Convergence of Adaptive Markov Chain Monte Carlo Algorithms

Christian Rudnick

August 31, 2009; last revised September 7, 2009

STA496 Reading in Statistics:
Investigations of Adaptive Markov Chain Monte Carlo Algorithms
Summer 2009
University of Toronto
Prof. Jeffrey S. Rosenthal

Christian Rudnick
96 Gerrard Street East
Toronto, ON, M5B1G7
e-mail: christian.rudnick@utoronto.ca

Contents

1 Markov Chain Monte Carlo Algorithms

In the past century, several techniques to obtain random samples from arbitrary distributions have been developed. Standard techniques such as the Inverse CDF method, importance sampling, rejection sampling, and slice sampling fail in many instances because they cannot be (efficiently) implemented. Markov chain Monte Carlo (MCMC) techniques, in turn, can be used to obtain approximate samples from a very large class of probability distributions. This includes distributions that have a density with respect to the Lebesgue measure which is known up to the normalizing constant. To introduce MCMC algorithms, the concept of a Markov chain, which is a discrete-time stochastic process where, intuitively, the next step depends only on the current state, is needed.

Definition 1 (Markov chain). *Let $\mathcal{X}$ be a set with σ-field $\mathcal{F}$. A stochastic process $(X_n)_{n=0}^{\infty}$ is a Markov chain on $\mathcal{X}$ if there exists a transition kernel $P(\cdot,\cdot)$ which is a collection satisfying that for each $x \in \mathcal{X}$, $P(x,\cdot)$ is a probability measure and for each $A \in \mathcal{F}$, $P(\cdot, A)$ is a non-negative measurable function such that for every initial distribution $\mu(\cdot)$, every $n \in \mathbf{N}$, and every measurable $X_i \subseteq A_i$ for $i = 1, 2, \ldots, n$,*

$$\begin{aligned}&\mathbf{P}_\mu\left(X_0 \in A_0, X_1 \in A_1, \ldots, X_n \in A_n\right)\\ =\ &\int_{y_0 \in A_0} \int_{y_1 \in A_1} \cdots \int_{y_{n-1} \in A_{n-1}} \mu(\mathrm{d}y_0) P(y_0, \mathrm{d}y_1) \cdots P(y_{n-1}, A_n).\end{aligned}$$

If $\mu = \delta_x$ is the distribution that assigns probability one to the point x, it is written as $\mathbf{P}_x$ instead of $\mathbf{P}_\mu$. Proceed in the same way for the operator of the expected value. The n-step transition kernels are obtained by convoluting the transition kernel with itself n times.

Definition 2 (n-step transition kernel). *Consider a Markov chain on $\mathcal{X}$ with transition kernel $P(\cdot,\cdot)$. The n-step transition kernels are given by*

$$P^0(x, A) = \delta_x(A),$$

and for $n \in \mathbf{N}$, inductively by

$$P^n(x, A) = \int_{x \in \mathcal{X}} P(x, \mathrm{d}y) P^{n-1}(y, A),$$

where $x \in \mathcal{X}$ and $A \in \mathcal{F}$.

If there is a distribution $\pi(\cdot)$ on the state space $\mathcal{X}$ which is invariant to the application of the transition kernel, the Markov chain is said to be stationary and $\pi(\cdot)$ is called the stationary distribution.

Definition 3 (Stationary Distribution)**.** *Consider a Markov chain with transition kernel $P(\cdot, \cdot)$. Then the Markov chain is stationary with stationary distribution $\pi(\cdot)$ if*

$$\pi(\mathrm{d}x) = \int_{y \in \mathcal{X}} P(y, \mathrm{d}x)\pi(\mathrm{d}y).$$

If a Markov chain has a stationary distribution, the n-step transition probabilities converge to the stationary distribution under some regularity conditions. Such a Markov chain is said to be ergodic. To define ergodicity, the total variation distance needs to be introduced:

Definition 4 (Total variation distance)**.** *The total variation distance between two probability measures $\nu_1(\cdot)$ and $\nu_2(\cdot)$ is*

$$\|\nu_1(\cdot) - \nu_2(\cdot)\| := \sup_{A \in \mathcal{F}} |\nu_1(A) - \nu_2(A)|.$$

Definition 5 (Ergodicity)**.** *A Markov chain with transition kernel $P(\cdot, \cdot)$ is ergodic towards its stationary distribution $\pi(\cdot)$ if*

$$\lim_{n \to \infty} \sup_{x \in \mathcal{X}} \|P^n(x, \cdot) - \pi(\cdot)\| = 0.$$

It is now straightforward to give an informal definition of MCMC algorithms: They are techniques that sample from a distribution $\pi(\cdot)$ via a Markov chain which has this distribution as a stationary distribution. For large n, the transition probabilities of the Markov chain and the stationary distribution will be approximately equal.

The regularity conditions needed for convergence are irreducibility and aperiodicity. A Markov chain is irreducible if each set of positive measure can be reached in a finite number of steps from any point in the state space.

Definition 6 (ϕ-Irreducibility)**.** *A Markov chain which has transition kernel $P(\cdot, \cdot)$ is ϕ-irreducible if there exists a non-zero, σ-finite measure $\phi(\cdot)$ on $\mathcal{X}$ such that, for all $A \subseteq \mathcal{X}$ with $\phi(A) > 0$ and all $x \in \mathcal{X}$, there exists $k \in \mathbf{N}$ such that $P^k(x, A) > 0$.*

A Markov chain is aperiodic, if there does not exist a partition of the state space such that the Markov chain deterministically cycles through the partition.

Definition 7 (Aperiodicity)**.** *A Markov chain with transition kernel $P(\cdot,\cdot)$ is aperiodic if there does not exist a partition of the state space $\mathcal{X}$, that is disjoint sets $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k$ with $k \geq 2$, $\bigcup_{i=1}^k \mathcal{X}_i = \mathcal{X}$, and $\pi(\mathcal{X}_i) > 0$ for all $i = 1, 2, \dots, k$ and such that for $i = 1, 2, \dots, k-1$,*

$$\forall x \in \mathcal{X}_i : P(x, \mathcal{X}_{i+1}) = 1$$

and

$$\forall x \in \mathcal{X}_k : P(x, \mathcal{X}_1) = 1.$$

Otherwise, the Markov chain is said to be periodic.

The convergence result is summarized in the following theorem:

Theorem 1 (Markov chain convergence theorem)**.** *An aperiodic and ϕ-irreducible Markov chain with stationary distribution $\pi(\cdot)$ is ergodic towards its stationary distribution.*

Proof. An upcoming subsection will be devoted to the proof of this theorem. □

1.1 Algorithms

Several MCMC algorithms have been proposed. The most prominent is the Metropolis-Hastings algorithm [8] which proceeds as follows: One starts with some initial value $X_0 = x_0$ and then iterates: Given X_n, one draws $Y_n \sim q(X_n, \cdot)$, where q is an arbitrary probability distribution that may depend only on X_n. q is the so-called proposal. Define the acceptance function by

$$\alpha(x,y) := \begin{cases} \min\left\{1, \frac{q(y,x)\pi(y)}{q(x,y)\pi(x)}\right\} & q(x,y)\pi(x) > 0 \\ 1 & q(x,y)\pi(x) = 0 \end{cases}.$$

Then, draw independently $U \sim \text{Unif}(0,1)$ and sets $X_{n+1} = Y_n$ if $U < \alpha(X_n, Y_n)$ ("accepting the proposal") and $X_{n+1} = X_n$ otherwise ("rejecting the proposal").

A predecessor of the Metropolis-Hastings algorithm is the Metropolis algorithm [11] which is only suitable for symmetric proposals, that is, proposals that satisfy

$$\forall x, y \in \mathcal{X} : q(x, y) = q(y, x).$$

In that case, the quantity $\alpha(\cdot, \cdot)$ becomes

$$\alpha(x, y) = \frac{\pi(y)}{\pi(x)}.$$

If $\mathcal{X} = \mathbf{R}^d$, there is a further important special case, the Metropolis algorithm with normal proposals. Its family of proposals is given by

$$q_d(\mathbf{X}_n, \mathbf{Y}_n) := \frac{1}{(2\pi)^{d/2}(\det(\Sigma))^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{Y}_n - \mathbf{X}_n)'\Sigma^{-1}(\mathbf{Y}_n - \mathbf{X}_n)\right\},$$

for some positive definite covariance matrix Σ. Both special cases are important since they lead to more extensive theory.

1.2 Proof of the Markov Chain Convergence Theorem

This section provides a proof of the Markov chain convergence theorem that relies on the so-called coupling construction (a self-contained proof by other means can be found in [1]). For this one needs the following definition.

Definition 8 (Minorisation Condition). *Consider a Markov chain on a state space $\mathcal{X}$ with transition kernel $P(\cdot, \cdot)$. It satisfies a minorisation condition if there exists a probability measure $\nu(\cdot)$ on $\mathcal{X}$, a set $C \subseteq \mathcal{X}$, an integer $n_0 \in \mathbf{N}$, and a constant $\varepsilon > 0$ such that*

$$\forall x \in C : P^{n_0}(x, \cdot) \geq \varepsilon\nu(\cdot).$$

In this case, $\nu(\cdot)$ is called minorisation measure and the set C is said to be a small set.

The coupling construction uses two independent Markov chains $(X_n)_{n=0}^{\infty}$ and $(X'_n)_{n=0}^{\infty}$ with identical transition kernels, but different starting distributions: $X_0 = x$, whereas $X'_0 \sim \pi(\cdot)$. Even though the Markov chains are (marginally) updated using the same transition kernel $P(\cdot, \cdot)$, their joint construction is such that they "couple", i. e. become equal, with maximal probability. To achieve this, consider a small set C and define the following loop:

(i) If $(X_n, X'_n) \in C \times C$, with probability ε, choose $X_{n+n_0} \sim \nu(\cdot)$ and set $X'_{n+n_0} = X_{n+n_0}$ or otherwise, choose (conditionally independent)

$$\begin{aligned} X_{n+n_0} &\sim \tfrac{1}{1-\varepsilon}\left(\mathbf{P}^{n_0}(X_n, \cdot) - \varepsilon\nu(\cdot)\right), \\ X'_{n+n_0} &\sim \tfrac{1}{1-\varepsilon}\left(\mathbf{P}^{n_0}(X'_n, \cdot) - \varepsilon\nu(\cdot)\right). \end{aligned}$$

Construct the intermediate steps

$$X_{n+1}, X_{n+2}, \ldots, X_{n+n_0-1}$$

and

$$X'_{n+1}, X'_{n+2}, \ldots, X'_{n+n_0-1}$$

from the appropriate conditional distributions given X_n and X_{n+n_0} (or X'_n and X'_{n+n_0}, respectively). Increase n by n_0. In words, if the joint Markov chain hits the set $C \times C$, there is a small chance that the coupling occurs.

(ii) If $X_n = X'_n$, then update $X_n \sim P(X_n, \cdot)$, set $X'_n = X_n$ and increase n by one; this is the case where the Markov chains have already coupled, so nothing has to be changed.

(iii) If $X_n \neq X'_n$ and $(X_n, X'_n) \notin C \times C$, sample

$$\begin{aligned} X_{n+1} = X_n &\sim P(X_n, \cdot), \\ X'_{n+1} = X'_n &\sim P(X'_n, \cdot), \end{aligned}$$

and increase n by one.

The usefulness of this construction becomes evident by the following lemma.

Lemma 1 (Coupling inequality)**.** *Let X_n and X'_n be the Markov chains described above. Then,*

$$\|P^n(x, \cdot) - \pi(\cdot)\| \leq \mathbf{P}[X_n \neq X'_n].$$

Proof. From the coupling construction, in particular the fact that X_n and X'_n are marginally updated from the transition kernel P, it follows that

$P^n(x, A) = \mathbf{P}[X_n \in A]$ and $\pi(A) = \mathbf{P}[X'_n \in A]$ for all $n \in \mathbf{N}$. Therefore

$$\begin{aligned}
\|P^n(x,\cdot) - \pi(\cdot)\| &= \sup_{A\in\mathcal{F}} |P^n(x,A) - \pi(A)| \\
&= \sup_{A\in\mathcal{F}} |\mathbf{P}[X_n \in A] - \mathbf{P}[X'_n \in A]| \\
&= \sup_{A\in\mathcal{F}} |\mathbf{P}[X_n \in A, X_n = X'_n] + \mathbf{P}[X_n \in A, X_n \neq X'_n] \\
&\quad -\mathbf{P}[X'_n \in A, X_n = X'_n] - \mathbf{P}[X'_n \in A, X_n \neq X'_n]| \\
&= \sup_{A\in\mathcal{F}} |\mathbf{P}[X_n \in A, X_n \neq X'_n] - \mathbf{P}[X'_n \in A, X_n \neq X'_n]| \\
&\leq \mathbf{P}[X_n \neq X'_n].
\end{aligned}$$

□

It should be mentioned that the coupling inequality is valid for the distributions of two arbitrary random variables (the proof remains the same).

To prove the Markov chain convergence theorem, three further auxiliary results are needed and two of them will only be stated without proof. The first one guarantees, under fairly general conditions, the existence of a minorisation criterion.

Lemma 2. *Consider a ϕ-irreducible Markov chain on a state space $\mathcal{X}$ that is equipped with a countably generated σ-field. Then the Markov chain satisfies a minorisation criterion with small set C and minorisation measure $\nu(\cdot)$. Moreover, the small set $C \subseteq \mathcal{X}$ satisfies $\nu(C) > 0$.*

Proof. See [9] or [12], theorem 5.2.1. □

The second one is a rather technical result which makes use of the aperiodicity assumption in the Markov chain convergence theorem.

Lemma 3. *Consider an aperiodic Markov chain with transition kernel $P(\cdot,\cdot)$ and stationary distribution $\pi(\cdot)$. Assume that the Markov chain satisfies a minorisation criterion with minorisation measure $\nu(\cdot)$ and a small set C such that*

$$\forall x \in \mathcal{X} : \exists n_0 \in \mathbf{N} : P^{n_0}(x, C) > 0.$$

Let

$$T = \{k \geq 1 : \exists \delta_k > 0 \text{ s. t. } (\nu P)(\cdot) \geq \delta_k \nu(\cdot)\}$$

be non-empty. Then there is $n^* \in \mathbf{N}$ such that

$$\{n^*, n^* + 1, n^* + 2, \ldots\} \subseteq T.$$

Proof. See [15], lemma 35 in the appendix. □

The third one, finally, is a fact about return probabilities.

Lemma 4. *Consider a Markov chain with transition kernel $P(\cdot,\cdot)$ and stationary distribution $\pi(\cdot)$. Let $A \subseteq \mathcal{X}$ and $\tau_A := \inf\{k \in \mathbf{N} : X_k \in A\}$. If for all $x \in \mathcal{X}$*

$$\mathbf{P}_x[\tau_A < \infty] > 0, \tag{1}$$

then for π-a. e. $x \in \mathcal{X}$

$$\mathbf{P}_x[\tau_A < \infty] = 1,$$

Proof. Suppose the conclusion is false. Then the set

$$B := \{x \in \mathcal{X} : \mathbf{P}_x[\tau_A = \infty] > 0\} \tag{2}$$

has a positive measure, i. e. $\pi(B) > 0$. Using countable additivity one can extract a set $B_1 \subseteq B$ with $\pi(B_1) > 0$ and a constant $\delta_1 > 0$ such that

$$\forall x \in B_1 : P(x, A) \leq 1 - \delta_1.$$

Similarly, using equation (1), one concludes that there is $B_2 \subseteq B_1$ with $\pi(B_2) > 0$, a constant $\delta_2 > 0$, and an integer $\ell_2 \in \mathbf{N}$ such that

$$\forall x \in B_2 : \delta_2 \leq P^{\ell_2}(x, A) \leq 1 - \delta_1.$$

Define $\eta = \#\{k \in \mathbf{N} : X_{k\ell_2} \in B_2\}$. Then, for any $x \in \mathcal{X}$,

$$\mathbf{P}[\eta = k, \tau_A = \infty] \leq \left(P^{\ell_2}(x, B)\right)^k \leq (1 - \delta_2)^k.$$

This easily yields

$$\begin{aligned}
\mathbf{P}[\eta < \infty, \tau_A = \infty] &= 1 - \mathbf{P}[\eta = \infty, \tau_A = \infty] - \mathbf{P}[\tau_A < \infty] \\
&\geq 1 - \lim_{k\to\infty} \mathbf{P}[\eta = k, \tau_A = \infty] + (1 - \delta_1) \\
&\geq 1 - \lim_{k\to\infty} (1 - \delta_2)^k + (1 - \delta_1) \\
&= \delta_1.
\end{aligned}$$

Thus, there is a set $B_3 \subseteq B_2$ with $\pi(B_3) > 0$, an integer $\ell_3 \in \mathbf{N}$, and a constant $\delta_3 > 0$ such that for all $x \in B_3$

$$\mathbf{P}_x\left[\tau_A = \infty, \sup\{k \in \mathbf{N} : X_{k\ell_3} \in B_2\}\right] \geq \delta_3.$$

However, B_3 is contained in B_2, and the conclusion is, of course, valid for the smaller set, i. e.

$$\mathbf{P}_x\left[\tau_A = \infty, \sup\{k \in \mathbf{N} : X_{k\ell_3} \in B_3\}\right] \geq \delta_3.$$

Now, let $L := \ell_2\ell_3$ and define

$$S := \begin{cases} \sup\{k \in \mathbf{N} : X_{kL} \in B\} & \exists k \in \mathbf{N} : X_{kL} \in B \\ -\infty & \text{otherwise} \end{cases}.$$

Then,

$$\begin{aligned}
\pi(A^c) &= \int_{x\in\mathcal{X}} \pi(\mathrm{d}x) P^{jL}(x, A^c) \\
&= \int_{x\in\mathcal{X}} \pi(\mathrm{d}x) P[X_{jL} \in A^c] \\
&= \int_{x\in\mathcal{X}} \sum_{i=1}^{\infty} \pi(\mathrm{d}x) P[X_{jL} \in A^c, S = i] \\
&\geq \int_{x\in\mathcal{X}} \sum_{i=1}^{j} \pi(\mathrm{d}x) P[X_{jL} \in A^c, S = i] \\
&= \sum_{i=1}^{j} \int_{x\in\mathcal{X}} \pi(\mathrm{d}x) P[X_{jL} \in A^c, S = i] \\
&= \sum_{i=1}^{j} \int_{x\in\mathcal{X}} \pi(\mathrm{d}x) \int_{y\in B_3} P^{rL}(x, \mathrm{d}y) \mathbf{P}_y[X_{(j-r)L} \in A^c, S = -\infty] \\
&= \sum_{i=1}^{j} \int_{y\in B_3} \pi(\mathrm{d}y) P^{rL}(x, \mathrm{d}y) \mathbf{P}_y[X_{(j-r)L} \in A^c, S = -\infty] \\
&\geq \sum_{i=1}^{j} \int_{y\in B_3} \pi(\mathrm{d}y)\delta_3 \\
&= \sum_{i=1}^{j} \pi(B_3)\delta_3 \\
&= j\pi(B_3)\delta_3 \\
&> 1
\end{aligned}$$

if $j > 1/(\pi(B_3)\delta_3)$. This is a contradiction. □

With this machinery, the proof of the Markov chain convergence theorem is straightforward. First, lemma 2 yields existence the of a minorisation condition with a small set C. From the coupling construction one has the chain $(X_n', X_n)_{n=0}^{\infty}$. Consider the set

$$G := \left\{(x, x') \in \mathcal{X} \times \mathcal{X} : \mathbf{P}_{(x,x')}[\exists n_0 \in \mathbf{N} : X_{n_0} = X_{n_0}']\right\}.$$

If $(X_0, X_0') \in G$, then $X_k = X_k'$ for some $k \in \mathbf{N}$ by the definition of the coupling construction. Then, by the setup of the coupling construction, $X_i = X_i'$ for all $i \geq k$, hence, $\lim_{n\to\infty} \mathbf{P}[X_n = X_n'] = 1$. The coupling inequality will yield the desired $\lim_{n\to\infty} \|P^n(x, \cdot) - \pi(\cdot)\| = 0$.

