Normalizations of the Proposal Density in Markov Chain Monte Carlo Algorithms

Antoine E. Zambelli

1Anderson School of Management - University of California, Los Angeles

October 14, 2014

Abstract

We explore the effects of normalizing the proposal density in Markov Chain Monte Carlo algorithms in the context of reconstructing the conductivity term $K$ in the 2-dimensional heat equation, given temperatures at the boundary points, $d$.

We approach this nonlinear inverse problem by implementing a Metropolis-Hastings Markov Chain Monte Carlo algorithm. Markov Chains produce a probability distribution of possible solutions conditional on the observed data. We generate a candidate solution $K'$ and solve the forward problem, obtaining $d'$. At step $n$, with some probability $\alpha$, we set $K_{n+1} = K'$.

We identify certain issues with this construction, stemming from large and fluctuating values of our data terms. Using this framework, we develop normalization terms $z_0, z$ and parameters $\lambda$ that preserve the inherently sparse information at our disposal. We examine the results of this variant of the MCMC algorithm on the reconstructions of several 2-dimensional conductivity functions.

Keywords: Ill-posed, Inverse Problems, MCMC, Normalization, Numerical Analysis.
Classification: 35R30, 65C40

*This is a preprint, presented at the 6th International Conference on Numerical Analysis, Chania, Greece, Sept 2-5 2014*
1 Introduction

The idea of an inverse problem is to reconstruct, or retrieve, information from a set of measurements. In many problems, the quantities we measure are different from the ones we wish to study; and this set of \( d \) measurements may depend on several elements. Our goal is thus to reconstruct, from the data, that which we wanted to study. In essence, given an effect, what is the cause? For example: If you have measurements of the temperature on a surface, you may want to find the coefficient in the heat equation.

The nonlinearity and ill-posedness of this problem lends itself well to Markov Chain Monte Carlo algorithms. We detail this algorithm in later sections, but we note now that there has been much work done on Metropolis-Hastings MCMC algorithms. However, much of it has been trying to determine optimal proposal densities \([3],[5]\). Luengo and Martino \([3]\) treat this idea by defining an adaptive proposal density under the framework of Gaussian mixtures. Our work, however, is focused on improving the reconstruction given a proposal density.

We take no views on the optimality of the structure of the proposal density in our case, which we take from \([1]\). We simply observe possible improvements to this density by normalizing it’s terms through context-independent formulations. Eventually, we would like to implement the GM-MH algorithm of \([3]\) on our proposal density, and provide a rigorous definition of our construction in an analogous manner to their work.

The paper is structured as follows. We first present the framework of our problem in the subsection below. Section 2 presents the MHMCMC algorithm and proposal densities along with non-normalized results. The error analysis of those results (in Section 3) motivates this work while Sections 4 to 6 present the new constructions and associated results.

1.1 Heat Diffusion

In this problem, we attempt to reconstruct the conductivity \( K \) in a steady state heat equation of the cooling fin on a CPU. The heat is dissipated both by conduction along the fin and by convection with the air, which gives rise to our equation:

\[
\frac{u_{xx}}{K} + \frac{u_{yy}}{\delta} = \frac{2H}{K\delta}u
\]  

(1.1.1)

with \( H \) for convection, \( K \) for conductivity, \( \delta \) for thickness and \( u \) for temperature. The CPU is connected to the cooling fin along the bottom half of the left edge of the fin. We use standard Robin Boundary Conditions with

\[
Ku_{\text{normal}} = Hu
\]  

(1.1.2)

Our data in this problem is the set of boundary points of the solution to (1.1.1), which we compute using a standard Crank-Nicolson scheme for an \( n \times m \) mesh (here \( 20 \times 20 \)). We denote the correct value of \( K \) by \( K_{\text{correct}} \) and the data by \( d \). In order to reconstruct \( K_{\text{correct}} \), we will take a guess \( K' \), solve the forward problem using \( K' \), obtaining \( d' \), and compare those boundary points to \( d \) by implementing the Metropolis-Hastings Markov Chain Monte Carlo algorithm (or MHMCMC).

2 Metropolis-Hastings MCMC

Markov Chains produce a probability distribution of possible solutions (in this case conductivities) that are most likely given the observed data (the probability of reaching the next step in the chain is entirely determined by the current step). The algorithm is as follows (see \([1]\)). Given \( K_n \), \( K_{n+1} \) can be found using the following:

1. Generate a candidate state \( K' \) from \( K_n \) with some distribution \( g(K'|K_n) \). We can pick any \( g(K'|K_n) \) so long as it satisfies

   (a) \( g(K'|K_n) = 0 \Rightarrow g(K_n|K') = 0 \)
(b) \( g(K'|K_n) \) is the transition matrix of the Markov Chain on the state space containing \( K_n, K' \).

