on the scale i). This continuous process $\mathbf{z}$ is then discretized to a particular resolution by letting $\boldsymbol{\psi}$ be the value of $\mathbf{z}$ at the center of each grid block. Thus $\boldsymbol{\psi}^{(i)}$, a deterministic transformation of $\boldsymbol{\phi}^{(i)}$, has a dimension which depends on the resolution i, but the parameter vector $\boldsymbol{\phi}^{(i)}$ has the same dimension and interpretation (although possibly different values) for each scale. Note that $\boldsymbol{\lambda}^{(i)}$ would contain only the noise parameter, $\sigma^{(i)}$, as finer scales require no additional parameters compared to coarser scales. We examine a more complex version of this example in Section 4.

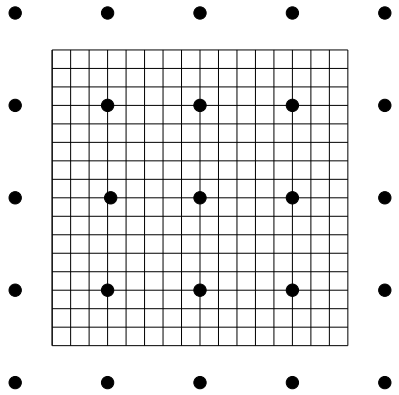

Fig. 1. Basis point locations and a field, ψ, to be estimated (discretized into a 16×16 grid in this figure).

2.2. Multi-resolution Genetic Algorithm

Whether we have multi-scale data, or our interest focuses only on the finest scale, information from all scales can often be useful for facilitating maximization of the finest scale. To take advantage of all scales we introduce a genetic algorithm modified to accommodate multiple resolutions. The key is that $\boldsymbol{\phi}$ has the same dimension, C, and common interpretation regardless of the scale i. To initialize the algorithm, we choose $M > 2 \times I$ random solution vectors, $\mathbf{v}_{(0)}^1, \ldots, \mathbf{v}_{(0)}^M$, where $\mathbf{v}_{(0)}^m = (\boldsymbol{\phi}_{(0)}^m, \boldsymbol{\lambda}_{(0)}^m, i_{(0)}^m)$ for $m = 1, \ldots, M$. The last parameter in this solution vector identifies the resolution (and portion of the full likelihood) to which the solution vector corresponds. There should be at least two solution vectors for each resolution, though the algorithm will work better with several solution vectors at each resolution. Once initialized, the algorithm follows the selection, crossover, and mutation steps as follows.

Selection. Since the conditionally independent portions of the full likelihood are all non-negative functions, they can be used as a measure of fitness for each of the proposed solutions. Here, our algorithm differs from the traditional genetic algorithm since I different fitness functions exist against which solutions are evaluated. Consider the subset of $n_{(t)}^{(i)}$ solution vectors for which the last element in the solution vector is some fixed $i \in \{1, \ldots, I\}$ at some time t; that is, consider all solution vectors corresponding to a single resolution at time t. From these $n_{(t)}^{(i)}$ vectors, we select $n_{(t)}^{(i)}$ with replacement with probability proportional to their value on the corresponding portion of the full likelihood, $L^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)} \mid \mathbf{y}^{(i)})$, to form a new population. This procedure should be performed for each $i \in \{1, \ldots, I\}$.

Crossover. The crossover step must be modified from the ordinary genetic algorithm since the lengths of the vectors $\mathbf{v}_{(t)}^1, \ldots, \mathbf{v}_{(t)}^M$ may not all be the same. In this situation, we can still perform crossovers (as described in Section 1.1) on the portion of the vectors that are of the same length, the $\boldsymbol{\phi}$ vectors. The last part of each vector, $(\boldsymbol{\lambda}, i)$, is not involved in the crossover. The following diagram shows a hypothetical one point crossover between two vectors. Again, the subscript denoting the iteration, (t), is suppressed within the expanded vectors for clarity.

$$\begin{aligned} \mathbf{v}_{(t)}^1 &= \{\phi_1^1, \ldots, \phi_{z^*}^1, \phi_{z^*+1}^1, \ldots, \phi_C^1, \boldsymbol{\lambda}^1, i^1\} \underset{\text{Swap Elements}}{\longrightarrow} \{\phi_1^1, \ldots, \phi_{z^*}^1, \phi_{z^*+1}^2, \ldots, \phi_C^2, \boldsymbol{\lambda}^1, i^1\} \\ \mathbf{v}_{(t)}^2 &= \{\phi_1^2, \ldots, \phi_{z^*}^2, \phi_{z^*+1}^2, \ldots, \phi_C^2, \boldsymbol{\lambda}^2, i^2\} \qquad\qquad \{\phi_1^2, \ldots, \phi_{z^*}^2, \phi_{z^*+1}^1, \ldots, \phi_C^1, \boldsymbol{\lambda}^2, i^2\} \end{aligned}$$

Because all of the $\boldsymbol{\phi}$ vectors are the same length and general meaning, crossovers across scales are straightforward. Alternative crossovers are possible for strings of varying lengths (Goldberg et al., 1989); however, the crossover we propose takes full advantage of the common interpretation of $\boldsymbol{\phi}$ vectors on different scales. It is through these multi-scale crossovers that information is shared across resolutions.

Mutation. In the mutation step, the values of some of the parameters in the solution vectors are perturbed. With real valued vectors random noise is generally added to each component with some small probability. The value of the final parameter, i, takes values in $\{1, \ldots, I\}$ and can be mutated to any of these values. Mutating the value of i is unnecessary, but may speed up convergence in some cases. If i is mutated, care should be taken to allow at least one solution at each resolution to remain in the solution space.

As with ordinary genetic algorithms, using an elitist strategy that saves the best solution on the scale of interest from being destroyed through crossover or mutation may increase the speed of convergence (DeJong, 1975).

