IMPROVING MARKOV CHAIN MONTE CARLO ESTIMATORS USING OVERRELAXATION AND COUPLING TECHNIQUES

by

Ruxandra L. Pinto

A thesis submitted in conformity with the requirements for the
Degree of Doctor of Philosophy
Graduate Department of Statistics
University of Toronto

Improving Markov Chain Monte Carlo Estimators Using Overrelaxation and Coupling Techniques

Ruxandra L. Pinto
Department of Statistics, University of Toronto
Ph.D. Thesis, 2002

Abstract

Markov chain Monte Carlo algorithms have been used to explore complicated distributions in physics, chemistry, and most recently Bayesian and classical statistics. The computation time needed to obtain estimates for the parameters of the target distribution can sometimes be extremely large.

In this thesis, we propose three Markov chain Monte Carlo algorithms that reduce the computational effort for obtaining estimates at a given precision. The first two extend the scope of overrelaxation methods. The first algorithm, random sequence overrelaxation, suppresses random walks for Gaussian distributions with highly negatively-correlated components. The second algorithm, Jacobian overrelaxation, is applicable to non-Gaussian distributions for which the components are highly positively-correlated. The last method is based on coupling two chains, one sampling from the target distribution and one from an approximating distribution. By exploiting the correlation between these two chains we can obtain more precise estimates for a given computational effort.

To demonstrate empirically the performance of these algorithms, we perform a simulation study for each and we compare their performance with the performance of an appropriate existing algorithm. These simulations show that the proposed algorithms improve the efficiency of the Markov chain Monte Carlo estimators.

Acknowledgements

I would like to thank my supervisor, Radford Neal for his guidance, patience and encouragement throughout this undertaking. His enthusiasm and confidence were an inspiration for me.

My thanks also go to Jeff Rosenthal, Jerry Brunner, Mike Evans and Radu Craiu who helped me in successfully bringing this project to completion.

I am grateful to Laura Kerr, Andrea Carter, Sylvia Williams and Dermot Whelan for the endless help with all administrative and computing problems.

The inspiration provided by my mother's ambition, the patience and understanding of my husband, Austin, and the support from all my friends have all been instrumental in getting this task accomplished.

For my mother, Victoria

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Markov chain Monte Carlo (MCMC) algorithms were first applied in statistical physics and were later used in spatial statistics and image restoration. For the past few years, MCMC methods have been extensively used in Bayesian inference and in classical statistics as well.

Gibbs sampling and simple forms of the Metropolis algorithm tend to produce a chain that moves in random walk fashion. Therefore the chain needs n^2 steps to move to a point n steps away. For Bayesian inference problems of high dimensions, for which the components are usually highly-dependent, the distance the chain moves in one step is very small, so it is important to reduce the number of steps the chain needs to move to a state n steps away. Random walks could be suppressed either by hybrid Monte Carlo, introduced by Duane, Kennedy, Pendleton and Roweth (1987), or through the overrelaxation technique presented by Adler (1981) in a physics context. Adler's overrelaxation is applicable to distributions whose full-conditional distributions are Gaussian.

Standard overrelaxation suppresses random walks for Gaussian distributions with highly positively-correlated components, but fails to do so for highly negatively-correlated components. Standard overrelaxation applied to Gaussian distribution with highly negatively-correlated components with α, the overrelaxation parameter, set close to -1 produces a chain that has a circular movement. Because the chain moves a small amount in one step,and moreover moves nearly in a circle, the chain explores the state space very slowly and inefficiently. The algorithm we introduce, solves this problem by updating components in a random order by a random sequence that is repeated several times, causing the chain to move in a consistent direction. Different random sequences cause the chain to move in different directions, thereby allowing the chain to efficiently explore the state space. The algorithm is presented in Chapter 3, along with a simulation study for Gaussian distributions of different dimensions and covariance matrices with different correlation magnitudes.