The proof is completed by showing that for π-a. e. $x \in \mathcal{X}$, it is true that $\mathbf{P}[(X_0, X_0') \in G] = 1$. Consider the slices $G_x := \{y \in \mathcal{X} : (x, y) \in G\}$ for $x \in \mathcal{X}$ and $\overline{G} := \{x \in \mathcal{X} : \pi(G_x) = 1\}$.

By lemma 2, $\nu(C) > 0$, so by lemma 3, the probability is greater than zero that the joint chain will eventually hit $C \times C$ from any starting point $(x, y) \in \mathcal{X} \times \mathcal{X}$. By lemma 4, the joint chain will return to $C \times C$ almost surely from $\pi \times \pi$-a. e. $(x, y) \in C \times C$. As soon as the chain reaches $C \times C$, conditional on not coupling, application of lemma 4 yields that the chain will hit $C \times C$ almost surely once again; therefore, the joint chain will hit $C \times C$ until coupling occurs, that is, $X_n = X_n'$. Hence $(\pi \times \pi)(G) = 1$. Now assume $\pi(\overline{G}) < 1$. Then

$$0 = (\pi \times \pi)(G^c) = \int_{x \in \mathcal{X}} \pi(\mathrm{d}x)\pi(G_x^c) = \int_{\overline{G}^c} \pi(\mathrm{d}x)\left(1 - \pi(G_x)\right) > 0.$$

A contradiction, so $\pi(\overline{G}) = 1$ which proves the theorem. Even though this proof is relatively short, there is still doubt whether:

Open Problem 1. *Is it possible to obtain a shorter proof of the Markov chain convergence theorem following the idea of the proof above?*

1.3 Ergodicity

In the upcoming section, the ergodicity of some explicit MCMC algorithms will be proven. First, it will be shown that $\pi(\cdot)$ is actually the stationary distribution for the Markov chain described above. This is easily done by showing that the Markov chain satisfies the reversibility condition.

Definition 9 (Reversibility)**.** *A Markov chain on $\mathcal{X}$ with transition kernel $P(\cdot,\cdot)$ is reversible with respect to a distribution $\pi(\cdot)$ if*

$$\forall x, y \in \mathcal{X} : \pi(\mathrm{d}x)P(x, \mathrm{d}y) = \pi(\mathrm{d}y)P(y, \mathrm{d}x).$$

Lemma 5. *Consider a Markov chain with transition kernel $P(\cdot,\cdot)$ which is reversible with respect to $\pi(\cdot)$. Then $\pi(\cdot)$ is a stationary distribution for the Markov chain.*

Proof. Using the reversibility condition in the first step, obtain

$$\begin{aligned}\int_{y\in\mathcal{X}} P(y, \mathrm{d}x)\pi(\mathrm{d}y) &= \int_{y\in\mathcal{X}} P(x, \mathrm{d}y)\pi(\mathrm{d}x) \\ &= \pi(\mathrm{d}x)\int_{y\in\mathcal{X}} P(x, \mathrm{d}y) \\ &= \pi(\mathrm{d}x).\end{aligned}$$

□

We can now easily deduce:

Proposition 1. *Assume a Metropolis-Hastings algorithm with the following properties: $\pi(\cdot)$ and the proposal distribution both have densities with respect to the Lebesgue measure,*

$$\pi(\mathrm{d}x) = \pi(x)\mathrm{d}x$$
$$q(x, \mathrm{d}y) = q(x, y)\mathrm{d}y.$$

Then $\pi(\cdot)$ is a stationary distribution of the Markov chain produced by the Metropolis-Hastings algorithm.

Proof. It suffices to show that the Markov chain is reversible and apply the previous lemma. The equation is trivial if $x = y$. Otherwise,

$$\begin{aligned}\pi(\mathrm{d}x)P(x, \mathrm{d}y) &= \pi(x)\mathrm{d}x\alpha(x, y)q(x, y)\mathrm{d}y \\ &= \pi(x)\min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\} q(x, y)\mathrm{d}x\mathrm{d}y \\ &= \min\left\{\pi(x)q(x, y), \pi(y)q(y, x)\right\}\mathrm{d}x\mathrm{d}y \\ &= \pi(y)\min\left\{1, \frac{\pi(x)q(x, y)}{\pi(y)q(y, x)}\right\} q(y, x)\mathrm{d}y\mathrm{d}x \\ &= \pi(y)\mathrm{d}y\alpha(y, x)q(y, x)\mathrm{d}x \\ &= \pi(\mathrm{d}y)P(y, \mathrm{d}x).\end{aligned}$$

□

According to the Markov chain convergence theorem, there are still two conditions to check: aperiodicity and irreducibility. As for the stationarity of $\pi(\cdot)$, there exists a condition that is easy to check and implies aperiodicity.

Lemma 6. *Consider a Markov chain with transition kernel $P(\cdot,\cdot)$. Suppose that for some $x \in \mathcal{X}$, there exists a sequence of sets $A_1 \supseteq A_2 \supseteq \ldots$ with $P(x, A_i) > 0$ for all $i \in \mathbf{N}$ and the property that for any $B \subseteq \mathcal{X}$ with $x \in B$ and $\pi(B) > 0$, there exists $n_0 \in \mathbf{N}$ such that*

$$\forall i > n_0 : A_i \subseteq B.$$

Then the Markov chain is aperiodic.

Proof. Suppose the Markov chain is periodic. Then there exist $k \in \mathbf{N}$ and disjoint sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ such that $\bigcup_{i=1}^k \mathcal{X}_i = \mathcal{X}$ with $P(x, \mathcal{X}_i) = 1$ for $x \in \mathcal{X}_i$ when $i = 2, \ldots, k$ and $P(x, \mathcal{X}_1) = 1$ for $x \in \mathcal{X}_k$. Without loss of generality, assume that $x \in \mathcal{X}_1$. By assumption, there exist n_0 such that $P(x, A_{n_0}) > 0$ and $A_{n_0} \subseteq \mathcal{X}_1$. Since $P(x, \mathcal{X}_2) = 1$ as well, a contradiction follows:

$$1 = P(x, \mathcal{X}) = \sum_{i=1}^{k} P(x, \mathcal{X}_i) \geq \sum_{i=1}^{2} P(x, \mathcal{X}_i) \geq P(x, A_{n_0}) + P(x, \mathcal{X}_2) > 1.$$

□

The following property is useful when investigating Markov chains.

Definition 10 (Regularity)**.** *A distribution $\nu(\cdot)$ is regular if it has a density with respect to the Lebesgue measure and it is bounded away from zero and infinity on compact sets, that is, for every compact set $D \subset \mathbf{R}^d$ there exist constant C_D and c_D such that*

$$\forall x \in D : c_D \leq \nu(x) \leq C_D.$$

This leads to the following corollary:

Corollary 1. *Consider a Metropolis algorithm on $\mathbf{R}^d$ with stationary distribution $\pi(\cdot)$. If the proposal distribution is regular, then the Metropolis-Hastings algorithm is aperiodic.*

Proof. First, calculate for arbitrary $x \in \mathbf{R}^d$ and $B \subseteq \mathbf{R}^d$.

$$\begin{aligned}
P(\mathbf{x}, B) &= \mathbf{P}[\mathbf{X}_n \in B | \mathbf{X}_{n-1} = \mathbf{x}] \\
&= \int_{\mathbf{y} \in \mathbf{R}^d} \mathbf{P}[\mathbf{X}_n \in B | \mathbf{Y}_n = \mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] \mathbf{P}[\mathbf{Y}_n \in \mathrm{d}\mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] \\
&= \mathbf{1}_B(\mathbf{x})(\mathbf{x}) \int_{\mathbf{y} \in B^c} \mathbf{P}[\mathbf{X}_n \in B | \mathbf{Y}_n = \mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_{B^c} \int_{\mathbf{y} \in B^c} \mathbf{P}[\mathbf{X}_n \in B | \mathbf{Y}_n = \mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_B(\mathbf{x}) \int_{\mathbf{y} \in B} \mathbf{P}[\mathbf{X}_n \in B | \mathbf{Y}_n = \mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_{B^c}(\mathbf{x}) \int_{\mathbf{y} \in B} \mathbf{P}[\mathbf{X}_n \in B | \mathbf{Y}_n = \mathbf{y}, \mathbf{X}_{n-1} = \mathbf{x}] q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \mathbf{1}_{B^c}(\mathbf{x}) \int_{\mathbf{y} \in B^c} (1 - \alpha(\mathbf{x}, \mathbf{y})) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_B(\mathbf{x}) \int_{\mathbf{y} \in B} q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} + \mathbf{1}_{B^c}(\mathbf{x}) \int_{\mathbf{y} \in B} \alpha(\mathbf{x}, \mathbf{y}) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \mathbf{1}_{B^c}(\mathbf{x}) \int_{\mathbf{y} \in B^c} (1 - \alpha(\mathbf{x}, \mathbf{y})) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_B(\mathbf{x}) \int_{\mathbf{y} \in B} (\alpha(\mathbf{x}, \mathbf{y}) + 1 - \alpha(\mathbf{x}, \mathbf{y})) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \mathbf{1}_{B^c}(\mathbf{x}) \int_{\mathbf{y} \in B} \alpha(\mathbf{x}, \mathbf{y}) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \mathbf{1}_B(\mathbf{x}) \int_{\mathbf{y} \in \mathbf{R}^d} (1 - \alpha(\mathbf{x}, \mathbf{y})) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} + \int_{\mathbf{y} \in B} \alpha(\mathbf{x}, \mathbf{y}) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y}.
\end{aligned}$$

Now, let $B(\mathbf{z}, \varepsilon)$ be the ball of radius $\varepsilon > 0$ with center $\mathbf{z}$ in $\mathbf{R}^d$. Then, for $k \in \mathbf{N}$,

$$\begin{aligned}
P(\mathbf{x}, B(\mathbf{x}, 1/k)) &= \mathbf{1}_{B(\mathbf{x},1/k)}(\mathbf{x}) \int_{\mathbf{y} \in \mathbf{R}^d} (1 - \alpha(\mathbf{x}, \mathbf{y})) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\quad + \int_{\mathbf{y} \in B(\mathbf{x},1/k)} \alpha(\mathbf{x}, \mathbf{y}) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&\geq \int_{\mathbf{y} \in \overline{B(\mathbf{x},1/2k)}} \alpha(\mathbf{x}, \mathbf{y}) q(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \\
&> 0
\end{aligned}$$

since $\overline{B(\mathbf{x}, 1/2k)}$ is compact, so q is positive on that set by regularity. Thus, the sequence $(B(\mathbf{x}, 1/k))_{k=1}^{\infty}$ satisfies the requirements in the previous lemma and proves that the Markov chain is aperiodic. $\square$

As an immediate corollary, one obtains

Corollary 2. *The Metropolis algorithm with normal proposals is aperiodic.*

Proof. This follows from the previous corollary since the normal distribution is regular. $\square$

Next consider irreducibility:

Proposition 2. *Consider a Metropolis-Hastings algorithm on $\mathbf{R}^d$ with target distribution $\pi(\cdot)$ that has a density with respect to the Lebesgue measure. Suppose that the proposal distribution is regular. Then the algorithm is π-irreducible.*

Proof. Let $B \subseteq \mathcal{X}$ such that $\pi(B) > 0$. Every set of positive measure in $\mathbf{R}^d$ contains a compact set of positive measure, so there exists a compact set $B' \subseteq B$ with $\pi(B') > 0$. Since $q(\cdot,\cdot)$ is bounded away from zero and infinity on compact sets,

$$\inf_{\mathbf{y} \in B'} \min\{q(\mathbf{x},\mathbf{y}), q(\mathbf{y},\mathbf{x})\} \geq \delta$$

for some $\delta > 0$. Hence, for any $\mathbf{x} \in \mathbf{R}^d$,

$$\begin{aligned} P(\mathbf{x}, B) &\geq P(\mathbf{x}, B') \\ &= \mathbf{1}_{B'}(\mathbf{x}) \int_{\mathbf{y} \in \mathcal{X}} (1 - \alpha(\mathbf{x},\mathbf{y})) q(\mathbf{x}, \mathrm{d}\mathbf{y}) + \int_{\mathbf{y} \in B'} \alpha(\mathbf{x},\mathbf{y}) q(\mathbf{x}, \mathrm{d}\mathbf{y}) \\ &\geq \int_{\mathbf{y} \in B'} \min\{q(\mathbf{x},\mathbf{y}), q(\mathbf{y},\mathbf{x})\} \mathrm{d}\mathbf{y} \\ &\geq \delta \lambda(\mathbf{y} \in B' : \pi(\mathbf{y}) \geq \pi(\mathbf{x})) + \varepsilon K \pi(\mathbf{y} \in B' : \pi(\mathbf{y}) < \pi(\mathbf{x})) \\ &> 0, \end{aligned}$$

where $\lambda(\cdot)$ denotes the Lebesgue measure of a set. The last inequality follows from the fact that $\pi(\cdot)$ has a density with respect to the Lebesgue measure and B' has positive Lebesgue-measure, so both terms cannot be zero. $\square$

Consequently, the following result holds:

Theorem 2. *Consider a Metropolis-Hastings algorithm where the target distribution as well as the proposal distribution are regular. Then the algorithm is ergodic towards its stationary distribution.*

Proof. Combine the previous corollaries to prove that the assumptions of the Markov chain convergence theorem are satisfied. □

In particular,

Corollary 3. *The Metropolis algorithm with normal proposals is ergodic, provided that the target distribution is regular.*

Proof. The normal distribution is regular, so the previous theorem applies. □

1.4 Central Limit Theorems

Even though the main focus in MCMC algorithms is to ensure that the algorithm is ergodic to the target distribution, there are other features that can be important. One of those is whether the Markov chain satisfies some central limit theorem (CLT).

Definition 11 (Central Limit Theorem)**.** *Consider a Markov chain $(X_n)_{n=0}^{\infty}$ on a state space $\mathcal{X}$ with stationary distribution $\pi(\cdot)$. Assume the Markov chain is ergodic and consider some functional $h : \mathcal{X} \to \mathbf{R}$ with finite stationary mean, i. e.*

$$\pi(h) := \int_{x \in \mathcal{X}} h(x)\pi(\mathrm{d}x) < \infty.$$

A Markov chain is said to satisfy a CLT if

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} (h(X_i) - \pi(h))$$

converges weakly to a normal distribution with zero mean and some finite variance σ^2 as $n \to \infty$.

Whether Markov chains satisfy a CLT is related to modes of convergence that are stronger than mere ergodicity. Two such properties are uniform and geometric ergodicity.