2. With probability

\[
\alpha(K'|K_n) = \min \left\{ 1, \frac{Pr(K'|d)g(K_n|K')}{Pr(K_n|d)g(K'|K_n)} \right\}
\]

set \( K_{n+1} = K' \), otherwise set \( K_{n+1} = K_n \) (ie. accept or reject \( K' \)). Proceed to the next iteration.

More formally, if \( \alpha > u \sim U[0,1] \), then \( K_{n+1} = K' \). Using the probability distributions of our example, \( \text{(2.0.3)} \) becomes

\[
\alpha(K'|K_n) = \min \left\{ 1, e^{-\frac{1}{2\sigma^2} \sum_{i,j=1}^{n,m} [(d_{ij} - d'_{ij})^2 - (d_{ij} - d_{n_{ij}})^2]} \right\}
\]

where \( d' \) and \( d_n \) denote the set of boundary temperatures from \( K' \) and \( K_n \) respectively, and \( \sigma = 0.1 \). To simplify \( \text{(2.0.4)} \), collect the constants and separate the terms relating to \( K' \) and \( K_n \):

\[
\frac{-1}{2\sigma^2} \sum_{i,j=1}^{n,m} [(d_{ij} - d'_{ij})^2 - (d_{ij} - d_{n_{ij}})^2] = \frac{-1}{2} [f' - f_n]
\]

\[
= -(D_1)
\]

Now, \( \text{(2.0.4)} \) reads

\[
\alpha(K'|K_n) = \min \left\{ 1, e^{-D_1} \right\}
\]

Note that we are taking this formulation as given, and that the literature mentioned above (most notably Gaussian Mixture based algorithms) would be going from \( \text{(2.0.3)} \) to \( \text{(2.0.4)} \) perhaps differently.

### 2.1 Generating \( K' \)

To generate our candidate states, we will perturb \( K_n \) by a uniform random number \( \omega \in [-0.005, 0.005] \). In the simplest case, where we are dealing with a constant \( K_{\text{correct}} \), then we could proceed by changing every point in the mesh by \( \omega \), and the algorithm converges rapidly.

Looking at non-constant conductivities forces us to change our approach. If we simply choose to change one randomly chosen point at a time, then we have a systemic issue with the boundary points, which exhibit odd behavior and hardly change value. To sidestep this, we will change a randomly chosen grid \( (2 \times 2) \) of the mesh at once. Thereby pairing up the troublesome boundary points with the well-behaved inner points.

### 2.2 Priors

While a gridwise change enables us to tackle non-constant conductivities, two issues remain. The first is that our reconstructions are still marred with “spikes” of instability. The second, more profound, is that the ill-posedness of the problem means there are in fact infinitely many solutions, and we must isolate the correct one. This brings us to the notion of priors. These can be thought of as weak constraints imposed on our reconstructions. However, we do not wish to rule out any possibilities, keeping our bias to a minimum. So we define

\[
T' = \sum_{j=1}^{n} \sum_{i=2}^{m} (K'(i,j) - K'(i-1,j))^2 + \sum_{i=1}^{n} \sum_{j=2}^{m} (K'(i,j) - K'(i,j-1))^2
\]

\[
T_n = \sum_{j=1}^{n} \sum_{i=2}^{m} (K_n(i,j) - K_n(i-1,j))^2 + \sum_{i=1}^{n} \sum_{j=2}^{m} (K_n(i,j) - K_n(i,j-1))^2
\]

let \( D_2 = T' - T_n \), and modifying \( \text{(2.0.5)} \), we obtain

\[
\alpha_c(K'|K_n) = \min \left\{ 1, e^{-\lambda_1 D_1 - \lambda_2 D_2} \right\}
\]