2.3. Bayesian Model Formulation

In the Bayesian approach, one is typically interested in estimating a posterior distribution for the parameters, rather than simply finding a maximal value. In particular, knowing the full distribution makes estimating and accounting for uncertainty straightforward. The formulation of the likelihood and decomposition of the parameter vector $\boldsymbol{\psi}$ is the same as in Section 2.1. To these we must add a prior distribution for the parameters $(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)})$ which we denote $\pi^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)})$ at each scale i. The resulting posterior distribution for each scale (up to a normalizing constant) is

$$\pi^{(i)}(\boldsymbol{\psi}^{(i)} \mid \mathbf{y}^{(i)}) \;=\; \pi^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)} \mid \mathbf{y}^{(i)}) \;\propto\; L(\boldsymbol{\psi}^{(i)} \mid \mathbf{y}^{(i)})\, \pi^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)}).$$

We note that if the relationship $\boldsymbol{\psi}^{(i)} = g^{(i)}(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)})$ is not deterministic, then we must add another term to the right-hand side for the distribution induced by $g^{(i)}$, and that this term will need to be included when appropriate in expressions in the next section. As before, once the posterior is defined on a single scale, generalization to multiple resolutions is simple. The full posterior is simply the product of the posteriors from each scale.

Because Bayesian inference is based on a posterior *distribution*, rather than likelihood-based *maximization*, in the next section we describe a Markov chain Monte Carlo algorithm that takes advantage of the multi-resolution nature of the problem to efficiently explore the parameter space and draw samples from the posterior.

2.4. Multi-resolution Genetic Algorithm-style MCMC

Though the posterior distributions on different scales are related, they are conditionally independent given the data and could be sampled with separate MCMC chains. However, independent sampling does not allow information to be shared across scales. In order to gain advantages from both the faster mixing speed of the coarse scale chains and greater detail of the fine scale chains, the algorithm must swap information between the scales of the joint posterior distribution.

In our MCMC scheme we run multiple MCMC chains simultaneously with several chains for each component of the joint posterior. Periodically, the realizations from this MCMC implementation attempt to trade information across scales by performing a swap of values. The idea of swapping MCMC realizations across conditionally independent posteriors was proposed by Geyer (1991) and applied to multi-scale

problems by Higdon et al. (2002). The concept of defining an MCMC chain that can move between scales is related to simulated tempering as introduced in Section 1.2. We label the chains $\Gamma_1, \ldots, \Gamma_M$, so that for any $m \in \{1, \ldots, M\}$, Γ_m explores one part of the posterior: $\pi^{(i)}(\boldsymbol{\psi}^{(i)} \mid \mathbf{y}^{(i)})$ for some $i \in \{1, \ldots, I\}$. Unlike the genetic algorithm introduced in Section 2.2 where solution vectors could be allowed to alter the resolution with which they are associated, here once a chain is associated with a certain resolution, it remains associated with that resolution.

The values of the random variables in any chain at any time step determine the current state of that chain. We denote the state of the first chain at time k as $\Gamma_1(\boldsymbol{\phi}^1_{(k)}, \boldsymbol{\lambda}^1_{(k)})$. In this notation, the superscripts without parentheses are necessary to differentiate between different parameter vectors at the same time step (*e.g.*, $\boldsymbol{\phi}^1_{(1)} \neq \boldsymbol{\phi}^2_{(1)}$). A pictorial representation of a portion of the scheme for $M = 3$ might look like this:

$$\begin{array}{ccccccc}
\Gamma_1(\boldsymbol{\phi}^1_{(1)}, \boldsymbol{\lambda}^1_{(1)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_1(\boldsymbol{\phi}^1_{(2)}, \boldsymbol{\lambda}^1_{(2)}) & \underset{\text{SWAP ATTEMPT}}{\longrightarrow} & \Gamma_1(\boldsymbol{\phi}^{1\prime}_{(3)}, \boldsymbol{\lambda}^{1\prime}_{(3)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_1(\boldsymbol{\phi}^{1\prime}_{(4)}, \boldsymbol{\lambda}^{1\prime}_{(4)}) \\
\Gamma_2(\boldsymbol{\phi}^2_{(1)}, \boldsymbol{\lambda}^2_{(1)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_2(\boldsymbol{\phi}^2_{(2)}, \boldsymbol{\lambda}^2_{(2)}) & & \Gamma_2(\boldsymbol{\phi}^{2\prime}_{(3)}, \boldsymbol{\lambda}^{2\prime}_{(3)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_2(\boldsymbol{\phi}^{2\prime}_{(4)}, \boldsymbol{\lambda}^{2\prime}_{(4)}) \\
\Gamma_3(\boldsymbol{\phi}^3_{(1)}, \boldsymbol{\lambda}^3_{(1)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_3(\boldsymbol{\phi}^3_{(2)}, \boldsymbol{\lambda}^3_{(2)}) & & \Gamma_3(\boldsymbol{\phi}^3_{(3)}, \boldsymbol{\lambda}^3_{(3)}) & \overset{\text{MCMC}}{\longrightarrow} & \Gamma_3(\boldsymbol{\phi}^3_{(4)}, \boldsymbol{\lambda}^3_{(4)}).
\end{array}$$

Here, analogously to mutation steps, all chains advance using ordinary within-chain MCMC steps for a fixed number of iterations until a swap step is reached. At that point, several pairs of chains attempt to swap values. In the diagram, a prime mark after a realization (*e.g.*, $\boldsymbol{\lambda}^{1\prime}_{(3)}$) indicates that the variable will take different values depending on whether the swap is successful or not. If the swap is unsuccessful, the primed variables take the same values as their unprimed counterparts at the previous time step (*e.g*, $\boldsymbol{\phi}^1_{(2)} = \boldsymbol{\phi}^{1\prime}_{(3)}$). If the swap is successful, the values of the primed variables depend on the type of swap performed. Each pair performs one of two types of swaps: a full swap or a crossover swap. A full swap occurs with some probability p_{swap} specified by the programmer, and the crossover swap occurs with probability $1 - p_{swap}$. In this particular diagram, only one pair of chains (Γ_1 and Γ_2) attempts a swap.