Definition 12 (Uniform Ergodicity)**.** *A Markov chain with stationary distribution $\pi(\cdot)$ is uniformly ergodic, if there exists $\rho \in (0,1)$ and $M \in \mathbf{R}^+$ such that for all $n \in \mathbf{N}$ and all $x \in \mathcal{X}$*

$$\|P^n(x,\cdot) - \pi(\cdot)\| \leq M\rho^n.$$

Definition 13 (Geometric Ergodicity)**.** *A Markov chain with stationary distribution $\pi(\cdot)$ is geometrically ergodic if there exists $\rho \in (0,1)$ and a function $M : \mathcal{X} \to \mathbf{R}^+$ such that for all $n \in \mathbf{N}$ and all π-a. e. $x \in \mathcal{X}$*

$$\|P^n(x,\cdot) - \pi(\cdot)\| \leq M(x)\rho^n.$$

It is immediate that uniform ergodicity implies geometric ergodicity which in turn implies ergodicity. For uniformly or geometrically ergodic Markov chains, there are some very simple conditions that guarantee a CLT:

Theorem 3. *Consider a uniformly ergodic Markov chain with stationary distribution $\pi(\cdot)$. If the functional h satisfies $\pi(h^2) < \infty$, then a CLT exists.*

Theorem 4. *Consider a geometrically ergodic Markov chain with stationary distribution $\pi(\cdot)$. If for some $\delta > 0$, the functional h satisfies $\pi(|h|^{2+\delta}) < \infty$, then a CLT exists.*

There are several other conditions that do not depend only on the finiteness of some moments of the functional and provide CLTs for Markov chains, see [15]. From a theoretical perspective, these theorems are appealing because they can be proved by a technique that is similar to the coupling technique which turned out to be useful in the proof of the Markov chain convergence theorem. This regeneration time method works as follows: Consider a Markov chain $(X_n)_{n=0}^{\infty}$. Furthermore, assume the existence of a minorisation condition with some small set C. This time, given X_n, it proceeds as follows:

(i) If $X_n \in C$, then with probability ε, choose $X_{n+n_0} \sim \nu(\cdot)$, or otherwise, choose
$$X_{n+n_0} \sim \frac{1}{1-\varepsilon}\left(\mathbf{P}^{n_0}(X_n,\cdot) - \varepsilon\nu(\cdot)\right).$$
Construct the intermediate steps $X_{n+1}, X_{n+2}, \ldots, X_{n+n_0-1}$ from the appropriate conditional distribution given X_n and X_{n+n_0}. Increase n by n_0.

(ii) If $X_n \notin C$, sample

$$X_{n+1} = X_n \sim P(X_n, \cdot).$$

and increase n by one.

The benefit of this procedure is that the Markov chain is split up into tours by the so-called regeneration times, i. e. the times $T_1, T_2, \dots$ where X_i has been sampled from $\nu(\cdot)$; this means that n_0 time steps earlier, the chain had entered the small set C and the case with probability ε occurred. Note that, by convention, entries into C within n_0 times steps of the last regeneration time are ignored. That way, sums can be rewritten as

$$\sum_{i=0}^{n} (h(X_i) - \pi(h)) = \sum_{j=1}^{r(n)} \sum_{i=T_j}^{T_{j+1}-1} (h(X_i) - \pi(h)) + E(n),$$

where $T_0 := 0$. Furthermore, $r(n) := \sup\{i \in \mathbf{N} : T_i \leq n\}$ is the number of regenerations that occurred until time n, and $E(n) := \sum_{i=T_{r(n)+1}}^{n} (h(X_i) - \pi(h))$ is the sum of the terms which occur after the last regeneration. The main point is to realize that the tours are independent, and the first step in each tour is being sampled from the distribution $\nu(\cdot)$; the tours are hence i. i. d. and allow the application of the conventional central limit theorem, provided the second moments of the variables exist, and the error term is negligible.

Using this construction, [15] give a short proof of theorem 4 for geometrically ergodic Markov chains. The proof of theorem 3 for uniformly ergodic Markov chains has been independently approached by [10] and [4]. The latter authors emphasize that the tours defined in a more general setting by

$$\sum_{j=1}^{r(n)} \sum_{i=mT_j}^{mT_{j+1}-1} (h(X_i) - \pi(h)) + E(n)$$

for some $m \in \mathbf{N}$ are not necessarily i.i.d. if $m \geq 2$, even if they start by sampling from $\nu(\cdot)$. Consequently, the conventional CLT cannot be directly applied. However, it remains the question whether a CLT holds in this situation:

Open Problem 2. *Does*

$$\frac{1}{\sqrt{n}} \sum_{j=1}^{r(n)} \sum_{i=mT_j}^{mT_{j+1}-1} (h(X_i) - \pi(h)) + E(n)$$

converge in distribution to a normal distribution with mean zero and some finite variance?

1.5 Drawbacks

Whenever a Metropolis-Hastings algorithm is employed, the question on which proposal distribution to chose arises. Naturally, one would like to select a proposal distribution in such a way that X_n and $pi(\cdot)$ are approximately equal as early as possible. To formalize this notion, the following quantity has to be defined:

Definition 14 (ε-convergence function)**.** *Consider an ergodic Markov chain with transition kernel $P(\cdot,\cdot)$ and stationary distribution $\pi(\cdot)$. For $\varepsilon > 0$ and $x \in \mathcal{X}$, define the ε-convergence function by*

$$M_\varepsilon(x) := \inf \left\{ n \in \mathbf{N} : \sup_{x \in \mathcal{X}} \|P^n(x,\cdot) - \pi(\cdot)\| \leq \varepsilon \right\}.$$

The smaller this quantity is for a fixed $x \in \mathcal{X}$ and $\varepsilon > 0$, the faster the convergence of the Markov chain will be[1].

The ε-convergence function does relate to other features of the MCMC algorithm. Assume that $\mathcal{X}$ is some subset $U \subseteq \mathbf{R}^d$, the target has a continuous density and there is a family of continuous proposals densities $q_\sigma(\cdot,\cdot)$ which differ only by a scale factor σ. If one chooses a proposal with a very small scale, the algorithm will usually propose values that are in a small neighborhood of the current state. Since the target and proposal densities are continuous, the density at the current state and the proposed state will be roughly the same, thereby leading to an acceptance rate that is close to one. However, the small step size will prevent the Markov chain from moving fast throughout the state space. Conversely, if the scale is very large, the proposal values are likely to be far away from the current state. Of course, the mass in the peripheral areas is small in the sense that for any $\mathbf{x} \in U$,

$$\lim_{M \to \infty} \int_{\mathbf{y} \in U / B(\mathbf{x}, M)} \pi(\mathbf{y}) \mathrm{d}\mathbf{y} = 0.$$

Therefore, the acceptance ratio will be small and once again there will be very little exploration of the state space. Evidently, in both cases a fast

[1]Those so-called quantitative bounds on the convergence rates have been studied in detail by several authors, see for example [18] and [5].

convergence of the transition probabilities to the stationary distribution is prevented. Since the acceptance rate can be easily checked by the applied statistician (whereas this cannot be done with the ε-convergence function), it is suggested as a general guideline to chose the proposal variance such that the acceptance rate will be neither to small nor too large. For a long time, this suggestions had to be considered as informal since there was no mathematical support. However, [14] were able to derive the optimal acceptance ratio in a specific setting.

This setting is mainly characterized by two restrictions: First, there are several regularity conditions in the sense that it is valid only for a Metropolis algorithm with normal proposals and only specific target distributions. Second, to derive the results, one considers a sequence of Metropolis algorithms in $\mathbf{R}^n$ where the dimension of the state space is increasing. In practice, this means that the result will be valid only in a large-dimensional context.

Suppose one has a sequence of target distributions which have densities with respect to the Lebesgue measure $\pi_n : \mathbf{R}^n \to \mathbf{R}$ taking the product form

$$\pi_n(\mathbf{x}^n) = \prod_{i=1}^{n} f(x_i^n),$$

where $\mathbf{x}^n = (x_1^n, x_2^n, \ldots, x_n^n)$ for $n \in \mathbf{N}$ and $f : \mathbf{R} \to \mathbf{R}$ which , for the moment, is an arbitrary function. For each $n \in \mathbf{N}$, the Metropolis algorithm with normal proposals q_n produces a Markov chain

$$\mathbf{X}^n = (\mathbf{X}_1^n, \mathbf{X}_2^n, \mathbf{X}_3^n, \ldots)$$

where $\mathbf{X}_i^n = \left(X_{i,0}^n, X_{i,1}^n, \ldots, X_{i,n}^n\right) \in \mathbf{R}^n$ for $i \in \mathbf{N}$, i. e. $X_{i,j}^n \in \mathbf{R}$ for $j \in \{0, 1, \ldots, i\}$. Then the following holds:

Theorem 5. *Suppose that f is positive, continuously differentiable, f/f' is Lipschitz continuous, and that*

$$\begin{aligned} \mathbf{E}_f\left[\left(\frac{f'(X)}{f(X)}\right)^8\right] &:= M < \infty, \\ \mathbf{E}_f\left[\left(\frac{f''(X)}{f(X)}\right)^4\right] &< \infty. \end{aligned}$$

Let $\mathbf{X}_0^\infty = (X_{0,1}^1, X_{0,2}^2, \ldots)$ be such that all its components are distributed according to f and assume that

$$\forall i \leq j : X_{0,j}^i = X_{0,j}^j$$

Then, as $n \to \infty$, the process

$$U_t^n := X_{\lfloor nt \rfloor, 1}^n$$

converges weakly in the Skorokhod topology to a stochastic process U which satisfies the stochastic differential equation

$$\mathrm{d}U_t = \sqrt{h(\ell)}\mathrm{d}B_t + h(\ell)\frac{f'(U_t)}{2f(U_t)}\mathrm{d}t,$$

where

$$h(\ell) = 2\ell^2 \Phi\left(-\frac{\ell}{2} \cdot \sqrt{\mathbf{E}_f\left[\left(\frac{f'(X)}{f(X)}\right)\right]}\right),$$

and U_0 is distributed according to f. Furthermore, the average acceptance rate

$$a_n(\ell) = \int\int \pi_n(\mathbf{x}^n)\alpha(\mathbf{x}^n, \mathbf{y}^n)q_n(\mathbf{x}^n, \mathbf{y}^n)\mathrm{d}\mathbf{x}^n\mathrm{d}\mathbf{y}^n$$

satisfies

$$\lim_{n\to\infty} a_n(\ell) = \frac{h(\ell)}{\ell^2}$$

which is maximized by

$$\hat{\ell} = \frac{2.38}{\sqrt{\mathbf{E}_f\left[\left(\frac{f'(X)}{f(X)}\right)\right]}}$$

with $a(\hat{\ell}) = 0.234$.

In other words, if the average acceptance rate is tuned such that it equals 0.234, the speed of the diffusion approximation is maximized, or equivalently, the Markov chain converges fastest to its stationary distribution.

Even though this result is illuminating, there are two drawbacks to it. First, since it is derived from a sequence of processes with increasing dimension of the target distribution, it can only be assumed to be valid if the state space is high-dimensional. However, [6] have shown by the means of a simulation study that the optimal acceptance ratio are close to the asymptotically optimal values even if the dimension is as low as 4. Second, in MCMC the transition kernel has been chosen in advance, but one usually has no information on how the choice of a transition kernel relates to the acceptance ratio. This drawback has led to the development of adaptive MCMC (AMCMC).

2 Adaptive Markov Chain Monte Carlo Algorithms

A means of addressing the problem of choosing the appropriate proposal covariance matrix in Metropolis-Hastings algorithms with normal proposals is AMCMC, where proposal covariance is not constant across iterations, but is being optimized. Therefore, a family of kernels $\{P_\gamma\}_{\gamma\in\mathcal{Y}}$ is used. Consequently, one needs not only an updating rule for the Markov chain (which is given by the kernel P_γ, and corresponds to the $\mathcal{X}$-valued random variable X_n), but also an updating rule for the kernel which is given by a $\mathcal{Y}$-valued random variable Γ_n (representing the choice of kernel when updating from X_n to X_{n+1}).To be formally correct, introduce the filtration $\mathcal{G}_n := \sigma(X_0, \ldots, X_n, \Gamma_0, \ldots, \Gamma_n)$. Then

$$P\left(X_{n+1} \in A | X_n = x, \Gamma_n = \gamma, \mathcal{G}_{n-1}\right) = P_\gamma(x, B),$$

where $x \in \mathcal{X}$, $\gamma \in \mathcal{Y}$, and $B \in \mathcal{F}$. Also, the conditional distribution of Γ_{n+1} given $\mathcal{G}_n$ needs to be specified. The quantity of interest is then

$$A^{(n)}\left((x, y), B\right) := P(X_n \in B | X_0 = x, \gamma_0 = \gamma)$$

for $B \in \mathcal{F}$. Note that this is not a Markov process. The concept of ergodicity, however, can be transfered to this stochastic process.

Definition 15 (Ergodicity of AMCMC)**.** *An AMCMC algorithm is ergodic if*

$$\lim_{n\to\infty} \sup_{(x,\gamma)\in\mathcal{X}\times\mathcal{Y}} \left\| A^{(n)}((x,\gamma),\cdot) - \pi(\cdot) \right\| = 0.$$

2.1 Algorithms

Generally, adaptive versions of existing MCMC algorithms can be easily constructed. The most popular algorithm is the adaptive Metropolis algorithm, first proposed by [7], and is, as suggested by its name, an adaptive version of the Metropolis algorithm on $\mathbf{R}^d$. The family of proposal distributions is now given by a two-parameter family where not only the center of the distribution (given by the current state), but also the covariance matrix is variable. Indeed, the covariance matrix Σ_n used in the nth step is given by

$$\Sigma_n = \begin{cases} \Sigma_0 & n \le n_0 \\ \frac{(2.38)^2}{d}\left(\mathrm{Cov}(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n) + \varepsilon \mathbf{I}_d\right) & n > n_0. \end{cases}$$

where $\mathbf{X}_0, \mathbf{X}_1, \ldots$ are the states of the stochastic process, Σ_0 is a fixed starting covariance matrix and $n_0 \in \mathbf{N}$ is the fixed length of the initial phase. Note that $\mathbf{I}_d$ denotes the $d \times d$-identity matrix. As usual,

$$\operatorname{Cov}(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n) := \frac{1}{n} \sum_{i=0}^{n} (\mathbf{X}_i - \overline{\mathbf{X}}_n)(\mathbf{X}_i - \overline{\mathbf{X}}_n)'.$$

where $\overline{\mathbf{X}}_n$ is the arithmetic mean of the iterations $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n$. Note that the ε-multiple of the identity matrix is added to prevent the covariance matrix from getting singular. Therefore, the proposal covariance is merely a modification of the covariance matrix of the first n iterations. In the first few trials, the proposal covariance matrix is held fixed since the initial states of the Markov chain are expected not to lead to good estimates of the covariance matrix. Note that the choice of the constant $(2.38)^2/d$ is a result of theorem 5 and reflects an attempt to approach the most efficient proposal covariance matrix.

A slightly modified version has been suggested by [17]. The covariance matrix is now calculated by

$$\Sigma_n' = \begin{cases} \Sigma_0 & n \leq 2d \\ (1-\beta)^2 \frac{(2.38)^2}{d} \operatorname{Cov}(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n) + \beta^2 \frac{(0.1)^2}{d} \mathbf{I}_d & n > 2d \end{cases}$$

for some $\beta \in (0, 1)$.

2.2 Ergodicity

Whenever one expects an AMCMC algorithm to be ergodic, it is sensible to assume that for each $\gamma \in \mathcal{Y}$, the kernel P_γ is ergodic for $\pi(\cdot)$; this will be assumed throughout the remaining sections. The first thing to notice is that this is not sufficient for an adaptive algorithm to be ergodic for $\pi(\cdot)$. A counterexample is given by [16], example 2. Obtaining sufficient conditions for an AMCMC algorithm to be ergodic is usually a difficult task. A key concept, however, is that of diminishing adaptation which largely states that, asymptotically, the amount of adaptation tends to zero.

Definition 16 (Diminishing adaptation). *Let $A^{(n)}((x, \gamma), \cdot)$ be an adaptation algorithm. It satisfies the diminishing adaptation condition if*

$$\lim_{n \to \infty} \sup_{x \in \mathcal{X}} \left\| P_{\Gamma_{n+1}}(x, \cdot) - P_{\Gamma_n}(x, \cdot) \right\| =^d 0.$$

The second class of conditions usually needed is some kind of assumption on the family of proposal distribution $\{P_\gamma\}_{\gamma\in\mathcal{Y}}$. The next subsections give an overview over conditions that suffice for an AMCMC algorithm to be ergodic. First, results that apply on general spaces $\mathcal{X}\times\mathcal{Y}$ will be reported. In the second section, compactness assumptions are imposed on (parts of) the space. The final section deals with the adaptive Metropolis algorithm which is defined for subsets of $\mathbf{R}^d$.

2.2.1 General Spaces

A straightforward but rather strong condition is presented by [16]:

Theorem 6 (Simultaneous Uniform Ergodicity). *Consider an AMCMC algorithm which satisfies diminishing adaptation and simultaneous uniform ergodicity, that is, for all $\varepsilon > 0$, there exists $n_0 \in \mathbf{N}$ such that*

$$\forall\gamma\in\mathcal{Y}, x\in\mathcal{X} : \left\|P_\gamma^{n_0}(x,\cdot)-\pi(\cdot)\right\| \le \varepsilon.$$

Then the algorithm is ergodic.

Proof. See theorem 1 of [16]. □

However, it turns out that in their proof, the following and slightly weaker condition suffices to prove ergodicity:

Proposition 3. *Define the ε-convergence function $M_\varepsilon(x,\gamma) : \mathcal{X}\times\mathcal{Y}\to\mathbf{N}$ by*

$$M_\varepsilon(x,\gamma) := \inf\left\{n\in\mathbf{N} : \left\|P_\gamma^{n}(x,\cdot)-\pi(\cdot)\right\| \le \varepsilon\right\}.$$

An adaptive algorithm is ergodic if $\{M_\varepsilon(X_n,\Gamma_n)\}_{n=0}^\infty$ is bounded in probability for all $\gamma\in\mathcal{Y}$ and $x\in\mathcal{X}$.

Proof. See [16], theorem 2. □

Another approach relates to the minorisation criterion mentioned in previous sections and the existence of drift-functions.

Definition 17 (Drift function). *A Markov chain with transition kernel $P(\cdot,\cdot)$ satisfies a drift condition if there are constants $\lambda\in(0,1)$ and $b<\infty$, and a function $V:\mathcal{X}\to[1,\infty]$ such that*

$$\forall x\in\mathcal{X} : 8PV9(x) \le \lambda V(x) + \mathbf{1}_C(x);$$

where for $x \in \mathcal{X}$ and a function f,

$$(Pf)(x) := \int_{y \in \mathcal{X}} f(y) P(x, \mathrm{d}y).$$

In ordinary Markov chain theory, this condition is important because a Markov chain that satisfies the assumptions of the Markov chain convergence theorem, a minorisation condition and has a drift function, can be shown to be not only ergodic, but geometrically ergodic [15]. However, adaptations of this concept are useful to prove ergodicity of AMCMC algorithms.