By comparing the smoothness of \( K' \) not in an absolute sense, but relative to the last accepted guess, we hope to keep as many solutions as possible open to us, while ensuring a fairly smooth result. We introduce
one additional prior, this time imposing a condition on the gradient of our conductivity. The author explores the notion of priors more fully in [7], but much as we take the proposal density as given, the aim of this paper is not to examine priors per se. So we look at the mixed partial derivative of our candidate state and compare it to that of the last accepted guess

\begin{align}
M' &= \sum_{j=1}^{n} \sum_{i=2}^{m} (K'_{xy}(i, j) - K'_{xy}(i, j - 1))^2 \\
&\quad + \sum_{i=1}^{m} \sum_{j=2}^{n} (K'_{xy}(i, j) - K'_{xy}(i, j - 1))^2 \\
\tag{2.2.4}
\end{align}

\begin{align}
M_n &= \sum_{j=1}^{n} \sum_{i=2}^{m} (K_{nx}(i, j) - K_{nx}(i, j - 1))^2 \\
&\quad + \sum_{i=1}^{m} \sum_{j=2}^{n} (K_{nx}(i, j) - K_{nx}(i, j - 1))^2 \\
\tag{2.2.5}
\end{align}

where \(K'_{xy}\) and \(K_{nx}\) are computed using central and forward/backward finite difference schemes. We let \(D = M' - M_n\) and modify (2.0.5) to get

\begin{equation}
\alpha_s(K' \mid K_n) \equiv \min \{1, e^{-\lambda_1 D_1 - \lambda_3 D_3} \} \\
\tag{2.2.6}
\end{equation}

We now take the acceptance step of our algorithm as

\begin{equation}
\alpha = \max \{\alpha_c, \alpha_s\} \\
\tag{2.2.7}
\end{equation}

So the algorithm seeks to satisfy at least one of our conditions, though not necessarily both. We present some preliminary results in Figure 1 and Figure 2 below. Note that we are clearly on the right path, with the algorithm approaching it’s mark, but not to a satisfying degree.

![Figure 1: Reconstruction of a tilted plane with priors, 10 million iterations.](image)

3 Error Analysis

Our work so far has looked at qualitative improvements to our reconstructions, now we seek to quantify those improvements and the performance of the algorithm in general. Several metrics can be used for this purpose, but we will focus our writeup on the following: the difference between the data and the output using our guess \(\delta\), given by

\begin{equation}
\delta = (\delta_1 \cdots \delta_n) \quad \text{with} \quad \delta_i = \sum (d - d'_i)^2
\end{equation}
3.1 $\delta$, $\beta$, $\Gamma$ Results

The results of tests involving these parameters reveals some interesting information (see Figure 3). $\beta$ decreases, as expected, at a decreasing rate over time, slowing down around 6 – 7 million iterations, which seems in line with the qualitative results.

On the other hand, $\delta$ decreases much more rapidly. The difference between the data and simulated temperatures becomes very small starting at as early as 250000 iterations. In a sense, this fits with the problem of ill-posedness, the data is only useful to a certain degree, and it will take much more to converge to a solution (and we have been converging beyond 250k iterations). The most important result, however, comes from $\Gamma$. If we fit a line to our step function, we get slopes of 0.95 or more. This means we are accepting nearly every guess. While this could be troubling on its own, the fact that we are accepting at a constant rate as well is indicative of a deeper problem in our method.

Given that $\Gamma$ is dependent solely on the likelihood of accepting a guess, we take a look at $\alpha$ directly. What we find is that $\alpha$ is evaluating at 1 almost every iteration. The quantities we are looking at within it (comparing data and smoothness) are simply too large. We need to normalize our distribution.

4 Preliminary Structure

In the following sections, we examine the impact of normalizations on our data terms, and explore the motivations behind the various constructions. More rigorous data is provided concerning the final form, while
4.1 Motivation

We first take a moment to examine the sensitivities $\lambda_i$, and impose the following condition: $\lambda_1 > \lambda_2$ and $\lambda_1 > \lambda_3$. Not doing so would mean the algorithm could give us some false positives. This leads us to notice that a key aspect of the MHMCMC method is information. Due to the ill-posed nature of the problem, we need to keep every piece of information that can be gleaned. We will keep this idea in mind throughout the later sections.

As for the normalizations proper, the naive approach to our problem would be to divide each data term by a constant value. In this formulation, our normalization terms would have the form

$$z_i = \frac{1}{c_i} \quad (4.1.1)$$

where $c_i$ can be determined by looking at representative values of our data terms.

This approach has one advantage, which is that it retains information very well. The relationships between quantities is affected by a constant factor, and its evolution is therefore preserved across iterations. Unfortunately, this method is very unstable, and is not particularly viable. One can think of the opposite method to
this one being dividing each data term by itself. Clearly, this would erase all information contained within our results, but it would successfully normalize it, given a broad enough definition of success.

Concretely, we seek to find a normalization that delivers information about the evolution of our data terms, but bounds the results so that we may control their magnitudes and work with their relative relationships.