2.4.1. Full Swap

Full swap steps, derived from simulated tempering, trade entire $\boldsymbol{\phi}$ vectors between chains so that $\boldsymbol{\phi}$ vectors that were associated with one resolution of $\boldsymbol{\psi}$ are then associated with a different resolution of $\boldsymbol{\psi}$. We denote proposed values for a chain with stars (*e.g.*, $\boldsymbol{\phi}^{1*}$ is a proposed $\boldsymbol{\phi}$ vector for chain Γ_1). Thus, for a full swap as depicted in the preceding diagram, $\boldsymbol{\phi}^{1*} = \boldsymbol{\phi}^2_{(2)}$ and $\boldsymbol{\phi}^{2*} = \boldsymbol{\phi}^1_{(2)}$. To create a complete Metropolis-Hastings proposal in the space of the full posterior distribution, $\pi(\boldsymbol{\phi}, \boldsymbol{\lambda} \mid \mathbf{y})$, we must also propose new values $\boldsymbol{\lambda}^*$,

from some distributions $\xi^{(i)}(\boldsymbol{\lambda}^{j*} \mid \boldsymbol{\phi}^{j*}, \mathbf{y}^{(i)})$ which are based on the proposed $\boldsymbol{\phi}$ values for each chain j. The proposal distributions $\xi^{(i)}$ may be different for chains of differing resolution. With these proposals, the acceptance probability for the Metropolis-Hastings step in the diagram is

$$\alpha = \min\left(1, \frac{\pi^{(1)}(\boldsymbol{\phi}^{1*}, \boldsymbol{\lambda}^{1*} \mid \mathbf{y}^{(1)})\pi^{(2)}(\boldsymbol{\phi}^{2*}, \boldsymbol{\lambda}^{2*} \mid \mathbf{y}^{(2)})\xi^{(1)}(\boldsymbol{\lambda}^{1}_{(2)} \mid \boldsymbol{\phi}^{1}_{(2)}, \mathbf{y}^{(1)})\xi^{(2)}(\boldsymbol{\lambda}^{2}_{(2)} \mid \boldsymbol{\phi}^{2}_{(2)}, \mathbf{y}^{(2)})}{\pi^{(1)}(\boldsymbol{\phi}^{1}_{(2)}, \boldsymbol{\lambda}^{1}_{(2)}, \mid \mathbf{y}^{(1)})\pi^{(2)}(\boldsymbol{\phi}^{2}_{(2)}, \boldsymbol{\lambda}^{2}_{(2)}, \mid \mathbf{y}^{(2)})\xi^{(1)}(\boldsymbol{\lambda}^{1*} \mid \boldsymbol{\phi}^{1*}, \mathbf{y}^{(1)})\xi^{(2)}(\boldsymbol{\lambda}^{2*} \mid \boldsymbol{\phi}^{2*}, \mathbf{y}^{(2)})}\right). \quad (1)$$

If the swap is successful the values of $\boldsymbol{\phi}$ that were in chain Γ_2 are moved to Γ_1 while the values that were in Γ_1 are sent to Γ_2. Thus, $\boldsymbol{\phi}^{1}_{(2)} = \boldsymbol{\phi}^{2*} = \boldsymbol{\phi}^{2\prime}_{(3)}$ and $\boldsymbol{\phi}^{2}_{(2)} = \boldsymbol{\phi}^{1*} = \boldsymbol{\phi}^{1\prime}_{(3)}$. In addition, the newly proposed values $\boldsymbol{\lambda}^{1*}$ and $\boldsymbol{\lambda}^{2*}$ replace the old values in each chain. If the swap move is rejected, each chain retains its values of $\boldsymbol{\phi}$ and $\boldsymbol{\lambda}$, and within-chain MCMC updates continue until the next swap step is reached. It should be noted that in this particular sequence, Γ_3 did not attempt to swap values with the other chains. In our implementations chains that do not attempt a swap retain their current values, so $\Gamma_3(\boldsymbol{\phi}^{3}_{(2)}, \boldsymbol{\lambda}^{3}_{(2)}) = \Gamma_3(\boldsymbol{\phi}^{3}_{(3)}, \boldsymbol{\lambda}^{3}_{(3)})$. However, it would be possible to perform a within-chain MCMC update on non-swapping chains while others perform a swap.

2.4.2. Crossover Swap

Despite the advantages offered by the full swap, parts of the space may still be left unexplored due to the slow mixing time of even the coarsest scale. Recently, elements of genetic algorithms have been incorporated into MCMC schemes to make intelligent Metropolis-style proposals in high-dimensional spaces (see, for example, Holmes and Mallick (1998) and Liang and Wong (2001)). Incorporating these genetic algorithm elements into our MCMC can induce a more thorough exploration of the parameter space. We introduce these elements through the second type of swap step – the crossover swap.

In our algorithm, when a swap step is reached and a crossover swap is chosen chains are again paired. As with the full swap, it is possible to pair chains randomly with uniform probability; however, choosing pairs using selection ideas from genetic algorithms may produce better proposals. For non-uniform pairing, we choose two chains without replacement from the full set of M chains with probability proportional to some appropriate measure of fitness. One possible fitness criterion would be the unnormalized posterior density of a realization on its portion of the full posterior. This criterion is valid if priors, likelihoods, and known values are similar on all scales, since the unknown normalizing constants for each scale should be about the same. If the likelihoods are not comparable across scales, then one could select chains in a two-step process: first,

select a scale randomly with probabilities proportional to the number of available (unpaired) chains at that scale; second, select chains from that scale with probabilities proportional to their unnormalized likelihoods. In addition to the first pair of chains chosen, other chains could be paired to perform swaps as well. When pairing in parallel implementations, making as many non-overlapping pairs as possible is most efficient. In our implementations, we choose chains without replacement until as many pairs as possible have been made and attempt swaps on all pairs simultaneously.

After pairing, we perform a uniform crossover on the $\boldsymbol{\phi}$ vectors of the two chains in each pair with the probability of each element being swapped at 50% to create proposals. While in some problems, tuning this probability can increase performance, we have found that 50% is a useful default. After proposing new values for $\boldsymbol{\lambda}$ (again, from $\xi^{(i)}(\boldsymbol{\lambda}^{j*} \mid \boldsymbol{\phi}^{j*}, \mathbf{y}^{(i)})$) we accept or reject the swap according to the Metropolis-Hastings rule with α as in Equation (1). In this way, chains in states of high probability are combined to create intelligent proposals and information is shared across scales simultaneously.