Theorem 7 (Simultaneous Strong Aperiodic Geometric Ergodicity)**.** *Consider an adaptive algorithm such that the family of transition kernels $\{P_\gamma\}_{\gamma \in \mathcal{Y}}$ is simultaneously strongly aperiodically geometrically ergodic, that is, there exist a set $C \in \mathcal{F}$, a function $V : \mathcal{X} \to [1, \infty)$, and constants $\delta > 0$, $\lambda \in (0, 1)$, and $b < \infty$ such that*

(i) (Minorisation condition). For all $\gamma \in \mathcal{Y}$ there exists a probability measure $\nu_\gamma(\cdot)$ on C with

$$\forall x \in C : P_\gamma(x, \cdot) \geq \delta \nu_\gamma(\cdot)$$

(ii) (Drift function). $\forall \gamma \in \mathcal{Y}, x \in \mathcal{X} : (P_\gamma V)(x) \leq \lambda V(x) + b \mathbf{1}_C(x)$.

(iii) $\sup_C V < \infty$.

Furthermore, assume $\mathbf{E}[V(X_0)] < \infty$. Then the adaptive algorithm is ergodic.

Proof. See [16], theorem 3. $\square$

As with the first theorem, there is a set of weaker conditions that guarantee the same result but may be less convenient to work with.

Proposition 4 (Simultaneous Polynomial Ergodicity)**.** *Consider an adaptive algorithm with a simultaneously polynomially ergodic family of transition kernels $\{P_\gamma\}_{\gamma \in \mathcal{Y}}$, i. e.*

(i) There exists a set $C \in \mathcal{F}$, constants $n_0 \in \mathbf{N}$, $\delta > 0$, and probability measures $\nu_\gamma(\cdot)$ on $\mathcal{X}$ such that $\pi(C) > 0$ and

$$\forall \gamma \in \mathcal{Y}, x \in \mathcal{X} : P_\gamma^{n_0}(x, \cdot) \geq \delta \nu_\gamma(\cdot).$$

(ii) There exists $q \in \mathbf{N}$ and measurable functions $\{V_i : \mathcal{X} \to (0,\infty)\}_{i=0}^{k}$, such that for $k = 0, 1, \ldots, q-1$, there are sequences $a_k \in (0,1)$, $b_k < 1$, and $c_k > 0$, and a constant $\beta \in (0,1)$ such that

$$\begin{gathered}\forall \gamma \in \mathcal{Y}, x \in \mathcal{X} : (P_\gamma V_{k+1})(x) \le V_{k+1}(x) - V_k(x) + b_k \mathbf{1}_C(x),\\ \forall x \in \mathcal{X} : V_k(x) \le c_k,\\ \forall x \in \mathcal{X}/C : V_k(x) - b_k \ge a_k V_k(X),\\ \sup_C V_q \ < \ \infty,\\ \pi(V_q^\beta) < \infty.\end{gathered}$$

Then the adaptive algorithm is ergodic.

Proof. See [16], theorem 4. □

Finally, there are the following two sets of conditions.

Theorem 8. *Consider an adaptive algorithm that satisfies the diminishing adaptation condition. Assume that*

(i) (Minorisation condition). There exists a probability measure $\nu(\cdot)$, a constant $\varepsilon > 0$, and a set $C \in \mathcal{F}$ such that

$$\forall x \in C : P_\gamma(x,\cdot) \ge \varepsilon \nu(\cdot).$$

(ii) (Drift function). There exists a measurable function $V : \mathcal{X} \to [1,\infty)$ and a constant $b > 0$ such that for any $\gamma \in \mathcal{Y}$

$$\forall \gamma \in \mathcal{Y}, x \in \mathcal{X} : (P_\gamma V)(x) \le V(x) - 1 + b\mathbf{1}_C(x).$$

(iii) For any sublevel set $\mathcal{D}_\ell := \{V(x) \le \ell\}$ of V,

$$\lim_{n\to\infty} \sup_{\mathcal{D}_\ell \times \mathcal{Y}} \|P_\gamma(x,\cdot) - \pi(\cdot)\| = 0.$$

Proof. See [2], theorem 4.3. □

Theorem 9. *Suppose an adaptive algorithm satisfies the diminishing adaptation condition. Then it is ergodic under each of the following cases:*

(i) The algorithm is simultaneously polynomially ergodic and the number q of drift functions is strictly greater than two;

(ii) The algorithm is simultaneously polynomially ergodic, the number q of drift functions is strictly greater than one, and there exists a increasing function $f : \mathbf{R} \to \mathbf{R}$ such that $V_1(\cdot) \leq f(V_0(\cdot))$.

(iii) There is a set $C \subseteq \mathcal{X}$, some integer $n_0 \in \mathbf{N}$, some constant $\varepsilon > 0$, and some probability measure $\nu_\gamma(\cdot)$ on $\mathcal{X}$ such that $\pi(C) > 0$, and

$$\forall x \in \mathcal{X}, \gamma \in \mathcal{Y} : P_\gamma^{n_0}(x, \cdot) \geq \varepsilon \mathbf{1}_C(X)\nu_\gamma(\cdot).$$

Furthermore, there exists $\beta \in (0, 1]$ such that $\pi(V_q^\beta) < \infty$, and the simultaneous drift condition has the form

$$\forall \alpha \in (0, 1), x \in \mathcal{X} : P_\gamma V_1(x) \leq V(x) - cV^\alpha(x) + b\mathbf{1}_C(x)$$

where $cV_\alpha(x) \geq b$ on C^c.

(iv) There is a set $C \subseteq \mathcal{X}$, an integer $n_0 \in \mathbf{N}$, a constant $\varepsilon > 0$, and for each $\gamma \in \mathcal{Y}$ a probability measure $\nu_\gamma(\cdot)$ on $\mathcal{X}$ such that $\pi(C) > 0$, and

$$\forall x \in C : P_\gamma^{n_0}(x, \cdot) \geq \varepsilon\nu_\gamma(\cdot).$$

and when the simultaneous drift condition has the form

$$\forall x \in \mathcal{X} : P_\gamma V_1(x) \leq V_1(x) - V_0(x) + b\mathbf{1}_C(x).$$

where $V_0(x) \leq b$ on C^c, and the process $(V_1(X_n))_{n=0}^\infty$ is bounded in probability.

Then the adaptive algorithm is ergodic.

Proof. See [2], theorem 5.7, 5.8, 5.10, and lemma 5.2. □

2.2.2 Compact Spaces

In this section, results that require (parts of) the space $\mathcal{X} \times \mathcal{Y}$ to be compact in some topology are presented. The first proposition can be easily derived from proposition 3.

Proposition 5. *Consider an adaptive algorithm that satisfies the diminishing adaptation condition. Furthermore, assume that there is some topology such that (i) $\mathcal{X} \times \mathcal{Y}$ is compact and (ii) the mapping*

$$(x, \gamma) \mapsto \sup_{\mathcal{X}\times\mathcal{Y}} \left\| A^{(n)}((x, \gamma), \cdot) - \pi(\cdot) \right\|$$

is continuous for every $n \in \mathbf{N}$. Then the adaptive algorithm is ergodic.

Proof. See [16], corollary 3. □

A drawback of the previous proposition is that it requires the product space $\mathcal{X} \times \mathcal{Y}$ to be compact. A condition that requires only the space $\mathcal{Y}$ to be compact is provided by [21].

Theorem 10. *Consider an adaptive algorithm satisfying the diminishing adaptation condition. Assume that*

(i) (Minorisation and drift condition.) There is $C \in \mathcal{F}$, $V : \mathcal{X} \to [1, \infty)$ with $\pi(V) < \infty$, $\varepsilon > 0$, and $b > 0$ such that, first, $\sup_C V < \infty$, second, for each $\gamma \in \mathcal{Y}$ there exists a probability measure $\nu_\gamma(\cdot)$ on C with

$$\forall x \in C : P_\gamma(x, \cdot) \geq \varepsilon \nu_\gamma(\cdot),$$

and third,

$$\forall \gamma \in \mathcal{Y}, x \in \mathcal{X} : (P_\gamma V)(x) \leq V(x) - 1 + b\mathbf{1}_C(x).$$

(ii) $\mathcal{Y}$ is compact in the metric

$$d(\gamma_1, \gamma_2) := \sup_{x \in \mathcal{X}} \|P_{\gamma_1}(x, \cdot) - P_{\gamma_2}(x, \cdot)\| .$$

(iii) The sequence $\{V(X_n)\}_{n=0}^{\infty}$ is bounded in probability given X_0 and Γ_0

Then the adaptive algorithm is ergodic.

Proof. See [21], theorem 3.1, and [2], theorem 3.3. □

In most cases, however, not even the space $\mathcal{Y}$ will be compact so that one has to use the general criteria provided in the previous section. The theory is more developed for adaptive Metropolis algorithms.

2.2.3 Adaptive Metropolis Algorithm

In this section, sufficient criteria for the adaptive Metropolis algorithm (defined on $\mathbf{R}^d$) are presented. Furthermore, there are certain restrictions: First, take $\mathcal{X}$ to be an open subset $U \subseteq \mathbf{R}^d$ and $\mathcal{F}$ to be the Borel sets of $\mathbf{R}^d$. Assume that the target distribution has a density with respect to the Lebesgue measure and is continuously differentiable, positive, and regular. Also, only adaptive Metropolis algorithms with symmetrical proposal,

$$\forall \gamma \in \mathcal{Y}, \mathbf{x}, \mathbf{y} \in U : q_\gamma(\mathbf{x} - \mathbf{y}) = q_\gamma(\mathbf{y} - \mathbf{x}),$$

are considered.

As results on the geometric ergodicity of symmetric Metropolis algorithms, the ergodicity of the adaptive Metropolis algorithm is strongly related to the tail behavior of the density [13]. A criterion for lighter-than-exponential tails is

Theorem 11 (Lighter-than-exponential tails)**.** *If a symmetrical adaptive Metropolis algorithm satisfies the diminishing adaptation and the following conditions:*

(i) (Lighter-than-exponential, strongly decreasing target.) The target density π is lighter-than-exponentially tailed,

$$-\limsup_{|\mathbf{x}|\to\infty} \left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \nabla \log \pi \right\rangle = \infty.$$

and strongly decreasing,

$$-\limsup_{|\mathbf{x}|\to\infty} \left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\nabla \mathbf{x}}{|\nabla \mathbf{x}|} \right\rangle > 0.$$

(ii) (Proposal's local positivity.) For each $\gamma \in \mathcal{Y}$, there exists $\delta_\gamma > 0$ and $\varepsilon_\gamma > 0$ such that

$$\forall \gamma \in \mathcal{Y}, |\mathbf{z}| \le \delta_\gamma : q_\gamma(\mathbf{z}) \ge \varepsilon_\gamma,$$

and there is a positive function $q^- : \mathbf{R}^d \to \mathbf{R}^+$ bounded away from zero in any compact set such that

$$\forall \gamma \in \mathcal{Y}, \mathbf{z} \in \mathbf{R}^d : q_\gamma \ge q^-(\mathbf{z}).$$

Then the adaptive algorithm is ergodic.

Proof. See [3], theorem 5.2. □

Somewhat stronger conditions are needed if the target distribution has exponential tails.

Theorem 12 (Exponential tails). *If a symmetrical adaptive Metropolis algorithm satisfies the diminishing adaptation and the following conditions:*

(i) (Exponentially tailed, strongly decreasing target. The target density π is lighter-than-exponentially tailed,

$$\eta_1 := -\limsup_{|\mathbf{x}|\to\infty} \left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \nabla \log \pi \right\rangle > 0,$$

and strongly decreasing,

$$\eta_2 := -\limsup_{|\mathbf{x}|\to\infty} \left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\nabla \mathbf{x}}{|\nabla \mathbf{x}|} \right\rangle > 0.$$

(ii) There are constants $\varepsilon \in (0, \eta_1)$, $\beta \in (0, \eta_2)$, $\delta > 0$, and $\Delta \in (0, \infty]$ with $0 < \frac{3}{\beta\varepsilon} \leq \delta < \Delta$ such that for any sequence $(\mathbf{x}_n, \gamma_n)$ with $|\mathbf{x}_n| \to \infty$ and $\gamma_n \in \mathcal{Y}$, there exists a subsequence $(\mathbf{x}_{n_k}, \gamma_{n_k})$ with $\mathbf{x}_{n_k} \to \infty$ such that

$$\lim_{k\to\infty} \int_{\left\{\mathbf{z}=a\mathbf{w} \middle| \delta \leq a \leq \Delta, \mathbf{w} \in \mathbf{S}^{n-1}, \left|\xi - \frac{\mathbf{x}_{n_k}}{|\mathbf{x}_{n_k}|}\right| < \varepsilon/3\right\}} |\mathbf{z}| q_{\gamma_{n_k}}(\mathbf{z}) \mathrm{d}\mathbf{z} > \frac{3}{\beta\varepsilon(e-1)},$$

where $\mathbf{S}^{n-1}$ is the unit sphere in $\mathbf{R}^n$, and $a\mathbf{w}$ represents a scalar multiple of $\mathbf{w} \in \mathbf{R}^n$ by $a \in \mathbf{R}$.

Then the adaptive algorithm is ergodic.

Proof. See [3], lemma 5.2 and theorem 5.1. □

The condition can be replaced by a stronger condition that is easier to verify, see [3]. If the tails are even heavier than exponential, there is, so far, no result that applies to arbitrary symmetric adaptive Metropolis algorithms. However, there is one result in the case that the proposal distribution is uniform on some compactly supported set and the target distribution has hyperbolic tails.

Theorem 13 (Hyperbolic tails). *Consider a symmetrical adaptive Metropolis algorithm satisfying the diminishing adaptation condition and the following conditions:*

(i) (Hyperbolically tailed, strongly decreasing target). The target density π is twice differentiable, strongly decreasing,

$$-\limsup_{|\mathbf{x}|\to\infty} \left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\nabla \mathbf{x}}{|\nabla \mathbf{x}|} \right\rangle > 0,$$

and hyperbolically tailed, that is, there exists $m \in (0,1)$ and finite constants c_i, C_i for $i = 0,1,2$ and $M > 0$ such that for all $|\mathbf{x}| \geq M$,

$$\begin{gathered}0 < c_0|\mathbf{x}|^m \leq -\log \pi(\mathbf{x}) \leq C_0|\mathbf{x}|^m,\\ 0 < c_1|\mathbf{x}|^{m-1} \leq -|\nabla \log \pi(\mathbf{x})| \leq C_1|\mathbf{x}|^{m-1},\\ 0 < c_2|\mathbf{x}|^{m-2} \leq -|\nabla^2 \log \pi(\mathbf{x})| \leq C_1|\mathbf{x}|^{m-2}.\end{gathered}$$

(ii) (Proposal's Uniform Compact Support and Uniform Upper Bound Density.) The proposal has a uniform compact support, that is, there exists a $L > 0$ such that

$$\forall \gamma \in \mathcal{Y}, |\mathbf{z}| > L : q_\gamma(\mathbf{z}) = 0,$$

and a uniform upper bound density, there exists a positive function $q^+ : \mathbf{R}^d \to \mathbf{R}^+$ with $\int q^+(\mathbf{z})\mathrm{d}\mathbf{z} < \infty$ such that

$$\forall \gamma \in \mathcal{Y}, \mathbf{x} \in \mathbf{R}^d : q_\gamma(\mathbf{x}) \leq q^+(\mathbf{x}).$$

Then the adaptive algorithm is ergodic.

Proof. See [3], theorem 5.4. □

Finally, one can apply these results to the adaptive Metropolis algorithm with normal proposals. If the tails of the target distribution are lighter-than-exponential, the diminishing adaptation condition can be checked directly. Therefore, the following theorem holds:

Theorem 14. *The adaptive Metropolis algorithm with normal proposals is ergodic provided that the target distribution is regular, strongly decreasing, and lighter-than-exponentially tailed.*

Proof. See [3], theorem 5.3. □

3 A Simulation Study

Most AMCMC algorithms suggest to adapt the proposal in each iteration. However, the authors in [7] raised the question whether it is more efficient to perform adaptations only every k_0 iterations for some fixed $k_0 \in \mathbf{N}$. As mentioned above, even if the algorithm is assumed to be ergodic, the trials from early iterations cannot be expected to be drawn from a distribution which is close to the target distribution and might, therefore, distort the estimate of the covariance matrix. The authors [7] suggested to use only the last half of the iterations to avoid this problem. In general, one could use the last $\lfloor n/k_0 \rfloor$ iterations where n is the current iteration and $k_0 \in \mathbf{N}$ is fixed (so $k_0 = 2$ if the last half of the iterations is being used). By taking a fixed proportion of the last iterations as opposed to a fixed number of the last iterations, diminishing adaptation is fulfilled which is essential in all the theorems of the last section to prove the ergodicity of the adaptive algorithm. Note, however, that diminishing adaptation is not a necessary condition for convergence.

Lemma 7. *Consider an adaptive Metropolis algorithm with normal proposals. If the target is exponentially-tailed, regular, and strongly decreasing, then the adaptive algorithm is ergodic.*

Proof. The proof is practically identical to proposition 5.3 and theorem 5.3 in [3]. The only difference is that the evaluation of the difference $\Sigma_{n+1} - \Sigma_n$ leads to numerous terms which, however, can all be shown to be bounded in probability by the methods used in the proof. □

However, it remains the following question:

Open Problem 3. *Does diminishing adaptation hold for arbitrary adaptive Metropolis algorithms with normal proposals?*

3.1 Method

To study whether the different strategies in updating the proposals do affect the efficiency of the algorithm, a simulation study was performed. As a baseline, the standard algorithm which updates the proposal covariance matrix each iteration and makes use of all previous iterations was run. The algorithm updating the proposal only each k_0 iterations was run with $k_0 = 2$ and

$k_0 = 10$ and the algorithm which uses only the last $\lfloor 1/k_0 \rfloor$ of the iterations was run with $k_0 = 2$ and $k_0 = 10$.