Extensions of standard overrelaxation, applicable to a wider class of distributions, were investigated by Brown and Woch (1987), Green and Han (1992), and Fodor and Jansen (1994). All these overrelaxation type methods have a common approach: they propose a new state based on the overrelaxation type update and the new state is accepted or rejected by criteria similar to the one used in Metropolis algorithm. Even a moderate rejection rate might prevent the chain from moving along the length of the distribution for a substantial period of time. Neal (1998) and Neal (2003) presents two solutions to this problem, one called ordered overrelaxation and the other overrelaxed slice sampling. For the first method the acceptance/rejection step is not present and for the latter method the rejection rate can be made very small. Therefore both methods allow the chain to move for a long time in the same direction. These methods might pose computational difficulties; therefore other overrelaxation type methods can still be developed.

In Chapter 4 we propose an algorithm, that we call *Jacobian overrelaxation*, that suppresses random walks for distributions with highly positively-correlated components even for non-Gaussian distributions. This is achieved by generating S states using standard overrelaxation updates, without adding Gaussian random noise, and then choosing the next state according to the density of each state multiplied by a Jacobian factor. The method requires two tuning parameters, a range for the overrelaxation parameter, α, and the number of states to chose the next state from, $S + 1$. We derive a relationship that requires the choice of only one parameter, say S, the other parameter α being derived from the relationship. A detailed simulation study for a logistic regression problem is conducted to investigate the performance of Jacobian overrelaxation versus the Metropolis algorithm, as well as to confirm the validity of the formula developed when the distributions are not Gaussian.

In Chapter 5 we show how large improvements in the accuracy of an estimator can be obtained by coupling a Markov chain with an approximating chain. The approximating chain can sample from a Gaussian approximation to the distribution of interest or from any other distribution that is easy to sample from, as long as high correlations between the chains are produced. We propose an efficient estimator that takes advantage of this correlation between the chains, and demonstrate it on the pumps data from Gelfand and Smith (1990).

In Chapter 6 we summarize the ideas of the thesis and present possible extensions of those ideas.

We begin in the next chapter with background material about basic Markov chain Monte Carlo algorithms, as well as an overview of the present overrelaxation techniques.

Chapter 2

Markov Chain Monte Carlo

In this chapter we present background material necessary to understand this thesis. In Section 2.1 we present some of the basic properties of Markov chains together with one way of evaluating different Markov chains based on autocorrelation time. In Section 2.2 we review two well-established algorithms: Metropolis algorithm and Gibbs sampling. Random walks behaviour and the overrelaxation techniques that are suppressing them are outlined in Section 2.3.

2.1 Markov Chain Monte Carlo Integration

Bayesian inference problems require calculation of the expectations of various functions of the model parameters with respect to their posterior distribution. If the posterior density, $f(y)$, is easy to sample from, the expectation of $a(y)$ with respect to f can be estimated using Monte Carlo integration by

$$E(a(y)) \approx \overline{a}_n = \frac{1}{n}\sum_{i=1}^{n} a(y_i) = \frac{1}{n}\sum_{i=1}^{n} a_i, \tag{2.1}$$

where $y_1, \ldots, y_n$ is a sample of n independent points drawn from f and $a_i = a(y_i), \quad i = 1, \ldots, n$.

Drawing samples independently from the posterior distribution is not feasible in most Bayesian inference problems because the posterior distribution, f, is usually too complicated. We can instead generate dependent points from f using a Markov chain $y_1, \ldots, y_n$ that has f as its *stationary distribution*. A Markov chain y is a discrete time stochastic process with the property that the distribution of y_t given all the previous states $y_1, \ldots, y_{t-1}$ depends only on y_{t-1}. The starting point y_1 is drawn from an *initial distribution* and each subsequent state y_t is generated using a *transition kernel* that we will denote by $P(y_t|y_{t-1})$. The distribution f is *stationary* or *invariant* with respect to the transition probabilities if once a state y_t has reached the distribution, f, $y_{t'}$ will have the same distribution f, for all $t' > t$. This condition can be written as:

$$\int P(y'|y)f(y)dy \quad = \quad f(y') \tag{2.2}$$

In practice, it is often easier to verify the stronger condition known as *detailed balance*:

$$P(y'|y)f(y) \quad = \quad P(y|y')f(y') \quad \text{for all } y \text{ and } y' \tag{2.3}$$

One can easily show that if detailed balance holds for a distribution f with respect to P, then f is stationary for P. By integrating both sides of equation (2.3) with respect to y, we obtain:

$$\begin{aligned} \int P(y'|y)f(y)dy &= \int P(y|y')f(y')dy && (2.4) \\ &= f(y') \int P(y|y')dy && (2.5) \\ &= f(y') && (2.6) \end{aligned}$$

A chain that satisfies the detailed balance condition is called *reversible*.