Caution should be used when performing the selection portion of the crossover swap with different fitness functions for different chains since all portions of the posterior are only known up to a normalizing constant. For this reason, fitness evaluations can be made arbitrarily large or small when using a portion of the posterior. However, if similar likelihoods and priors are used for all resolutions of $\boldsymbol{\psi}$, using different but related fitness functions may be appropriate. When in doubt, we recommend weighting chains equally instead of selecting by fitness.

2.4.3. Choosing the Number of Scales

It makes sense to borrow the ideas of simulated tempering to solve multi-resolution problems since the solution space is often complex and multimodal. Multi-scale modeling provides a straightforward way to make heated distributions. Since likelihood evaluations can often be performed faster at coarser resolutions, more iterations of an MCMC can be performed in the same amount of time on a coarser scale. In addition, coarser scales may reduce the problem of multimodality that can plague finer scales.

In simulated tempering schemes, it has been recommended that the temperature scale be chosen such that the acceptance probability of moves between adjacent scales is moderate – about 20% to 40% (Geyer and Thompson, 1995). This recommendation can be used as a guideline for selecting the scales to be used in

modeling multi-scale problems. For example, consider a problem on a fine grid of 32×32 where a 10×10 grid is found to mix well. Some exploratory analysis with a multi-scale MCMC might reveal that the posterior induced by a 32×32 grid performs swaps successfully with the posterior induced by a 22×22 grid about 30% of the time and that the posterior induced by the 22×22 grid performs swaps successfully with the posterior induced by a 10×10 grid about the same amount. As a result, we would use these three scales in our model.

2.4.4. Choosing the Number of Chains

In the MCMC scheme we have outlined, running several chains at each resolution is possible. Choosing an appropriate number of chains to run at each resolution is important. If there are too few chains, the algorithm may not have enough diversity in the population of chains to allow rapid exploration. At the other end of the spectrum, having too many chains wastes computing effort. For genetic algorithms, setting $M = 2 \times L$, that is, setting the number of solutions in the population at any time to be twice the length of the vector on which crossovers are performed, works well. Following this guideline for our multi-resolution genetic algorithm implies that $2 \times C$ solutions should be in the population at any time. For MCMC implementations, premature convergence as found in genetic algorithms with too few solution vectors is not a large threat, so fewer than $2 \times C$ chains may be used. In our implementations, we generally use only a couple of chains at each resolution.

2.5. Parallel Computing

Obviously, running so many chains can be a huge computational burden. However, these algorithms can be parallelized easily. For the multi-resolution genetic algorithm, independent execution of mutation steps and evaluations of the fitness function are possible for each element of the population. For the MCMC algorithm, factorization of the posterior distribution into conditionally independent components allows simultaneous sampling of those components. Synchronizing is only important for the swapping/crossover steps.

Unfortunately, implementing this algorithm in a parallel way can lead to some inefficiencies. For instance, some processors may have to wait for other processors to reach a certain point before a trade may take place, leading to idle processors and wasted computing time. One solution is to try to time the length

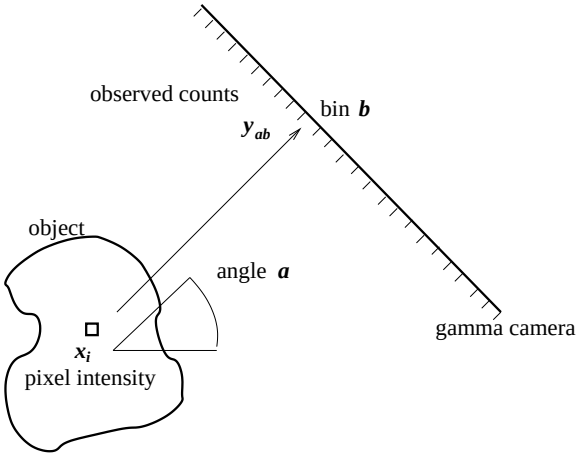

Fig. 2. SPECT reconstructions: A pixelated object emits photons with location dependent intensities x_i. The gamma camera array records photon emissions from angle a in bin b as y_{ab}.

of run for each processor so that they all work approximately the same amount of time before a swap is considered. For instance, coarser scales could do two cycles of within-scale parameter updates (mutation and selection for the multi-scale genetic algorithm or standard MCMC updates in the multi-scale MCMC algorithm) in the time the finest scale does one. Further information on parallel MCMC concerns can be found in Rosenthal (2000) and the references therein.

3. Example: SPECT

In single photon emission computed tomography (SPECT), the goal is to construct a photon emission intensity map of an object from photon counts detected by a gamma camera. Here we consider reconstructing a two-dimensional object. As the object emits photons, the gamma camera records the locations of photon hits along the camera array. The gamma camera array rotates around the object, recording counts at 120 positions indexed by the angle a (see Figure 2). At a given camera position, the counts are recorded as counts over 128 bins along the surface of the camera. The data, from Higdon et al. (2002), consist of counts y_{ab} obtained from bin b of the gamma camera while it was positioned at angle a. Lead columnators on the camera absorb any photon that fails to hit the camera at nearly a right angle. A photon may be scattered, absorbed, miss the gamma camera, or otherwise fail to be detected, thus the columnators, along with the physical characteristics of the object determine the chance that a photon emitted from pixel i hits bin b