For an efficient implementation of the algorithm, [7] used an easy recursive formula to update the covariance matrix; for the modified algorithms, these recursions had to be modified (see appendix A).

For the target, a distribution resulting from a statistical application was chosen to construct a situation which is typical for those emerging in the work of applied statisticians. It results from analyzing the quadratic model

$$Y_{ij} = \alpha + \beta x_i + \gamma x_i^2 + \varepsilon_{ij} \quad i \in \{1, , \dots, k\}, j \in \{1, , \dots, n_i\},$$

from a Bayesian perspective. The error terms are assumed to follow a t-distribution with mean 0, variance σ^2, and 4 degrees of freedom,

$$\varepsilon_{ij} \sim t_4(0, \sigma^2) \quad (\sigma > 0).$$

The purpose is to estimate (properties of) the posterior distribution of the parameter vector $\theta := (\alpha, \beta, \gamma, \sigma^2)$. The prior distributions are set to be

$$\begin{aligned} \alpha &\sim N(m_1, v_1^2) \quad (m_1 \in \mathbf{R}, v_1 > 0), \\ \beta &\sim N(m_2, v_2^2) \quad (m_2 \in \mathbf{R}, v_2 > 0), \\ \gamma &\sim N(m_3, v_3^2) \quad (m_3 \in \mathbf{R}, v_3 > 0), \\ \sigma^2 &\sim IG(s, t) \quad (s, t > 0). \end{aligned}$$

The posterior distribution is then easily calculated as

$$\begin{aligned} \pi(\theta) \quad \propto \quad & \exp\left(-\frac{1}{2}\left(\frac{(\alpha - m_1)^2}{v_1^2} + \frac{(\beta - m_2)^2}{v_2^2} + \frac{(\gamma - m_3)^2}{v_3^2}\right) - \frac{t}{\sigma^2}\right) \cdot \\ & \sigma^{-2s-2-N} \cdot \prod_{i=1}^{k} \prod_{j=1}^{n_i} \left(1 + \frac{(y_{ij} - \alpha - \beta x_i - \gamma x_i^2)^2}{\nu}\right)^{-(\nu+1)/2} . \qquad (3) \end{aligned}$$

Note that this distribution cannot be represented in closed form. Its characteristics, such as moments, are not easily computed, and the dimension of the parameter space $\Theta := \mathbf{R}^3 \times \mathbf{R}^+$ is 4. Therefore, it is sufficiently high to make numerical integration as well as other methods to sample from complicated distributions such as rejection sampling or importance sampling unattractive, so that using MCMC methods is indicated. The model is fit to data which have been reported in [20].

The adaptive Metropolis algorithm shown in equation 2.1 was used for the simulation. Each model was run with one million iterations and a value of

Table 1: The performance indicators for the standard algorithm (S), algorithms updating only every k_0 iterations (E2 and E10, where the number indicates the values of k_0), and the algorithms which only use a fraction of the iterations (L2, L10) to calculate the new proposal variance when the initial phase has length 8.

condition	S	E2	E10	L2	L10
acceptance rate	0.171	0.166	0.158	0.194	0.196
average jumping distance	10.27	9.90	10.02	9.41	9.28
computing time [min:sec]	40:37	40:41	38:20	40:42	40:50
M_α	8.320	8.288	8.362	8.298	8.375
SD_α	0.02649	0.02734	0.02599	0.02643	0.02591
M_β	-0.4033	-0.4004	-0.4114	-0.4008	-0.4091
SD_β	0.004047	0.004093	0.004019	0.003994	0.003930
M_γ	0.133220	0.133166	0.133556	0.133120	0.133393
SD_γ	0.0001570	0.0001588	0.0001587	0.0001551	0.0001529
M_{σ_2}	0.124441	0.124308	0.124405	0.124389	0.124453
SD_{σ_2}	0.0001698	0.0001639	0.0001603	0.0001714	0.0001681

Note. *M.* and *SD.* denote estimates of the mean and the standard deviation of the respective parameters.

β of 0.05 was used. Several indices were used to evaluate the performance of the different algorithms. To describe them, let $(x_0, \gamma_0), (x_1, \gamma_1), \ldots, (x_n, \gamma_n)$ be the realizations of a run of length n of the stochastic process. The proportion of accepted proposals (acceptance ratio, AR), the mean of the distance between consecutive iterations of the chain (average jumping distance, AJ); note that iterations with rejected proposals *are* considered in this average, so

$$\text{AJ} = \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_2,$$

where $\|\cdot\|_2$ denotes the Euclidean distance. Finally, the estimated standard errors for the parameters $\theta = (\alpha, \beta, \gamma, \sigma^2)$ and the time the computer needed to process the data.

The simulations were run using R [19], version 2.7.1. See appendix B for the corresponding code. The computing time was measured using the command `system.time`.

3.2 Results

Plots of the iteration of the Markov chain versus its value indicate that the Markov chain converged (see appendix C). Table 1 gives an overview of the algorithms and how they performed with respect to the performance indicators. The acceptance rate for the standard algorithm was 0.17. In conditions where the adaptation of the proposal covariance was done only every k_0 iterations, the acceptance rate was slightly lower (0.158 to 0.166), whereas if only the last $1/k_0$ iterations are being used to determine the proposal covariance, the acceptance rates was higher (0.194 to 0.196). In contrast, the average jumping distance was highest in the standard condition (10.3). If fewer updates were done, the average jumping distance was only slightly lower (9.9 to 10.0), but it was substantially lower in the condition that uses less information (9.3 to 9.4). This result is inconsistent since one usually expects that an optimal acceptance ratio is associated to a high average jumping distance [17]. The estimated parameters differ only slightly among the different conditions and the estimated standard errors tend to be similar across the conditions. There is no consistent advantage of any of the three types of algorithms. Finally, the differences in computing time showed the following pattern: If the updates are only done each ten iterations, the algorithm was fastest (38:20 minutes), whereas the remaining algorithms needed between 40:37 and 40:50 minutes for the computations. The time savings of the fastest algorithm compared to the standard algorithm is 6%.

Unexpectedly, there was very little variation in the performance of the various algorithms. One possible explanation is that the initial phase in which no adaptations are being done is too short with only $2d = 8$ iterations. The early start of the adaptation might lead to poor initial estimates of the proposal covariance matrix and hence distort the results. Therefore, part of the simulations were rerun with an initial phase of $50d = 200$ iterations. Since the effectivity of the algorithms with fewer updates of the proposal variance did not depend on the rate of the updates, and the algorithm using only a part of the last iterations did not depend on the fraction of the data that were used, only the algorithms updating the proposal every second iteration and the algorithm using the last half of the iterations were rerun.

Once again, the plots of iterations versus value of the Markov chain indicate that the Markov chain converges (see appendix C). The results of this simulation are shown in table 2. All the algorithms perform similar compared to their counterparts with the shorter initial period. The standard algorithm has a slightly smaller acceptance rate (0.16 vs. 0.17), whereas the average jumping distance is practically left unchanged. For the algorithm that updates the proposal only every second iteration, the acceptance rate is almost identical, whereas the average jumping distance is slightly higher (10.4 vs. 9.9). Finally, both the acceptance ratio and the average jumping distance are unchanged for the algorithm using only the last half of the it-

Table 2: The performance indicators for the standard algorithm (S), algorithms updating only every k_0 iterations (E2 and E10, where the number indicates the values of k_0), and the algorithms which only use a fraction of the iterations (L2, L10) to calculate the new proposal variance when the initial phase has length 200.

condition	S	E2	L2
acceptance rate	0.159	0.167	0.194
average jumping distance	10.30	10.37	9.44
computing time [min:sec]	39:11	39:56	40:35
M_α	8.286	8.271	8.322
SD_α	0.02679	0.02667	0.02698
M_β	-0.4016	-0.3956	-0.4077
SD_β	0.004112	0.004112	0.004026
M_γ	0.133232	0.132928	0.133463
SD_γ	0.0001600	0.0001604	0.0001553
M_{σ_2}	0.124368	0.124497	0.124267
SD_{σ_2}	0.0001675	0.0001639	0.0001714

Note. $M.$ and $SD.$ denote estimates of the mean and the standard deviation of the respective parameters.

erations to determine the proposal. As in the previous simulations, for any of the algorithms, the estimates of the parameters and their standard errors show very little variation. For all the three algorithms, the computing time tends to be faster when the initial phase is lengthened. The computing times are between 39:11 and 40:35 if the initial phase is short, and between 40:37 and 40:42 if the initial phase is long. It might be tempting to attribute the reduced computation time to the fact that the initial phase, in which no time-consuming adaptations need to be done, is longer. However, recall that since 1,000,000 iterations were done in total, the actual difference between the length of the initial phases should not have a significant effect.

3.3 Conclusion

The conclusion from the simulation studies that have been performed so far did not give conclusive evidence that any of the modifications applied to the process of updating the proposal covariance matrix has significant effects on the performance of the algorithms. However, most of the algorithms did not show any evidence of being more computationally efficient, with the notable exception of the algorithm that updates the proposal covariance only every tenth iteration. The savings of roughly 6% of the computing time with respect to the standard algorithm might not be worth the additional effort in implementing the algorithm if the simulation can be run in as little time as in the previous case; but if a simulation needs to be run for weeks, or even month, the absolute savings in computing time are substantial. However, so far this conclusion is limited to only a single Markov chain that has been investigated. Hence, future research that generalizes these finding is needed. Moreover, future explorations of how the changes of the updating process affect effectivity and efficiency of the algorithm might focus on how little adaptations an algorithm tolerates without showing a significant decay in the performance.

Acknowledgements

I thank Professor Jeffrey S. Rosenthal for supervising me on the reading course which led to this report and providing me with help when I needed it. Also, I thank Yi Su and Matthew Thompson for proofreading this report which led to many improvements.

References

[1] Krishna Balasundaram Athreya, Hani Doss, and Jayaram Sethuraman, *On the convergence of the Markov chain simulation method*, The Annals of Statistics **24** (1996), no. 1, 69–100.

[2] Yan Bai, *Simultaneous drift conditions for adaptive Markov chain Monte Carlo algorithms*, Unpublished Manuscript. Department of Mathematics, University of Toronto, Toronto, Canada, 2009.

[3] Yan Bai, Gareth Owen Roberts, and Jeffrey Seth Rosenthal, *On the containment condition for adaptive Markov chain Monte Carlo algorithms*, Unpublished Manuscript. Department of Mathematics, University of Toronto, Toronto, Canada, 2009.

[4] Witold Bednorz, Krzystof Łatuszyński, and Rafał Latała, *A regeneration proof of the central limit theorem for uniformly ergodic Markov chains*, Electronical Communications in Probability **13** (2008), 85–98.

[5] Randal Douc, Éric Moulines, and Jeffrey Seth Rosenthal, *Quantitative bounds on convergence of time-inhomogeneous Markov chains*, The Annals of Applied Probability **14** (2004), no. 4, 1643–1665.

[6] Andrew Gelman, Gareth Owen Roberts, and Walter R. Gilks, *Efficient Metropolis jumping rules*, Bayesian Statistics **5** (1996), 599–607.

[7] Heikki Haario, Eero Saksman, and Johanna Tamminen, *An adaptive Metropolis algorithm*, Bernoulli **7** (2001), no. 2, 223–242.

[8] W. Keith Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika **57** (1970), no. 1, 97–109.

[9] Naresh Jain and Benton Jamison, *Contributions to Doeblin's theory of Markov processes*, Zeitschrift füer Wahrscheinlichkeitstheorie und Verwandte Gebiete **8** (1967), 19–40.

[10] Ajay Jasra and Chao Yang, *A regeneration proof of the central limit theorem for uniformly ergodic Markov chains*, Statistics and Probability Letters **78** (2008), 1649–1655.

[11] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall Nicholas Rosenbluth, Augusta Maria Harkanyi Teller, and Edward Teller, *Equation of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

[12] Sean Peter Meyn and Richard Lewis Tweedie, *Markov chains and stochastic stability*, Springer, New York, 2005.

[13] Christian P. Robert and George Casella, *Monte Carlo statistical methods*, Springer, New York, 2004.

[14] Gareth Owen Roberts, Andrew Gelman, and Walter R. Gilks, *Weak convergence and optimal scaling of random walk Metropolis algorithms*, Annals of Applied Probability **7** (1997), no. 1, 110–120.

[15] Gareth Owen Roberts and Jeffrey Seth Rosenthal, *General state space Markov chains and MCMC algorithms*, Probability Surveys **1** (2004), 20–71.

[16] ______, *Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms*, Journal of Applied Probability **44** (2007), 458–475.

[17] ______, *Examples of adaptive MCMC*, Unpublished Manuscript. Department of Mathematics, University of Toronto, Toronto, Canada, 2009.

[18] Jeffrey Seth Rosenthal, *Quantitative convergence rates of Markov chains: A simple account*, Electronical Communcations in Probability **7** (2002), 123–128.

[19] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0.

[20] John Wilder Tukey, *Exploratory data analysis*, Addison-Wesley, New York, 1977.

[21] Chao Yang, *Recurrent and ergodic properties of adaptive MCMC*, Unpublished Manuscript. Department of Mathematics, University of Toronto, Toronto, Canada, 2009.

A Recursive Formulas for the Mean and the Covariance

Lemma 8. *Let $k, n \in \mathbf{N}$ with $k < n$ and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbf{R}^d$. Then*

$$\overline{\mathbf{x}}_{nk} = \frac{n-1}{n}\overline{\mathbf{x}}_{(n-1)k} + \frac{1}{nk}\sum_{i=(n-1)k+1}^{nk} \mathbf{x}_i$$

and

$$\begin{aligned}\Sigma_{nk} &= \frac{(n-1)k-1}{nk-1}\Sigma_{(n-1)k} + \frac{1}{nk-1}\left(\sum_{i=(n-1)k+1}^{nk} \mathbf{x}_i\mathbf{x}_i' - nk\overline{\mathbf{x}}_{nk}\overline{\mathbf{x}}_{nk}' \right. \\ &\quad \left. +(n-1)k\overline{\mathbf{x}}_{(n-1)k}\overline{\mathbf{x}}_{(n-1)k}'\right).\end{aligned}$$

Proof. For the mean,

$$\begin{aligned}\overline{\mathbf{x}}_{nk} &= \frac{1}{nk}\sum_{i=1}^{nk}\mathbf{x}_i \\ &= \frac{1}{nk}\left(\sum_{i=1}^{(n-1)k}\mathbf{x}_i + \sum_{i=(n-1)k+1}^{nk}\mathbf{x}_i\right) \\ &= \frac{n-1}{n}\overline{\mathbf{x}}_{(n-1)k} + \frac{1}{nk}\sum_{i=(n-1)k+1}^{nk}\mathbf{x}_i\end{aligned}$$

and for the covariance,

$$\begin{aligned}\Sigma_{nk} &= \frac{1}{nk-1}\left(\sum_{i=1}^{nk}\mathbf{x}_i\mathbf{x}_i' - nk\overline{\mathbf{x}}_{nk}\overline{\mathbf{x}}_{nk}'\right) \\ &= \frac{1}{nk-1}\left(\sum_{i=1}^{(n-1)k}\mathbf{x}_i\mathbf{x}_i' + \sum_{i=(n-1)k+1}^{nk}\mathbf{x}_i\mathbf{x}_i' - (n-1)k\overline{\mathbf{x}}_{(n-1)k}\overline{\mathbf{x}}_{(n-1)k}' \right. \\ &\quad \left. -nk\overline{\mathbf{x}}_{nk}\overline{\mathbf{x}}_{nk}' + (n-1)k\overline{\mathbf{x}}_{(n-1)k}\overline{\mathbf{x}}_{(n-1)k}'\right)\end{aligned}$$

$$= \frac{(n-1)k-1}{nk-1}\Sigma_{(n-1)k} + \frac{1}{nk-1}\left(\sum_{i=(n-1)k+1}^{nk} \mathbf{x}_i\mathbf{x}_i' - nk\overline{\mathbf{x}}_{nk}\overline{\mathbf{x}}_{nk}' \right.$$
$$\left. +(n-1)k\overline{\mathbf{x}}_{(n-1)k}\overline{\mathbf{x}}_{(n-1)k}'\right)$$

□

Lemma 9. *Let $m, n \in \mathbf{N}$ with $m < n$ and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbf{R}^d$. Define*

$$\overline{\mathbf{x}}_{n,m} := \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \sum_{i=n-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n} \mathbf{x}_i$$

and

$$\Sigma_{n,m} := \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor}\left(\sum_{i=n-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n} \mathbf{x}_i\mathbf{x}_i' - \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1\right)\overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}_{n,m}' \right).$$

Then

(i) if $n = km + 1$ for some $k \in \mathbf{N}$,

$$\overline{\mathbf{x}}_{n,m} = \frac{n-1}{n-1+m}\overline{\mathbf{x}}_{n-1,m} + \frac{m}{n-1+m}\mathbf{x}_n$$

and otherwise

$$\overline{\mathbf{x}}_{n,m} = \overline{\mathbf{x}}_{n-1,m} + \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1}\left(\mathbf{x}_n - \mathbf{x}_{n-\left\lfloor \frac{n-1}{m} \right\rfloor - 1}\right).$$