An *ergodic* Markov chain is a Markov chain that converges from any initial state to a unique invariant distribution called its *equilibrium distribution.*

Estimating the expectation with respect to a distribution f requires a Markov chain whose stationary distribution is f, that converges to this equilibrium distribution quickly, and that produces states that are not highly dependent.

Although the states of the Markov chain are dependent, estimator (2.1) will still be nearly unbiased if we discard sufficient initial burn-in states that are not from f. Because the states are dependent, we may require a larger number of states to produce an estimate with the same precision as the estimate based on independent states. If the states are independent with $Var(a_i) = \sigma^2,\ i = 1, \ldots, n$ then

$$Var(\overline{a}_n) = \frac{\sigma^2}{n} \tag{2.7}$$

For Markov chain states, the variance of estimator (2.1) is

$$Var(\overline{a}_n) \;=\; \frac{Var(a_1 + \ldots + a_n)}{n^2} \;=\; \frac{\sum_{i,j=1}^{n} Cov(a_i, a_j)}{n^2} \tag{2.8}$$

For any k, because the chain is stationary we can define

$$c(k) \;=\; Cov(a_i, a_{i+k}) \;=\; Cov(a_{i+k}, a_i) \;= c(-k) \tag{2.9}$$

and $c(0) \;=\; Cov(a_i, a_i) = Var(a_i)$. Therefore

$$\begin{aligned} Var(\overline{a}_n) \;&=\; \frac{1}{n^2} \sum_{k=-(n-1)}^{n-1} (n - |k|)c(k) & (2.10) \\ &=\; \frac{1}{n} \sum_{k=-(n-1)}^{n-1} \left(1 - \frac{|k|}{n}\right) c(k) & (2.11) \end{aligned}$$

For large n, the variance from equation (2.11) can be written as:

$$Var(\overline{a}_n) \;=\; \frac{1}{n} \left(\sigma^2 + 2 \sum_{k=1}^{\infty} c(k)\right) \tag{2.12}$$

$$= \frac{\sigma^2}{n}\left(1 + 2\sum_{k=1}^{\infty}\rho(k)\right) \tag{2.13}$$

$$= \frac{\sigma^2}{n/\tau} \tag{2.14}$$

where $\rho(k) = c(k)/\sigma^2$ is the autocorrelation function and $\tau = 1 + 2\sum_{k=1}^{\infty}\rho(k)$ is the autocorrelation time.

We will estimate τ by summing over all positive and negative autocorrelations up to the lag for which the autocorrelations seem to be nearly zero. The autocorrelations are positive for most chains, and therefore the variance of the estimator $\overline{a}_n$ when the states are dependent will be higher than when the states are independent. The autocorrelations could be negative as well; in this case the estimator based on dependent states will be more precise than if the states were independent. The unbiased estimate for σ^2 is:

$$\widehat{\sigma}_n^2 = \frac{\sum_{i=1}^{n}(a_i - \overline{a}_n)^2}{n - \tau} \tag{2.15}$$

Indeed, one can prove that

$$E\left((a_i - \overline{a}_n)^2\right) = Var(a_i) - \frac{2}{n}\sum_{j=1}^{n}Cov(a_i, a_j) + Var(\overline{a}_n) \tag{2.16}$$

Therefore, by summing over i in the above equation, we obtain

$$E\left(\sum_{i=1}^{n}(a_i - \overline{a}_n)^2\right) = \sum_{i=1}^{n}\sigma^2 - \frac{2}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}Cov(a_i, a_j) + nVar(\overline{a}_n) \tag{2.17}$$

$$= n\sigma^2 - \frac{2}{n}n^2 Var(\overline{a}_n) + nVar(\overline{a}_n) \tag{2.18}$$

$$= n\sigma^2 - \sigma^2\tau \tag{2.19}$$

$$= \sigma^2(n - \tau) \tag{2.20}$$

where in the equation (2.18) we used that

$$\sum_{i=1}^{n}\sum_{j=1}^{n}Cov(a_i, a_j) = n^2 Var(\overline{a}_n) \tag{2.21}$$