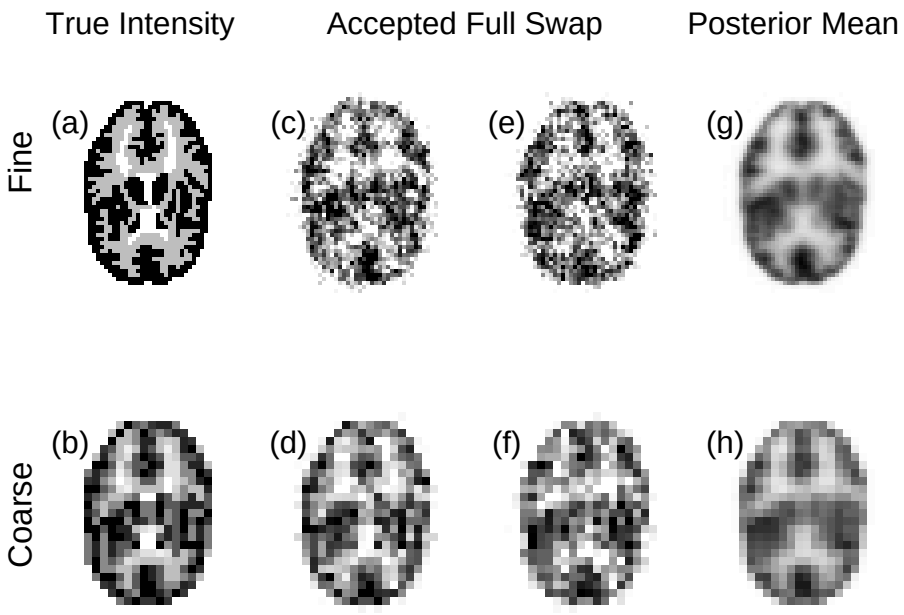

Fig. 3. Multi-scale MCMC for the SPECT example. (a) true emission intensities; (b) coarsened version of the true intensities; (c) & (d) snapshot of current values for $\boldsymbol{\psi}^1$ and $\boldsymbol{\psi}^2$ during the MCMC run; (e) & (f) proposed fine and coarse images for a full swap from the images in (c) & (d); (g) posterior mean for $\boldsymbol{\psi}^1$; (h) posterior mean for $\boldsymbol{\psi}^2$.

while the camera is positioned at angle a. Here we treat the probabilities p_{abi} that were used to generate the data as known. The true image source intensity is shown in Figure 3(a). Inference is ultimately desired at a fine (128×128) resolution, but computation is much faster at coarser resolutions. Thus we apply our multi-resolution MCMC algorithm to realize significant gains in efficiency from improved mixing of the Markov chains.

The counts y_{ab} are modeled as coming from a Poisson distribution with mean $\lambda_{ab} = \sum_i \psi_i p_{abi}$, or in matrix form, $\boldsymbol{\lambda}_{n\times 1} = \mathbf{P}_{n\times m}\boldsymbol{\psi}_{m\times 1}$, where $n = 128 \times 120$ is the number of bins times the number of angles and m is the number of pixels. Given the object intensities ψ_i, the likelihood can be written as

$$L(\mathbf{y}|\boldsymbol{\psi}) \propto \prod_{a,b} \lambda_{ab}^{y_{ab}} e^{-\lambda_{ab}} .$$

For more details on the derivation of the likelihood for SPECT, see Vardi et al. (1985).

Our analysis here uses only two resolutions, a fine scale of $m^{(1)} = 128 \times 128$ and a coarser scale of $m^{(2)} = 64 \times 64$. Each coarse pixel corresponds to four fine pixels, so the intensity of a coarse pixel should be the sum of the corresponding fine pixel intensities. Since emission intensities are zero outside of the brain,

there are uninteresting pixels in the full grid. Thus, as in Higdon et al. (2002), only a sub-grid consisting of the interior of the brain image is analyzed here. The key is the four-to-one ratio of pixels as the resolution changes.

Following the analysis in Higdon et al. (2002), we put a symmetric first-order Gaussian Markov Random Field (MRF) prior on the intensities at each scale:

$$\pi(\boldsymbol{\psi}|\theta) \;\propto\; \theta^{\frac{m}{2}} \exp\left\{ -\tfrac{1}{2}\theta \sum_{i\sim j} (\psi_i - \psi_j)^2 \right\} \;=\; \theta^{\frac{m}{2}} \exp\{-\tfrac{1}{2}\theta \boldsymbol{\psi}^T \mathbf{W} \boldsymbol{\psi}\} \tag{2}$$

where $i \sim j$ denotes pairs (i, j) of adjacent (neighboring) pixels and θ is a precision parameter that controls the overall smoothness of the field. In this case, the matrix $\mathbf{W}$ contains off-diagonal entries $w_{ij} = -1$ if pixel i is adjacent to pixel j and 0 otherwise, and diagonal elements w_{ii} equal to the number of neighbors of pixel i (four unless i is on the border of the image). Diffuse gamma priors are assigned to the precision parameters θ^i for each scale.

To put this problem into the notation of this paper, let $\boldsymbol{\phi}^{(2)} = \boldsymbol{\psi}^{(2)}$ and let $\boldsymbol{\phi}^{(1)}$ be the mean of each block of four fine pixels corresponding to a single coarse pixel, so that $\boldsymbol{\phi}^{(1)}$ and $\boldsymbol{\phi}^{(2)}$ have the same dimension. $\boldsymbol{\lambda}^{(1)}$ is then defined as the values of the upper left, upper right, and lower left pixels for each of the blocks of four fine pixels (since knowing those three and their mean determines the fourth pixel for each block). $\boldsymbol{\lambda}^{(1)}$ also contains $\theta^{(1)}$, and $\boldsymbol{\lambda}^{(2)} = \theta^{(2)}$. Finally, $\pi^{(i)}$ is the product of the the prior on $\theta^{(i)}$ and the prior induced on $(\boldsymbol{\phi}^{(i)}, \boldsymbol{\lambda}^{(i)})$ by Equation (2).

For a particular resolution, we carry out standard MCMC as described in (Weir, 1997) or (Higdon et al., 1997). Here a full swap consists of exchanging all $\boldsymbol{\phi}$ values, sampling the individual pixels in $\boldsymbol{\lambda}^{(1)}$ from their joint complete conditional, sampling $\theta^{(1)}$ and $\theta^{(2)}$ from their complete conditionals, and then accepting or rejecting via Metropolis-Hastings with probability as in Equation (1). A crossover swap exchanges only a subset of $\boldsymbol{\phi}$ values and samples only the associated pixels in $\boldsymbol{\lambda}^{(1)}$, as well as $\theta^{(1)}$ and $\theta^{(2)}$. About one in eight full swaps and about one in five crossover swaps were accepted. Figure 3 shows an accepted full swap along with coarse and fine scale posterior mean images. Autocorrelation functions are shown in Figure 4 for a representative fine pixel under two posterior sampling schemes. The multi-scale MCMC approach of this paper yields estimated autocorrelation times (Sokal, 1987) that are about a third of those obtained under a standard fine scale-only MCMC algorithm after adjusting for comparable CPU usage. Thus we achieve a significant computational gain from the algorithms of this paper. We note that a possible alternative

crossover scheme would swap clusters of coarse pixels, such as a 4×4 block, which would preserve more of the local structure of the MRF prior.