5 Normalized with Inertia

We introduce the concept of inertia in this framework. Inertia can be thought of as the weight (call it \(w\)) being applied to either previous method. Though we do not want to divide by only a constant, there is merit to letting some information trickle through to us. If we do not bound the quantities we are examining, then we will obtain very small or very large values for \(\alpha\), effectively 0 or 1, which is undoing the work of the MHMCMC. We attempt to bound our likelihood externally. We define \(\alpha_h\) such that

\[
\alpha(K' | K_n) \equiv z_0\alpha_h = z_0 e^{-\left(\sum_i \lambda_i z_i D_i\right)}
\]

5.1 Global Normalizations

Even a cursory analysis of our early attempts at solving this heat conductivity problem have revealed a desperate need to correctly normalize our data in order to get meaningful likelihoods. Some issues of note have been the idea that the inertia of the process, the value of previous guesses, contains information which is important to the successful convergence of our algorithm. Another is the fact that the variance of data terms means that we require a strong normalization term, at the expense, perhaps, of information, if we are to obtain meaningful results.

Addressing the second point, we decide to deviate slightly from one aspect of our method, and use a global result. Computationally, we will only be tracking one variable, and this poses no problem. But note that using a global result in computing \(\alpha\) implies that our process is no longer a Markov process, as the probability of reaching the next step is dependent on the past and not just the present.

5.2 Formulation of \(Z(1)\)

First, let \(\alpha_{h,m} = \max_j \{\alpha_{h,j}\}, \forall j\) and \(D_{i,m} = \max_j \{D_{i,j}\}, \forall j\). We denote \(Z(1)\) the normalization

\[
\begin{align*}
z_{i,j}^{(1)} &= w_0 \frac{1}{\alpha_{h,j}} + (1 - w_0) \frac{1}{\alpha_{h,m}} \\
(z_{i,j}^{(1)}) &= w \frac{1}{|D_{i,j}|} + (1 - w) \frac{1}{|D_{i,m}|}
\end{align*}
\]

While this effectively bounds our acceptance probability between \([0, 1]\), it does so at the expense of the Markov property of our algorithm. Removing this property exhibits some instability in the evolution of the algorithm. Namely, they appear to converge to false positives, an effect which must be explored more fully.

5.3 Restricted Random Interval

Examining the values of \(\alpha\) that we now produce reveals that we have greatly tightened the spread. Almost all of our values are contained in a narrow band (which changes depending on parameters), say between 0.6 and 0.75. Again, this means we are losing information, as the difference in the values of \(\alpha\) are lost by comparing them over the entire \([0, 1]\) interval.

We change the 2nd step in the MHMCMC algorithm, which was \(\alpha > u \sim U[0, 1] \Rightarrow K_{n+1} = K'\). We now restrict the interval over which we draw \(u\), taking its lower and upper bounds at the \(j\)th iteration to be \([u_{\text{min}}, u_{\text{max}}]\), where for some small constant \(\zeta\),

\[
u_{\text{min}} = \min_{i < j} \alpha_i - \zeta \quad \wedge \quad u_{\text{max}} = \max_{i < j} \alpha_i + \zeta
\]
While perhaps more restrictive, this formulation also greatly increases the speed at which the algorithm begins to converge by effectively selecting those guesses which are the most promising, relative to the past performance of the algorithm. This method implies that we will not, with probability 1, decide the outcome of a guess, they simply become (as per $\zeta$) extremely unlikely to be accepted or rejected.

6 Locally Focused Normalization

We now attempt to modify $Z^{(1)}$ in order to retain the original Markov property of the algorithm. The property was violated in the second term, which unfortunately also guarantees we bound our results.

6.1 Formulation of $Z^{(2)}$

Denote a new normalization scheme $Z^{(2)}$, given by

$$z_{0,j}^{(2)} = w_0 \frac{1}{\alpha_{h,j}} + (1 - w_0) \frac{1}{\alpha_{h,j-1}} \quad (6.1.1a)$$

$$z_{i,j}^{(2)} = w \frac{1}{|D_{i,j}|} + (1 - w) \frac{1}{|D_{i,j-1}|} \quad (6.1.1b)$$

While we have recovered the Markov property, we must now contend with unbounded values for $\alpha$. We note now that preliminary attempts to use $z_{i,j}^{(2)}$ with $z_{0,j}^{(1)}$ did not yield promising results.