(ii) If $n = km + 1$ for some $k \in \mathbf{N}$,

$$\begin{aligned}\Sigma_{n,m} &= \frac{n-1-m}{n-1}\Sigma_{n-1,m} + \frac{1}{n-1}\left(m\mathbf{x}_n\mathbf{x}_n' \right.\\ &\quad \left. + (n-1)\,\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}_{n-1,m}' - (n-1+m)\,\overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}_{n,m}'\right)\end{aligned}$$

and otherwise

$$\begin{aligned}\Sigma_{n,m} &= \Sigma_{n-1,m} + \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor}\Big(\mathbf{x}_n\mathbf{x}_n' - \mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}\mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}' \\ &\quad + \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1\right)\left(\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}_{n-1,m}' - \overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}_{n,m}'\right)\Big).\end{aligned}$$

Proof. (i) First assume that $n = km + 1$ for some $k \in \mathbf{N}$. Then $\frac{n-1}{m} \in \mathbf{N}$, and hence $\left\lfloor \frac{n-2}{m} \right\rfloor = \left\lfloor \frac{n-1}{m} \right\rfloor - 1 = \frac{n-1}{m} - 1$. Then

$$\mathbf{x}_{n-1}^{(m)} = \frac{1}{\left\lfloor \frac{n-2}{m} \right\rfloor + 1} \sum_{i=n-1-\left\lfloor \frac{n-2}{m} \right\rfloor}^{n-1} \mathbf{x}_i = \frac{1}{\frac{n-1}{m}} \sum_{i=n-\frac{n-1}{m}}^{n-1} \mathbf{x}_i.$$

Therefore,

$$\begin{aligned}
\mathbf{x}_n^{(m)} &= \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \sum_{i=n-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n} \mathbf{x}_i \\
&= \frac{1}{\frac{n-1}{m} + 1} \frac{n-1}{m} \cdot \left(\frac{1}{\frac{n-1}{m}} \sum_{i=n-\frac{n-1}{m}}^{n-1} \mathbf{x}_i \right) + \frac{1}{\frac{n-1}{m} + 1} \mathbf{x}_n \\
&= \frac{n-1}{n-1+m} \overline{\mathbf{x}}_{n-1}^{(m)} + \frac{m}{n-1+m} \mathbf{x}_n.
\end{aligned}$$

Now assume $n \neq km + 1$ for all $k \in \mathbf{N}$. Then $\left\lfloor \frac{n-2}{m} \right\rfloor = \left\lfloor \frac{n-1}{m} \right\rfloor$, so

$$\mathbf{x}_{n-1}^{(m)} = \frac{1}{\left\lfloor \frac{n-2}{m} \right\rfloor + 1} \sum_{i=n-1-\left\lfloor \frac{n-2}{m} \right\rfloor}^{n-1} \mathbf{x}_i = \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \sum_{i=n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n-1} \mathbf{x}_i$$

and

$$\begin{aligned}
\mathbf{x}_n^{(m)} &= \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \sum_{i=n-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n} \mathbf{x}_i \\
&= \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \left(-\mathbf{x}_{n-\left\lfloor \frac{n-1}{m} \right\rfloor - 1} + \sum_{i=n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n-1} \mathbf{x}_i + \mathbf{x}_n \right) \\
&= \overline{\mathbf{x}}_{n-1}^{(m)} + \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor + 1} \left(\mathbf{x}_n - \mathbf{x}_{n-\left\lfloor \frac{n-1}{m} \right\rfloor - 1} \right).
\end{aligned}$$

(ii) For the covariances, if $n = km + 1$ for some $k \in \mathbf{N}$,

$$\Sigma_{n-1,m} = \frac{1}{\left\lfloor \frac{n-2}{m} \right\rfloor} \left(\sum_{i=n-1-\left\lfloor \frac{n-2}{m} \right\rfloor}^{n-1} \mathbf{x}_i \mathbf{x}_i' \right.$$

$$-\left(\left\lfloor\frac{n-2}{m}\right\rfloor+1\right)\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}\Bigg)$$

$$= \frac{1}{\frac{n-1}{m}-1}\left(\sum_{i=n-\frac{n-1}{m}}^{n-1}\mathbf{x}_i\mathbf{x}'_i-\frac{n-1}{m}\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}\right)$$

and hence

$$\begin{aligned}
\Sigma_{n,m} &= \frac{1}{\left\lfloor\frac{n-1}{m}\right\rfloor}\left(\sum_{i=n-\left\lfloor\frac{n-1}{m}\right\rfloor}^{n}\mathbf{x}_i\mathbf{x}'_i-\left(\left\lfloor\frac{n-1}{m}\right\rfloor+1\right)\overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}'_{n,m}\right)\\
&= \frac{1}{\frac{n-1}{m}}\Bigg(\sum_{i=n-\frac{n-1}{m}}^{n-1}\mathbf{x}_i\mathbf{x}'_i-\frac{n-1}{m}\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}+\mathbf{x}_n\mathbf{x}'_n\\
&\quad +\frac{n-1}{m}\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}-\left(\frac{n-1}{m}+1\right)\overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}'_{n,m}\Bigg)\\
&= \frac{n-1-m}{n-1}\Sigma_{n-1,m}+\frac{1}{n-1}\big(m\mathbf{x}_n\mathbf{x}'_n\\
&\quad +(n-1)\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}-(n-1+m)\overline{\mathbf{x}}_{n,m}\overline{\mathbf{x}}'_{n,m}\big).
\end{aligned}$$

Otherwise,

$$\begin{aligned}
\Sigma_{n-1,m} &= \frac{1}{\left\lfloor\frac{n-2}{m}\right\rfloor}\Bigg(\sum_{i=n-1-\left\lfloor\frac{n-2}{m}\right\rfloor}^{n-1}\mathbf{x}_i\mathbf{x}'_i\\
&\quad -\left(\left\lfloor\frac{n-2}{m}\right\rfloor+1\right)\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}\Bigg)\\
&= \frac{1}{\left\lfloor\frac{n-1}{m}\right\rfloor}\Bigg(\sum_{i=n-1-\left\lfloor\frac{n-1}{m}\right\rfloor}^{n-1}\mathbf{x}_i\mathbf{x}'_i\\
&\quad -\left(\left\lfloor\frac{n-1}{m}\right\rfloor+1\right)\overline{\mathbf{x}}_{n-1,m}\overline{\mathbf{x}}'_{n-1,m}\Bigg),
\end{aligned}$$

so

$$\begin{aligned}
\Sigma_{n,m} &= \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor} \left(\sum_{i=n-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n} \mathbf{x}_i \mathbf{x}_i' - \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1 \right) \overline{\mathbf{x}}_{n,m} \overline{\mathbf{x}}_{n,m}' \right) \\
&= \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor} \left(\sum_{i=n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}^{n-1} \mathbf{x}_i \mathbf{x}_i' - \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1 \right) \overline{\mathbf{x}}_{n-1,m} \overline{\mathbf{x}}_{n-1,m}' \right. \\
&\quad - \mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor} \mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}' + \mathbf{x}_n \mathbf{x}_n' \\
&\quad - \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1 \right) \overline{\mathbf{x}}_{n,m} \overline{\mathbf{x}}_{n,m}' \\
&\quad \left. + \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1 \right) \overline{\mathbf{x}}_{n-1,m} \overline{\mathbf{x}}_{n-1,m}' \right) \\
&= \Sigma_{n-1,m} + \frac{1}{\left\lfloor \frac{n-1}{m} \right\rfloor} \left(\mathbf{x}_n \mathbf{x}_n' - \mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor} \mathbf{x}_{n-1-\left\lfloor \frac{n-1}{m} \right\rfloor}' \right. \\
&\quad \left. + \left(\left\lfloor \frac{n-1}{m} \right\rfloor + 1 \right) \left(\overline{\mathbf{x}}_{n-1,m} \overline{\mathbf{x}}_{n-1,m}' - \overline{\mathbf{x}}_{n,m} \overline{\mathbf{x}}_{n,m}' \right) \right).
\end{aligned}$$

□

B R Code for the Algorithms

B.1 Standard Algorithm

```
library(mvtnorm)                                                #load required packages

Y = read.table("data.txt",header=TRUE)                          #input data
N = dim(Y)[1]                                                   #sample size
nu = 4                                                          #hyperparameters
m = c(0,0,0)
v = c(5,5,5)
s = 2.266167
t = 3.266167

M = 10^6                                                        #number of iterations
B = M/2                                                         #length of burn-in
d = 4                                                           #dim of parameter space
np = 2                                                          #length of initial phase
beta = 0.05                                                     #beta from RR (in press)

g = function(X,log=FALSE) {                                     #(log of) ~density of g
   if (X[4] <= 0)
      if (log==TRUE)
         return( -Inf )
      else
         return( 0 )
   else
      E = -(1/2)*sum( ((X[1:3]-m)/v)^2 ) - t/X[4]
      S = sum(
              log( ( 1 + (1/nu) * (Y[,2] -X[1] -X[2]*Y[,1] -
                                    X[3]*Y[,1]^2)^2
                                   )
              )
          )
      res = E -(s+1+N/2) *log(X[4]) -(nu+1)/2 *S
      if (log==TRUE)
         return( res )
      else
         return( exp(res) )
}

X = matrix(0,M,4)                                               #for samples
numaccept = 0                                                   #for #accepted proposals
#X[1,] = rmvnorm(1, rep(0,4),diag(rep(1,4)) )                    #starting vector from
                                                                # overdispersered starting
                                                                # distribution
X[1,4] = 1#abs(X[1,4])                                             #4.component >0

lastC = diag(rep(1,4))                                          #for covariance matrix
lastm = X[1,]                                                   #for mean vector

for (i in 2:M) {
    if (i>np*d) {
       propcov = (1-beta)^2 * (2.38^2)/d * lastC +
                 beta^2 * diag(rep(0.01/d,d))
       Z = rmvnorm(1, X[i-1,], propcov)
    }
    else {
       Z = rmvnorm(1, X[i-1,], diag(rep(0.01/d,d)) )
    }
    logA = g(Z,log=TRUE) - g(X[i-1,],log=TRUE)
    logU = log( runif(1) )
    if (logU < logA) {
       X[i,] = Z;
       numaccept = numaccept+1
    }
    else {
       X[i,] = X[i-1,]
    }
    prelastm = lastm
    lastm = (i-1)/i *lastm + 1/i * X[i,]
    lastC=(i-1)/i*lastC + 1/i *
          ( i* prelastm %*% t(prelastm) -
            (i+1)*lastm %*% t(lastm) +
            X[i,] %*% t(X[i,]) )
}

est = colMeans(X[(B+1):M,])

ACF = c()                                                        #for autocorrelations
k = 1                                                            #count lag
mincorr = rep(1,4)                                               #minimum lag
while (any(mincorr>0.05)) {
   lagcor  = cor( X[(B+1):(M-k),], X[(B+1+k):M,] )               #calculate new lag
   ACF = rbind( ACF, diag(lagcor) )
   mincorr = pmin( mincorr, abs(ACF[k,]) )                       #determine minimum corr
   k = k+1
}

varfact=rep(0,4)                                                 #for varfact
for (i in 1:4) {
   bigacfs = abs(ACF[,i])>0.05
   firstbigacfs = ACF[1:( which.min(bigacfs)-1 ),i]              #acfs up to first <0.05
   varfact[i] = 1+2*sum(firstbigacfs)                            #compute varfact
}

varest = diag( var(X[(B+1):M,]) )                                #estimate sample ariance
seest = sqrt( varest * varfact / (M-B) )                         #estimate standard error
CIl = est+qnorm(0.025)*seest                                     #estimate CI for estimate
CIu = est+qnorm(0.975)*seest

jumpdist = rowSums((X[2:M,]-X[1:(M-1),])^2)                      #jumping distances
avjumpdist = mean(jumpdist[jumpdist!=0])                         #average jumpind distance

cat("proportion of accepted proposals:", numaccept/M, "\n",
    "estimates for alpha, beta, gamma, and sigma^2:", est, "\n",
    "corresponding standard errors:", seest, "\n",
    "integrated autocorrelation times:", varfact, "\n",
    "average jumping distance:", avjumpdist, "\n",
    "95% confidence intervals:","\n")
paste(CIl,CIu)
```

B.2 Algorithm Updating Every k_0 Iterations

```
library(mvtnorm)                                                 #load required packages

Y = read.table("data.txt",header=TRUE)                           #input data
N = dim(Y)[1]                                                    #sample size
nu = 4                                                           #hyperparameters
m = c(0,0,0)
v = c(5,5,5)
s = 2.266167
t = 3.266167

M = 10^6                                                         #number of iterations
B = M/2                                                          #length of burn-in
l = 2                                                            #update cov every l runs
d = 4                                                            #dim of parameter space
np = 2                                                           #length of initial phase
beta = 0.05                                                      #beta from RR (in press)

g = function(X,log=FALSE) {                                      #(log of) ~density of g
   if (X[4] <= 0)
      if (log==TRUE)
         return( -Inf )
      else
         return( 0 )
   else
      E = -(1/2)*sum( ((X[1:3]-m)/v)^2 ) - t/X[4]
      S = sum(
              log( ( 1 + (1/nu) * (Y[,2] -X[1] -X[2]*Y[,1] -
                                    X[3]*Y[,1]^2)^2
                                   )
              )
          )
      res = E -(s+1+N/2) *log(X[4]) -(nu+1)/2 *S
      if (log==TRUE)
         return( res )
      else
         return( exp(res) )
}

X = matrix(0,M,4)                                                #for samples
numaccept = 0                                                    #for #accepted proposals
#X[1,] = rmvnorm(1, rep(0,4),diag(rep(1,4)) )                     #starting vector from
                                                                 # overdispersered starting
                                                                 # distribution
X[1,4] = 1#abs(X[1,4])                                              #4.component >0

lastC = diag(rep(1,4))                                           #for covariance matrix
lastm = X[1,]                                                    #for mean vector

for (i in 2:l) {
    if (i>np*d) {
       propcov = (1-beta)^2 * (2.38^2)/d * lastC +
                 beta^2 * diag(rep(0.01/d,d))
       Z = rmvnorm(1, X[i-1,], propcov)
    }    else {
       Z = rmvnorm(1, X[i-1,], diag(rep(0.01/d,d)) )
    }
    logA = g(Z,log=TRUE) - g(X[i-1,],log=TRUE)
    logU = log( runif(1) )
    if (logU < logA) {
       X[i,] = Z;
       numaccept = numaccept+1
    }    else {
       X[i,] = X[i-1,]
    }
    if (i%%l==0) {
       prelastm = lastm
       lastm = colMeans(X[1:l,])
       lastC = cov(X[1:l,])
    }
}


for (i in (l+1):M) {
    if (M>np*d) {
       propcov = (1-beta)^2 * (2.38^2)/d * lastC +
                  beta^2 * diag(rep(0.01/d,d))
       Z = rmvnorm(1, X[i-1,], propcov)
    }    else {
       Z = rmvnorm(1, X[i-1,], diag(rep(0.01/d,d)) )
    }
    logA = g(Z,log=TRUE) - g(X[i-1,],log=TRUE)
    logU = log( runif(1) )
    if (logU < logA) {
       X[i,] = Z;
       numaccept = numaccept+1
    }    else {
       X[i,] = X[i-1,]
    }
    if (i%%l==0) {
       prelastm = lastm
       n=i/l
       lastm = ((n-1) *lastm +  colMeans(X[(i-l+1):i,]) ) /n
       sumsq = X[i-l+1,] %*% t(X[i-l+1,])
       for (j in (i-l+2):i ) {
          sumsq = sumsq + X[j,] %*% t(X[j,])
       }
       lastC=( (i-l-1)*lastC + sumsq - i*lastm %*% t(lastm) +
                (n-1)*l * prelastm %*% t(prelastm) ) /(i-1)
    }
}

est = colMeans(X[(B+1):M,])

ACF = c()                                                      #for autocorrelations
k = 1                                                          #count lag
mincorr = rep(1,4)                                             #minimum lag
while (any(mincorr>0.05)) {
   lagcor  = cor( X[(B+1):(M-k),], X[(B+1+k):M,] )             #calculate new lag
   ACF = rbind( ACF, diag(lagcor) )
   mincorr = pmin( mincorr, abs(ACF[k,]) )                     #determine minimum corr
   k = k+1
}
```

```
varfact=rep(0,4)                                              #for varfact
for (i in 1:4) {
   bigacfs = abs(ACF[,i])>0.05
   firstbigacfs = ACF[1:( which.min(bigacfs)-1 ),i]           #acfs up to first <0.05
   varfact[i] = 1+2*sum(firstbigacfs)                         #compute varfact
}

varest = diag( var(X[(B+1):M,]) )                             #estimate sample ariance
seest = sqrt( varest * varfact / (M-B) )                      #estimate standard error
CIl = est+qnorm(0.025)*seest                                  #estimate CI for estimate
CIu = est+qnorm(0.975)*seest

jumpdist = rowSums((X[2:M,]-X[1:(M-1),])^2)                   #jumping distances
avjumpdist = mean(jumpdist[jumpdist!=0])                      #average jumpind distance

cat("proportion of accepted proposals:", numaccept/M, "\n",
    "estimates for alpha, beta, gamma, and sigma^2:", est, "\n",
    "corresponding standard errors:", seest, "\n",
    "integrated autocorrelation times:", varfact, "\n",
    "average jumping distance:", avjumpdist, "\n",
    "95% confidence intervals:","\n")
paste(CIl,CIu)
```