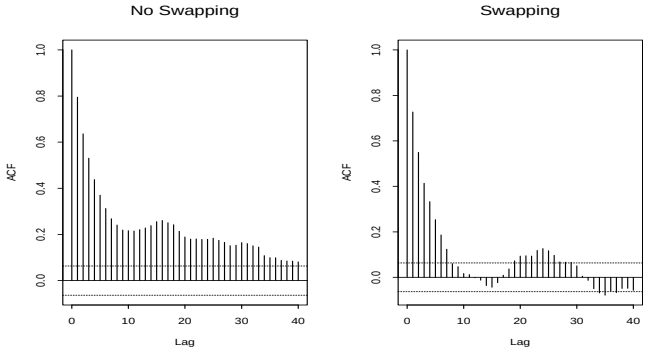

Fig. 4. Autocorrelation plots for the intensity of an interior pixel in the SPECT application under the single-chain standard MCMC at the fine scale only (left) and the multi-scale MCMC approach (right).

4. Example: Multi-scale Flow Problems

As another example, we consider multi-scale modeling of a groundwater hydrology problem known as the inverse problem. Here flow experiment data are used to determine subsurface geological properties of an aquifer, in particular, the permeabilities. Finding the permeability configuration most consistent with the data involves matching the observed flows to the output of flow experiments on the proposed permeability configuration as determined by computer simulators. Such simulators solve systems of differential equations numerically. We use the S3D streamtube code developed by King and Datta-Gupta (1998). Note that as an inverse problem, no direct permeability measurements may be available, and all inference must rely on the indirect information in the flow data. An overview of the inverse problem can be found in Yeh (1986). Under a Bayesian approach, probabilities can be assigned to proposed permeability configurations (Craig et al., 1996; Oliver et al., 1997; Lee et al., 2002) allowing for full accounting of uncertainty, a traditionally difficult task.

4.1. Inverse Problem Setup

Our example involves a two-dimensional permeability field with injector and producer wells arranged in an inverted nine-spot pattern as in Figure 5. Water is injected through the center well and extracted at the

eight producer wells located at the midpoint of each of the four edges and at the four corners. Injection and production continue until the system reaches equilibrium. Next, a tracer is injected at the center well. The *breakthrough time* for a specific producer well is the amount of time that elapses between the injection of the tracer and its arrival at that producer well. Figure 5 includes the breakthrough times in days at each of the eight producer wells from our simulated example. For instance, the tracer took 31.89 days to reach the producer well in the top right corner. These breakthrough times are the only data we have available for determining the permeability configuration of the entire aquifer.

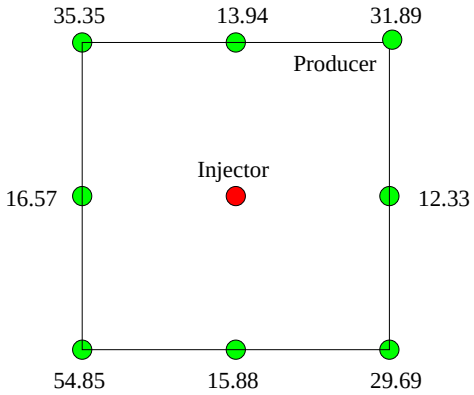

Fig. 5. Inverted nine-spot pattern of wells. The well in the center is an injector and the ones on the edges are producers. The breakthrough time for the tracer is given at each producer well.

We are interested in estimating the values of the permeabilities within the square delineated by the producer wells. We denote this region of interest D. Permeabilities theoretically vary continuously in space, giving us an infinite dimensional problem that can only be simplified by making several assumptions. We note that these are the standard assumptions in the hydrology literature. The first assumption is that we can divide the field up by imposing a grid onto it and assuming that within each grid block permeability is uniform in all directions. Our forward simulator requires permeabilities to be input on a grid, so this discretization makes sense. Here we assume that a 32×32 grid is the finest grid at which we need permeability estimates. We also note that the simulator is more accurate for finer grids, but runs more slowly.

The second assumption concerns the spatial structure of the permeability field. We use a Gaussian process prior to regularize the field. The full details are in Section 4.3. Such assumptions are necessary because the problem is ill-posed, in that without additional structure, multiple solutions (permeability configurations) can match the flow data closely. Note that this ill-posedness leads to severe multi-modality

problems in the likelihood, something that is addressed by the multi-resolution approach of our algorithms.

4.2. Likelihood

The likelihood of the data is based on the discrepancy between the true breakthrough times and the fitted breakthrough times for a given permeability configuration found by running the simulator. We use the standard likelihood for this problem, an *iid* Gaussian error structure:

$$L(\boldsymbol{\psi} \mid \mathbf{y}) \quad \propto \quad \exp\left\{-\frac{1}{2\sigma_y^2}\sum_{h=1}^{8}(y_h - \hat{y}_h(\boldsymbol{\psi}))^2\right\}. \tag{3}$$

In Equation (3), $\boldsymbol{\psi}$ denotes the vector of log-permeabilities, and $\hat{y}_h(\boldsymbol{\psi})$ denotes the breakthrough time at well h determined by the simulator for a specific permeability configuration, $\boldsymbol{\psi}$. Log-permeabilities are used for modeling since the distribution of permeabilities is typically assumed to be log-normal. We denote the vector of true breakthrough times $\mathbf{y} = (y_1, \ldots, y_8)$. In this example we take σ_y^2, the variance between the true and the fitted breakthrough times, to be known. Note that the same breakthrough time measurements apply regardless of the resolution of the permeability grid, so Equation (3) will apply at each scale.

4.3. Spatial Process Convolution Prior for Log-Permeabilities

Exploring the space of satisfactory permeability configurations without placing some sort of structure on the log-permeabilities themselves would prove difficult. The Bayesian framework allows structure to be added to the permeability parameters via a prior distribution. The standard approach in the literature is to impose a Gaussian Process structure on the log-permeabilities. For the algorithm we implement here, the problem is simplified by using a process convolution representation, as introduced in Section 2.1.2.