While this formulation provides good results, it does require us to find an empirical bound for $\alpha$, as it is no longer bounded by $z_0$. For the results presented below, we imposed $\alpha \in [0, 1.5]$, setting

$$\alpha (K' \mid K_n) = \min \left\{ 1.5, z_0 e^{-\left( \sum_{i=1}^{3} \lambda_i z_i D_i \right)} \right\} \quad (6.1.2)$$

6.2 Results

The parameters we have to determine are $\lambda_1, \lambda_2, \lambda_3, w, w_0$ and the cutoff for $\alpha$ as in (6.1.2). We have concluded we must set $\lambda_1 > \lambda_i, \forall i > 1$ and we have by definition $w, w_0 \in [0, 1]$. The exact values of the sensitivities and inertia factors are at the moment heuristically chosen to be

$$\lambda_1 = 0.5, \quad \lambda_2 = 0.15, \quad \lambda_3 = 0.45$$

$$w_0 = 0.1, \quad w = 0.75, \quad \alpha_{cutoff} = 1.5$$

For the tilted plane, we obtain Figure 4. As mentioned in Section 5.2, we have some instability in the form of incorrect convergence for $Z^{(1)}$, which is apparent in Figure 5 as well. On the other hand, $Z^{(2)}$ converges well and produces a smooth reconstruction. We can also note that it achieves slightly better results than

![Figure 4: $Z^{(1)}$ and $Z^{(2)}$ reconstructions of a tilted plane with priors, 2 million iterations.](image)
the no-normalizations case in only 2 million iterations. The instability in $Z^{(1)}$ is again apparent, and leads us to conclude that the loss of the Markov property in the algorithm may be detrimental to its performance. However, the reconstruction of the Gaussian well has substantially improved when using $Z^{(2)}$. It achieves a smoother reconstruction as without normalizations (see Figure 2), and in 4M iterations instead of 10M.

Going back to our error metric $\Gamma$, we see the improvement manifest itself rather clearly, with acceptances being on the order of $\sim 55\%$ instead $\sim 95\%$ as they were before.

7 Conclusion

The need for normalizing factors arose from the variance in the magnitudes of data terms $D_i$ from one iteration to the next. In formulating those factors, we focused on conserving the information contained in $D_i$ while bounding our quantities, and we confirmed the importance of retaining the Markov property in this context. However, by using the $Z^{(2)}$ formulation, we were able to obtain faster and better reconstructions of the conductivity for both the tilted plane and the Gaussian well.

Despite the encouraging results, several avenues need to be explored more fully. The long-run behavior of $Z^{(2)}$ seems to exhibit some stagnation, seemingly having converged as best as it can. In addition, very preliminary results have been obtained for a scheme that lies between $Z^{(1)}$ and $Z^{(2)}$, which updates the $(1 - \omega)$ terms only when a guess is accepted, has shown competitive performance relative to $Z^{(2)}$.

As the algorithm currently stands $\alpha_{\text{cutoff}}$, the sensitivities $\lambda_i$, and the inertia factors $w, w_0$ must be determined heuristically. It is possible we may be able to dynamically adjust the values as the algorithm runs,
through a constrained optimization of the acceptance rate, but that remains to be studied.

Finally, we would like to implement Gaussian-Mixture based MCMC algorithms, that treat the proposal density as an unknown to be approximated, and add to this framework our normalization schemes to observe the interaction of the two methods.

References

[1] Fox, C., Nicholls, G., Tan, S. Inverse Problems, Physics 707, The University of Auckland, ch. 7-9.

[2] Hastings, W. “Monte Carlo Sampling Methods Using Markov Chains and Their Applications.” Biometrika, Vol 57, No. 1, (1970), pp. 97-109.

[3] Luengo, D., Martino, L. “Fully Adaptive Gaussian Mixture Metropolis-Hastings Algorithm” Working Paper (2013), avail. at: http://arxiv.org/abs/1212.0122

[4] Metropolis, N., Rosenbluth, A., et. al. “Equations of State Calculations by Fast Computing Machines” Journal of Chemical Physics, Vol 21 (1953), pp. 1087-1092.

[5] Rosenthal, J. “Optimal Proposal Distributions and Adaptive MCMC.” Chapter for MCMC Handbook (2010), avail. at http://www.probability.ca/jeff/ftpdir/galinart.pdf

[6] Sauer, T., Numerical Analysis, Pearson Addison-Wesley, 2006.

[7] Zambelli, A., “A Multiple Prior Monte Carlo Method for the Backward Heat Diffusion Problem” Proc. 11th Int. Conf. Comp. Math. Meth. Sci. Eng. (CMMSE’2011), Vol 3 (2011), pp. 1192-1200.