B.3 Algorithm Using the Last $1/k_0$ of the Iterations

```
library(mvtnorm)                                              #load required packages

Y = read.table("data.txt",header=TRUE)                        #input data
N = dim(Y)[1]                                                 #sample size
nu = 4                                                        #hyperparameters
m = c(0,0,0)
v = c(5,5,5)
s = 2.266167
t = 3.266167

M = 10^6                                                      #number of iterations
B = M/2                                                       #length of burn-in
d = 4                                                         #dim of parameter space
np = 2                                                        #length of initial phase
k = 2                                                         #use last 1/k iterations
                                                              # for computations
beta = 0.05                                                   #beta from RR (in press)

g = function(X,log=FALSE) {                                   #(log of) ~density of g
   if (X[4] <= 0)
      if (log==TRUE)
         return( -Inf )
      else
         return( 0 )
   else
      E = -(1/2)*sum( ((X[1:3]-m)/v)^2 ) - t/X[4]
      S = sum(
              log( ( 1 + (1/nu) * (Y[,2] -X[1] -X[2]*Y[,1] -
                                   X[3]*Y[,1]^2)^2
                                  )
              )
          )
      res = E -(s+1+N/2) *log(X[4]) -(nu+1)/2 *S
      if (log==TRUE)
         return( res )
      else
         return( exp(res) )
}

X = matrix(0,M,4)                                                #for samples
numaccept = 0                                                    #for #accepted proposals
#X[1,] = rmvnorm(1, rep(0,4),diag(rep(1,4)) )                    #starting vector from
                                                                 # overdispersered starting
                                                                 # distribution
X[1,4] = 1#abs(X[1,4])                                              #4.component >0

lastC = diag(rep(1,4))                                           #for covariance matrix
lastm = X[1,]                                                    #for mean vector

for (i in 2:k) {
   propcov = (1-beta)^2 * (2.38^2)/d * lastC + beta^2 * diag(rep(0.01/d,d))
   if (i>np*d) {
      Z = rmvnorm(1, X[i-1,], propcov)
   }  else {
      Z = rmvnorm(1, X[i-1,], diag(rep(0.01/d,d)) )
   }
   logA = g(Z,log=TRUE) - g(X[i-1,],log=TRUE)
   logU = log( runif(1) )
   if (logU < logA) {
      X[i,] = Z
      numaccept = numaccept+1
   }  else {
      X[i,] = X[i-1,]
   }
   prelastm = lastm
   l=floor((i-1)/k)
   lastm=lastm+(X[i,]-X[i-l-1,])/(l+1)
}

for (i in (k+1):M) {
   propcov = (1-beta)^2 * (2.38^2)/d * lastC + beta^2 * diag(rep(0.01/d,d))
   if (i>np*d) {
      Z = rmvnorm(1, X[i-1,], propcov)
   }  else {
      Z = rmvnorm(1, X[i-1,], diag(rep(0.01/d,d)) )
   }
   logA = g(Z,log=TRUE) - g(X[i-1,],log=TRUE)
   logU = log( runif(1) )
   if (logU < logA) {
      X[i,] = Z
      numaccept = numaccept+1
   }  else {
      X[i,] = X[i-1,]
   }
   prelastm = lastm

   if (i%%k==1) {
         lastm=((i-1)*lastm+k*X[i,])/(i-1+k)
         lastC=( (i-1-k)*lastC+k*X[i,] %*% t(X[i,]) + (i-1)*
                   prelastm %*% t(prelastm) - (i-1+k)*lastm %*% t(lastm) )/
                   (i-1)
   }
   else {
      l=floor((i-1)/k)
      lastm=lastm+(X[i,]-X[i-l-1,])/(l+1)
      lastC=lastC+( X[i,] %*% t(X[i,]) - X[i-1-l,] %*% t(X[i-1-l,]) +
                    (l+1)*( prelastm %*% t(prelastm) - lastm %*% t(lastm) )
                   )/l
   }
}

est = colMeans(X[(B+1):M,])

ACF = c()                                                       #for autocorrelations
r = 1                                                           #count lag
mincorr = rep(1,4)                                              #minimum lag
while (any(mincorr>0.05)) {
   lagcor  = cor( X[(B+1):(M-r),], X[(B+1+r):M,] )              #calculate new lag
   ACF = rbind( ACF, diag(lagcor) )
   mincorr = pmin( mincorr, abs(ACF[r,]) )                      #determine minimum corr
   r = r+1
}

varfact=rep(0,4)                                                #for varfact
for (i in 1:4) {
   bigacfs = abs(ACF[,i])>0.05
   firstbigacfs = ACF[1:( which.min(bigacfs)-1 ),i]             #acfs up to first <0.05
   varfact[i] = 1+2*sum(firstbigacfs)                           #compute varfact
}

varest = diag( var(X[(B+1):M,]) )                               #estimate sample ariance
seest = sqrt( varest * varfact / (M-B) )                        #estimate standard error
CIl = est+qnorm(0.025)*seest                                    #estimate CI for estimate
CIu = est+qnorm(0.975)*seest

jumpdist = rowSums((X[2:M,]-X[1:(M-1),])^2)                     #jumping distances
avjumpdist = mean(jumpdist[jumpdist!=0])                        #average jumpind distance

cat("proportion of accepted proposals:", numaccept/M, "\n",
    "estimates for alpha, beta, gamma, and sigma^2:", est, "\n",
    "corresponding standard errors:", seest, "\n",
    "integrated autocorrelation times:", varfact, "\n",
    "average jumping distance:", avjumpdist, "\n",
    "95% confidence intervals:","\n")
paste(CIl,CIu)
```

C Plots for Simulations

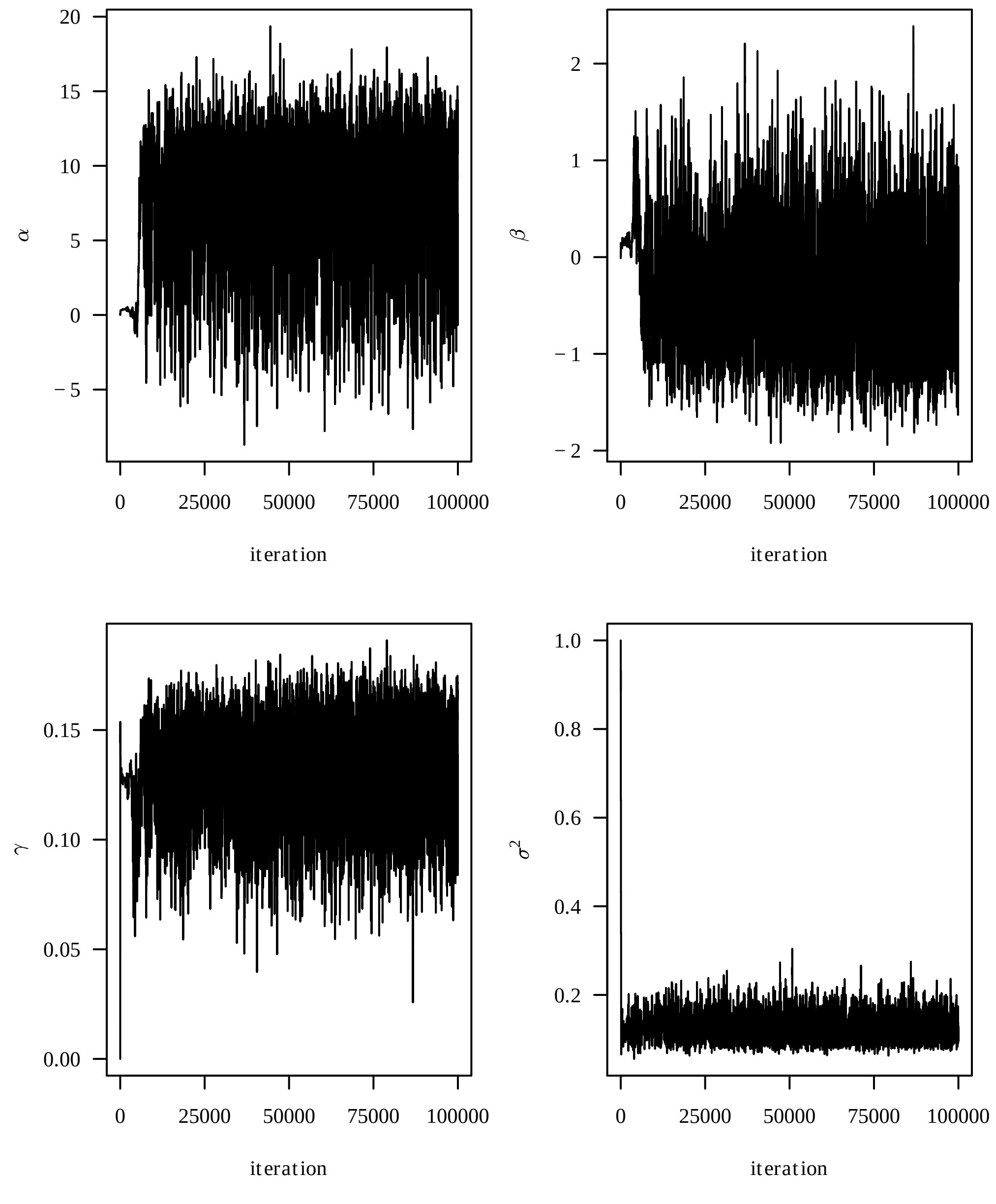

Figure 1: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the standard algorithm with initial phase of length 8.

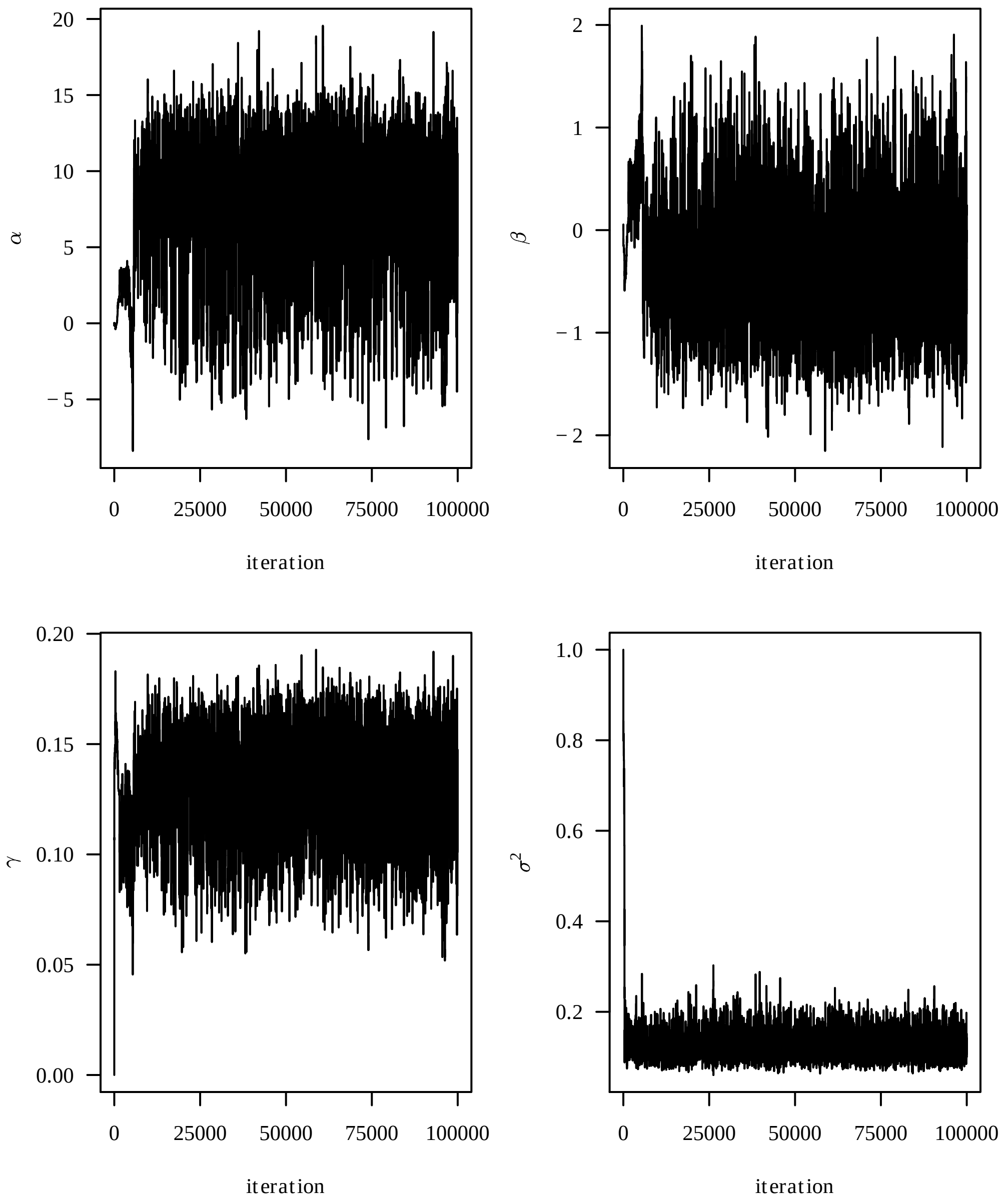

Figure 2: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal each 2^{nd} step which has an initial phase of length 8.

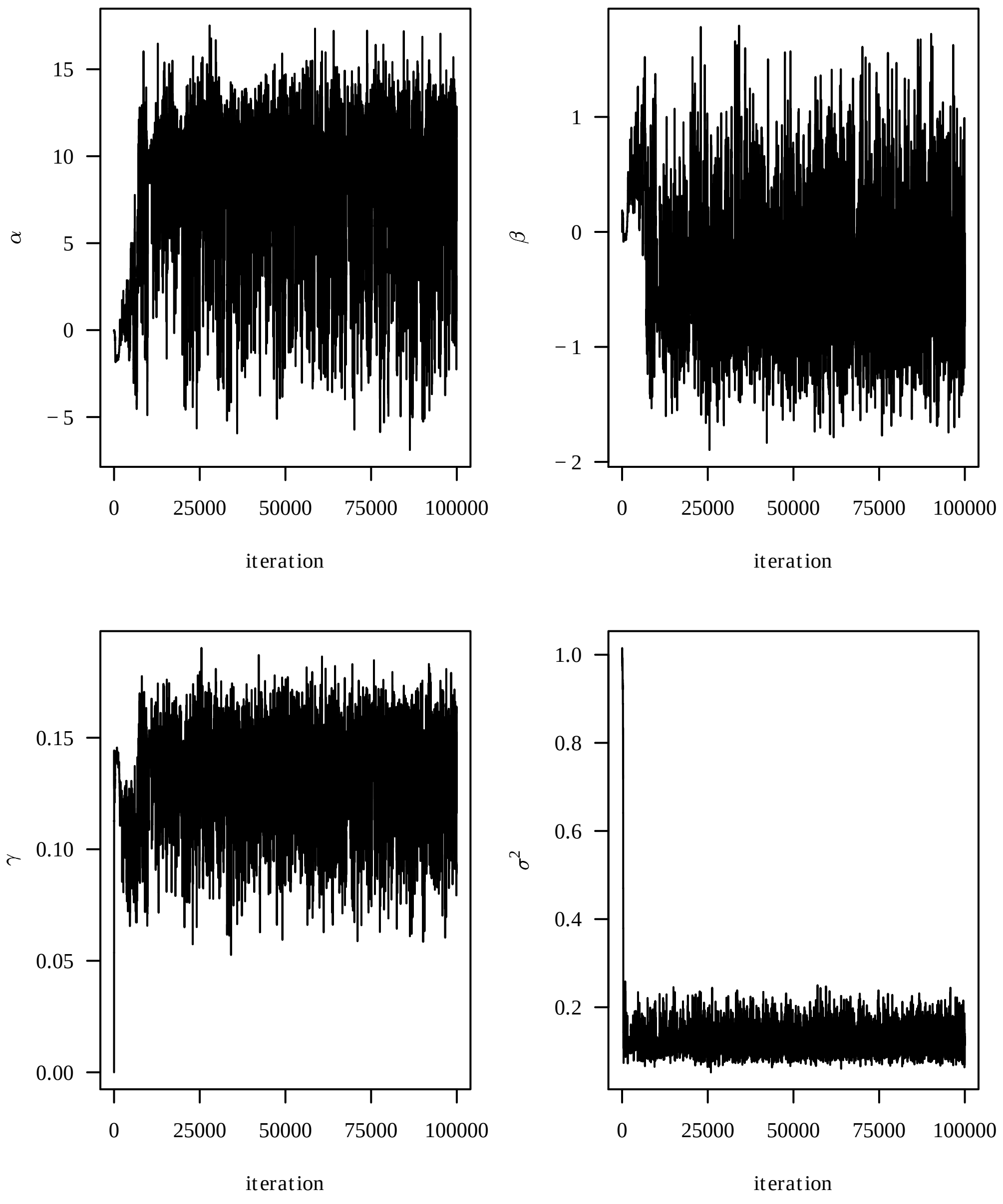

Figure 3: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal each 10th step which has an initial phase of length 8.