Process convolutions are a simple way of approximating a continuous surface over a spatial domain using a finite number of parameters (Higdon, 2001). The first step in creating the surface is the definition of a lattice over the field D, the region over which we define the continuous surface, that extends beyond the edges of D as in Figure 1. For this application, we choose a 5×5 lattice. We denote the locations in the lattice $\omega_1, \ldots, \omega_C$. At each of these locations, we define an independent random variable $\phi(\omega_j)$ which is normally distributed with mean zero and variance σ_x^2. By convolving this white noise process with some kernel k, we can define a continuous surface over the entire region, D. We also specify an overall mean level μ. Thus for a particular resolution i, the value of the surface $\boldsymbol{\psi}^{(i)}$ at any grid location s is the discretized

form of this continuous surface calculated as

$$\psi^{(i)}(s \mid \boldsymbol{\phi}) = \mu^{(i)} + \Delta\left(\sum_{j=1}^{C} k(d(s,\omega_j))\phi^{(i)}(\omega_j)\right). \tag{4}$$

In this equation, $d(s_i, s_j)$ denotes the Euclidean distance between locations s_i and s_j. The discretization function Δ assigns the value of a grid block to be the value of the continuous surface at the center of that grid block. Thus the log-permeabilities $\boldsymbol{\psi}$ are a deterministic transformation of the bases $\boldsymbol{\phi}$.

We use a circular normal distribution with a fixed variance for the kernel k, which leads to an isotropic process with a Gaussian covariogram. For more details on the relationship between the choice of kernel and the induced correlation structure, see Thiébaux and Pedder (1987), Barry and Ver Hoef (1996), or Higdon (2001). Note that we must assume that the variance of the kernel is known, as in hydrology inverse problems such as this one, generally insufficient information exists in the data to fit such correlation parameters.

One advantage of the process convolution approach is that the effect of changing a value of the white noise process, $\boldsymbol{\phi}$, on the overall surface is straightforward, as the effect is local. Such a feature is important in performing crossover swaps since strong dependencies between values of $\boldsymbol{\phi}$ in adjacent locations in the grid would make acceptance of swaps difficult.

For the values of the white noise process, $\phi(\omega_1), \ldots, \phi(\omega_C)$, we assign independent normal priors centered at zero with variance σ_x^2. We place a diffuse but proper inverse gamma prior on σ_x^2. Also, we take the overall mean of the process, μ, to be unknown, though we do typically have prior information regarding its value from engineers familiar with the site. In this example, we place a normal prior with mean 5.7 and variance .25 on μ.

In our implementation, we choose full and crossover swaps with equal probability, that is $p_{swap} = .5$. In crossover swaps, uniform crossovers are used with each element given a probability .5 of making a swap. Swaps are attempted every 5 iterations of the finest scale. One iteration consists of updates of all values of ϕ, μ, and σ_x^2.

4.4. Multi-scale Model

The likelihood and posterior are highly multimodal in this problem. In order to speed up convergence and enhance posterior exploration, we fit the model on several scales simultaneously. In Section 4.1 we mentioned that we would like the finest resolution to be a 32×32 grid. Following the ideas in Section 2.4.3, we also

Table 1. Trade probabilities for full and crossover swaps.

Full Swaps

	20×20	16×16
32×32	0.442	0.485
20×20	–	0.511

Crossover Swaps

	32×32	20×20	16×16
32×32	0.305	0.332	0.316
20×20	–	0.382	0.316
16×16	–	–	0.335

use a 20×20 and a 16×16 grid. Preliminary exploration finds that a 32×32 grid performs a successful full swap with a 20×20 grid approximately 39% of the time such a swap is attempted. A 20×20 and a 16×16 grid succeed in performing full swaps about 56% of the time. Intermediate scales swap more frequently, but are not as distinct, so they would not be likely to greatly improve efficiency. Below a resolution of 16×16, the simulator begins to noticeably lose accuracy because of the coarseness of the grid.

Using the prior and likelihood defined in the previous sections, the formula for the posterior at a single scale j is

$$\pi^{(j)}(\psi^{(j)}(\boldsymbol{\phi}^{(j)}), \mu, \sigma_x^{2(j)} \mid \mathbf{y}^{(j)}) \propto L^{(j)}(\psi^{(j)}(\boldsymbol{\phi}^{(j)}) \mid \mathbf{y}^{(j)}) \pi(\boldsymbol{\phi}^{(j)} \mid \mu^{(j)}, \sigma_x^{2(j)}) \pi(\sigma_x^{2(j)}) \pi(\mu)^{(j)}.$$

As before, we assume that the solution for each scale is independent of the others conditional on the data, so the full posterior distribution is the product of the posteriors for the individual scales. We also assume that the priors for all parameters are the same regardless of the scale on which the model is being fit.

4.5. Results

We ran the MCMC algorithm for 10,000 iterations after a burn-in period of 1000 iterations. Two chains were run at each of the three resolutions. Table 4.5 shows the proportion of attempted swaps between the scales that were successful. Figure 6 shows posterior means for the permeability field on each scale along with the true permeability field, with the posterior means closely matching the truth. Figure 7 displays acf plots for a single element of the $\boldsymbol{\phi}$ vector under single-chain and multi-scale MCMC. The autocorrelation dies off much faster using the swapping approach than in the ordinary MCMC algorithm, indicating that the multi-resolution approach greatly increases the mixing of the chain.

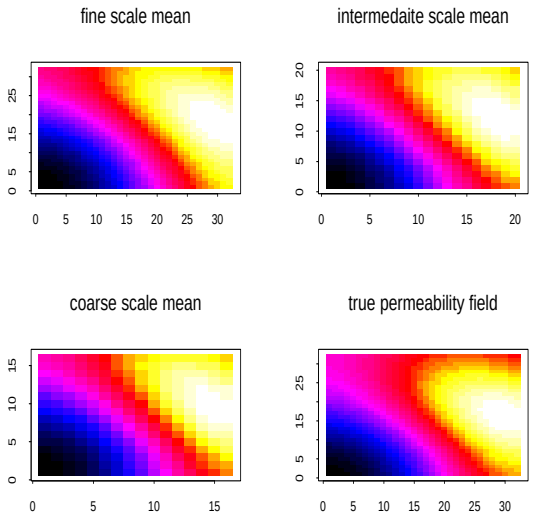

Fig. 6. Posterior means and true permeability field.

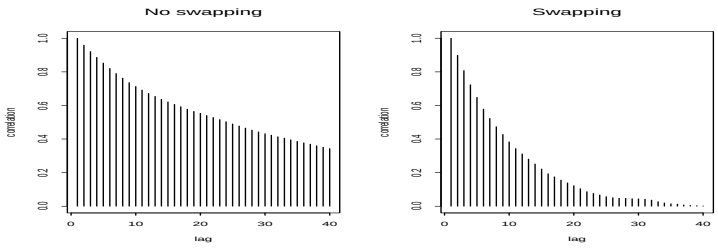

Fig. 7. Autocorrelation functions for a single element of the ϕ vector under single-chain standard MCMC at the fine scale only (left) and multi-scale MCMC (right).

5. Conclusions

By combining multi-resolution models with genetic algorithms, this paper presents methodology for improving speed of convergence or posterior mixing for analyses that may otherwise be plagued by multimodality or other problematic structures in the likelihood or posterior. In particular, standard algorithms for analyzing flow data, as in Section 4, typically have severe problems with local maxima in the likelihood, and the multi-scale approach helps immensely. The methods of this paper are also applicable for the analysis of truly multi-scale data.

While the examples presented herein involve spatial data, we note that the methodology is more general. Beyond obvious extensions such as time series data, other candidate data types include data with ordered categories, and data with a hierarchical categorical structure.

Acknowledgments

The authors would like to thank Jennifer Pittman for her many helpful comments and suggestions. This work was partially supported by National Science Foundation grant DMS 9873275.

References

Bäck, T. (1993). Optimal mutation rates in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann: San Mateo, CA.

Barry, R. P. and J. M. Ver Hoef (1996). Blackbox kriging: Spatial prediction without specifying variogram models. *Journal of Agricultural, Biological, and Environmental Statistics 1*, 297–322.

Bouman, C. and B. Liu (1991). Multiple resolution segmentation of textured images. *IEEE Transactions on Pattern Analysis and Machine Learning 13*(2), 99–113.

Chatterjee, S., M. Laudato, and L. A. Lynch (1996). Genetic algorithms and their statistical applications: an introduction. *Computational Statistics and Data Analysis 22*, 633–651.

Craig, P. S., M. Goldstein, A. H. Seheult, and J. A. Smith (1996). Bayes linear strategies for history matching of hydrocarbon reservoirs. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (Eds.), *Bayesian Statistics 5*, pp. 69–95. Oxford: Clarendon Press.

DeJong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics. Proceedings of the 23rd Symposium on the Interface*, pp. 156–163. Interface Foundation of North America (Fairfax Station, VA).

Geyer, C. J. and E. A. Thompson (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association 90*, 909–920.

Goldberg, D. E., B. Korb, and K. Deb (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems 3*, 493–530.

Goodman, J. and A. D. Sokal (1989). Multigrid Monte Carlo method. *Physical Review Letters D 40*, 2035–2072.

Higdon, D. (2001). Space and space-time modeling using process convolutions. Technical Report 01-03, Duke University, Institute of Statistics and Decision Sciences.

Higdon, D., H. Lee, and Z. Bi (2002). A Bayesian approach to characterizing uncertainty in inverse problems using coarse and fine scale information. *IEEE Transactions on Signal Processing*. To appear.

Higdon, D. M., V. E. Johnson, J. E. Bowsher, T. G. Turkington, D. R. Gilland, and R. J. Jaszczack (1997). Fully Bayesian estimation of Gibbs hyperparameters for emission computed tomography data. *IEEE Transactions on Medical Imaging 16*, 516–526.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Holmes, C. C. and B. K. Mallick (1998). Parallel Markov chain Monte Carlo sampling. Technical report, Imperial College.

King, M. J. and A. Datta-Gupta (1998). Streamline simulation: A current perspective. *In Situ 22*(1), 91–140.

Lakshmanan, S. and H. Derin (1993). *Markov Random Fields: Theory and Application*, Chapter 6. Academic Press, Inc.

Lee, H., D. Higdon, Z. Bi, M. Ferreira, and M. West (2002). Markov random field models for high-dimensional parameters in simulations of fluid flow in porous media. *Technometrics*. To appear.

Liang, F. and W. H. Wong (2001). Real parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association 96*(454), 653–666.

Liu, J. S. and C. Sabatti (1999). Simulated sintering: Markov chain Monte Carlo with spaces of varying dimensions (with discussion). In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (Eds.), *Bayesian Statistics 6*, pp. 389–413. Oxford University Press.

Liu, J. S. and C. Sabatti (2000). Generalised Gibbs sampler and multigrid Monte Carlo for Bayesian computation. *Biometrika 87*, 353–369.

Marinari, E. and G. Parisi (1992). Simulated tempering: a new Monte Carlo scheme. *Europhysics Letters 19*, 451–458.

Oliver, D. S., L. B. Cunha, and A. C. Reynolds (1997). Markov chain Monte Carlo methods for conditioning a permeability field to pressure data. *Mathematical Geology 29*(1), 61–91.

Rosenthal, J. S. (2000). Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics 4*(2), 207–236.

Sokal, A. D. (1987). Monte Carlo methods in statistical mechanics: foundations and new algorithms. *Cours de Troisiéme Cycle de la Physique en Suisse Romande*. Lausanne.

Thiébaux, H. J. and M. A. Pedder (1987). *Spatial Objective Analysis with Applications in Atmospheric Science*. London: Academic Press.

Vardi, Y., L. Shepp, and L. Kaufman (1985). A statistical model for positron emission tomography. *Journal of the American Statistical Association 80*, 8–25.

Weir, I. (1997). Fully Bayesian reconstructions from single photon emission computed tomography. *Journal of the American Statistical Association 92*, 49–60.

Yeh, W. W. (1986). Review of parameter identification in groundwater hydrology: the inverse problem. *Water Resources Research 22*, 95–108.

Yoon, S., A. H. Malallah, A. Datta-Gupta, D. W. Vasco, and R. A. Behrens (1999). A multiscale approach to production data integration using streamline models. Society of Petroleum Engineers 1999 Annual Technical Conference, SPE 56653.