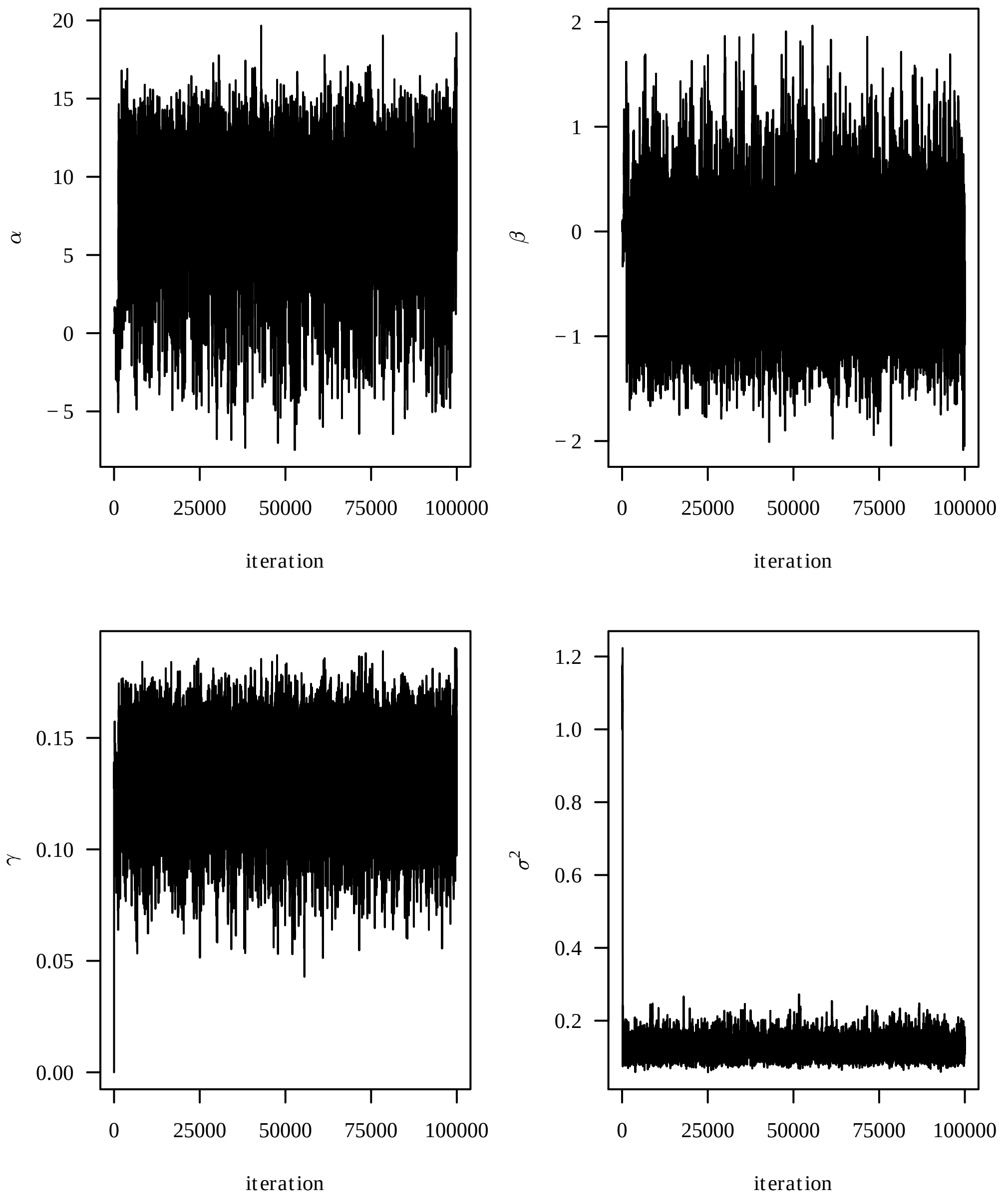

Figure 4: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal based on the last half of the iteration which has initial phase of length 8.

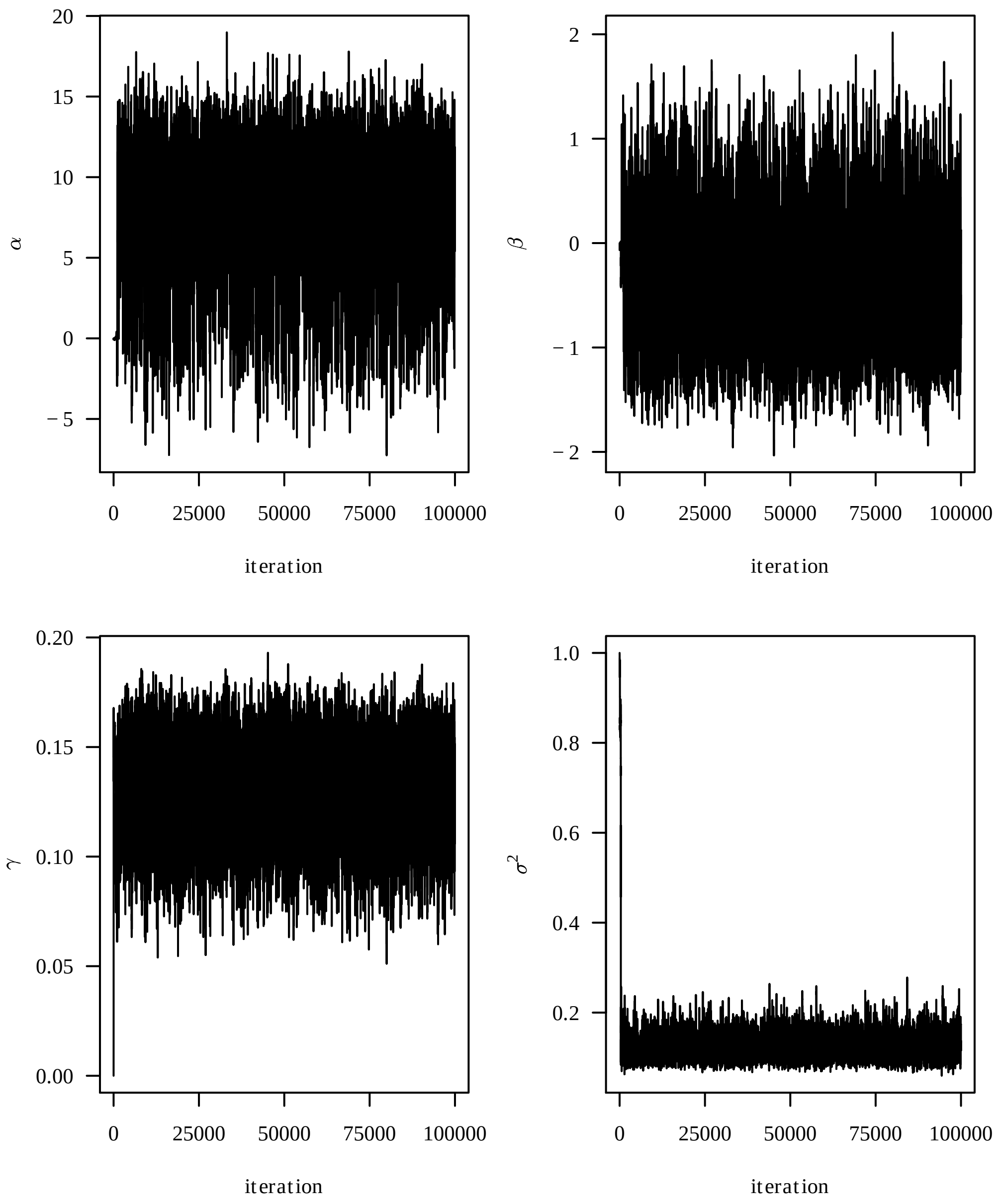

Figure 5: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal based on the last tenth of the iteration which has initial phase of length 8.

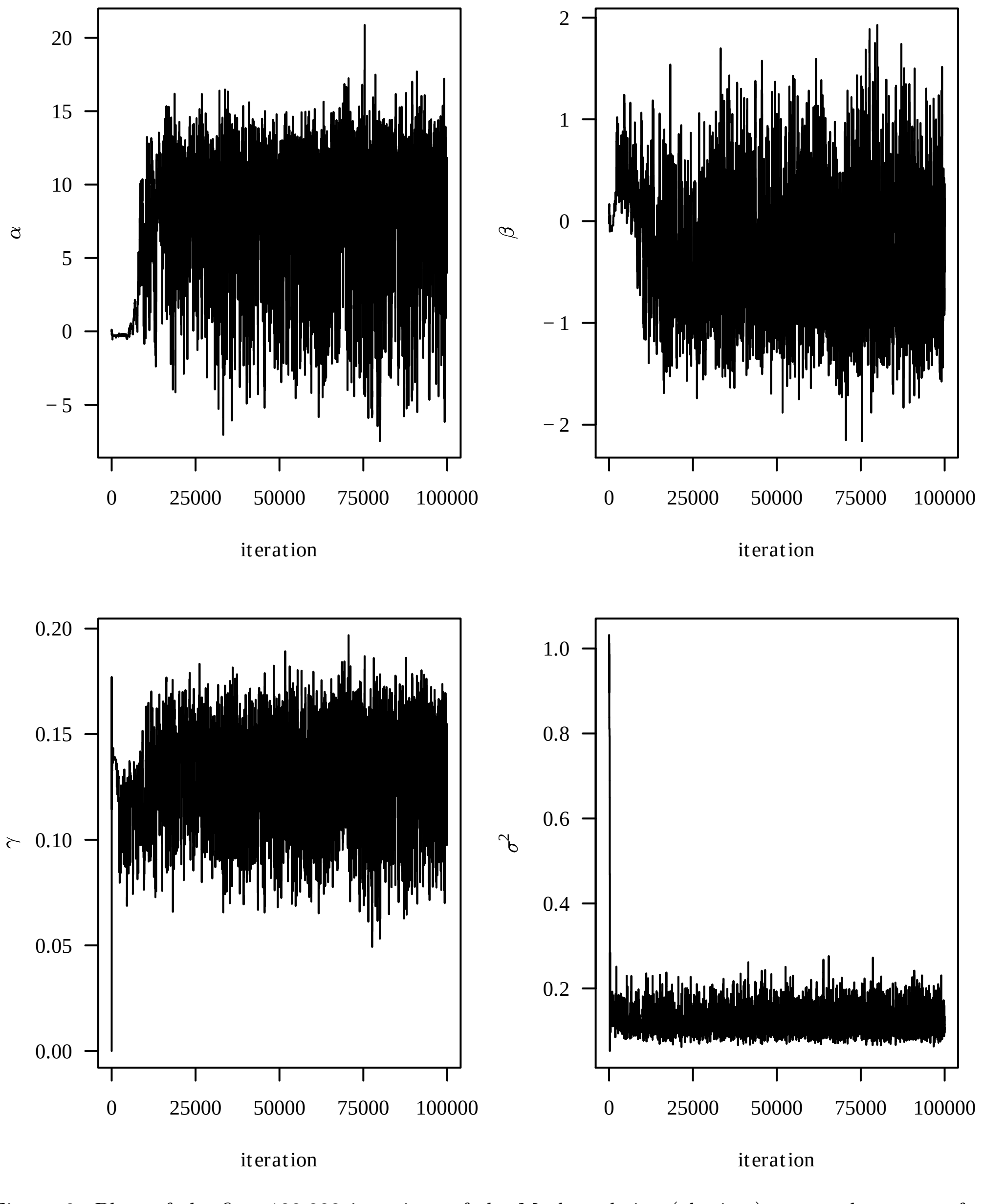

Figure 6: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the standard simulation with initial phase of length 200.

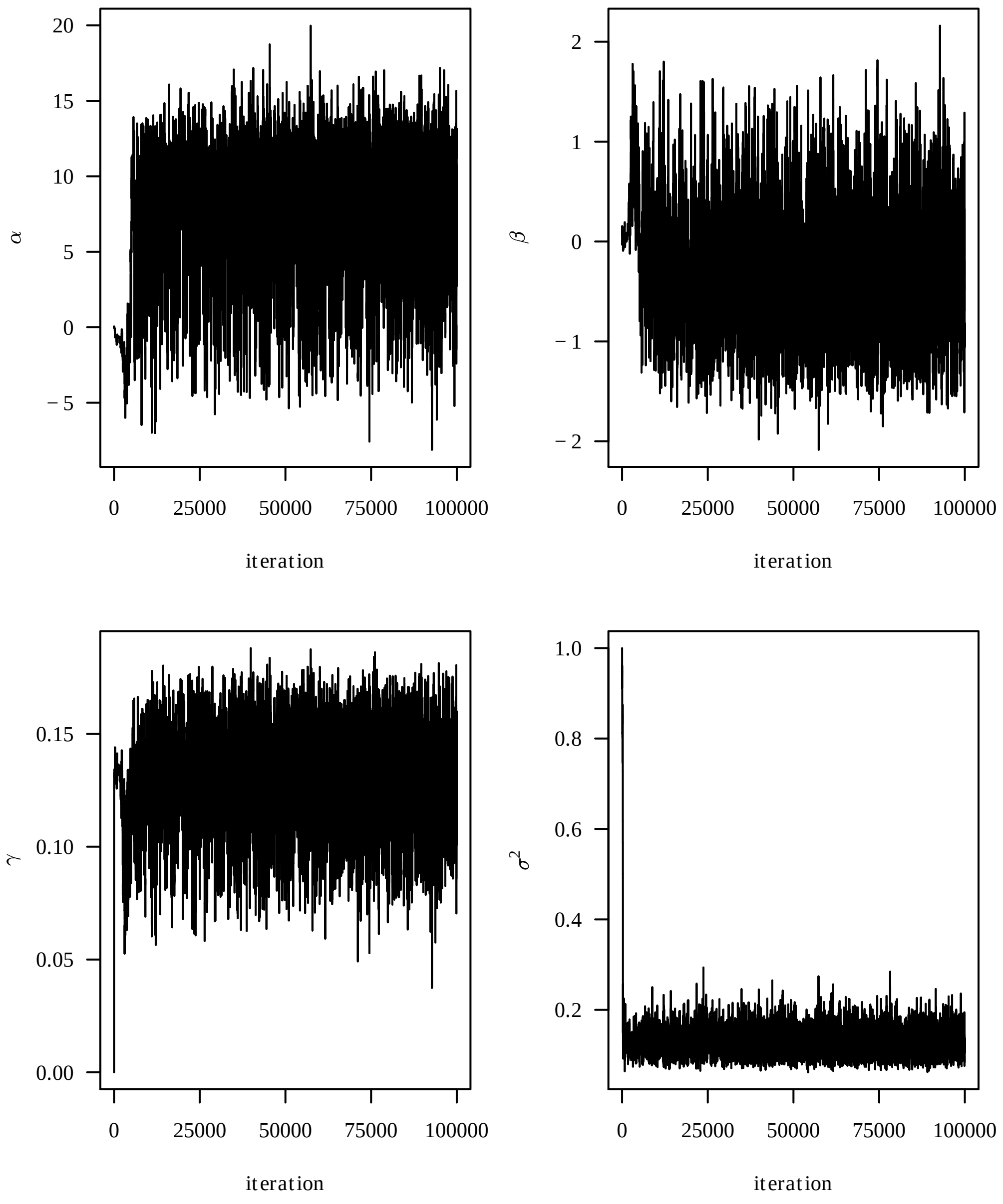

Figure 7: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal each 2^{nd} step which has an initial phase of length 200.

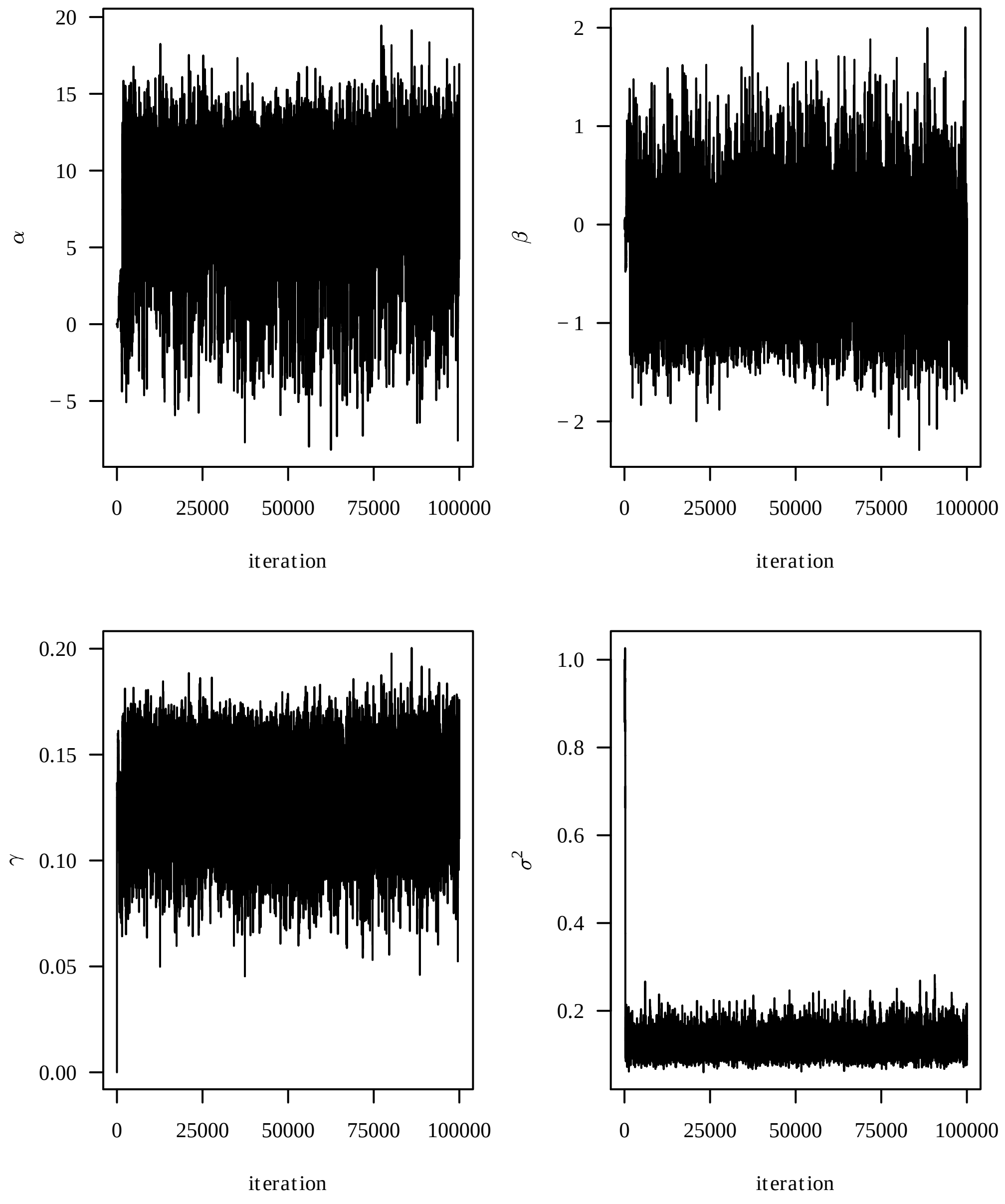

Figure 8: Plots of the first 100,000 iterations of the Markov chains (abscissa) versus the states of the Markov chain for each of the components (ordinate) for the algorithm updating the proposal based on the last half of the iteration which has initial phase of length 200.