and

$$Var(\overline{a}_n) = \frac{\sigma^2}{n/\tau} \tag{2.22}$$

The estimate for the variance of $\overline{a}_n$ is:

$$\widehat{Var}(\overline{a}_n) = \frac{\widehat{\sigma}_n^2}{n/\widehat{\tau}} = \frac{\sum_{i=1}^n (a_i - \overline{a}_n)^2}{n - \widehat{\tau}} \frac{\widehat{\tau}}{n} \tag{2.23}$$

2.2 Basic Markov Chain Monte Carlo Algorithms

This section presents a few of the existing algorithms that produce Markov chains converging to an equilibrium distribution, whose density will be denoted by f.

2.2.1 The Metropolis-Hastings Algorithm

The Metropolis algorithm was introduced in a statistical physics paper by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953) and generalized by Hastings (1970).

Transitions for the Metropolis-Hastings algorithm are defined as follows. A new state y' is generated from a proposal density $q(y'|y_t)$. The new state is accepted with probability:

$$\alpha(y_t, y') = \begin{cases} \min\left(1, \frac{f(y')q(y_t|y')}{f(y_t)q(y'|y_t)}\right) & \text{if } f(y_t)q(y'|y_t) > 0 \\ 1 & \text{if } f(y_t)q(y'|y_t) = 0 \end{cases} \tag{2.24}$$

If the new candidate is accepted, set $y_{t+1} = y'$ otherwise set $y_{t+1} = y_t$. We can check that f is a stationary distribution for this Markov chain by checking detailed balance. For example, in the discrete case when $y_{t+1} \neq y_t$,

$$\begin{aligned} f(y_t)P(y_{t+1}|y_t) &= f(y_t)q(y_{t+1}|y_t)\min\left(1, \frac{f(y_{t+1})q(y_t|y_{t+1})}{f(y_t)q(y_{t+1}|y_t)}\right) && (2.25) \\ &= \min(f(y_t)q(y_{t+1}|y_t), f(y_{t+1})q(y_t|y_{t+1})) && (2.26) \\ &= f(y_{t+1})P(y_t|y_{t+1}) && (2.27) \end{aligned}$$

The Metropolis update therefore leaves the distribution f invariant. The ergodicity of the chain depends on the proposal, q, as well as on the equilibrium distribution, f. Theoretical results regarding ergodicity can be found in Tierney (1994) and Nummelin (1984).

The special case where the proposal is symmetric, $q(y'|y) = q(y|y')$ is called the Metropolis algorithm. The acceptance probability (2.24) becomes $\min(1, f(y')/f(y_t))$ after canceling $q(y'|y) = q(y|y')$. Therefore the new state will be accepted with probability 1 if it is a point with higher density and with probability $f(y')/f(y_t)$ otherwise.

If f is the joint distribution of $\boldsymbol{y} = (y_1, \ldots, y_N)$, instead of updating the whole $\boldsymbol{y}_t$, we can update one component at a time at each iteration. Suppose at time $t+1$ component i is updated using a proposal $q_i(y_i'|y_{t,i}, \boldsymbol{y}_{t,-i})$, where $\boldsymbol{y}_{t,-i} = (y_{t,1}, \ldots, y_{t,i-1}, y_{t,i+1}, y_{t,N})$. The acceptance probability for the proposed value is:

$$\alpha(y_t, y') = \begin{cases} \min\left(1, \frac{f(y_i'|\boldsymbol{y}_{t,-i})q_i(y_{t,i}|y_i', \boldsymbol{y}_{t,-i})}{f(y_{t,i}|\boldsymbol{y}_{t,-i})q_i(y_i'|y_{t,i}, \boldsymbol{y}_{t,-i})}\right) & \text{if } f(y_{t,i}|\boldsymbol{y}_{t,-i})q_i(y_i'|y_{t,i}, \boldsymbol{y}_{t,-i}) > 0 \\ 1 & \text{if } f(y_{t,i}|\boldsymbol{y}_{t,-i})q_i(y_i'|y_{t,i}, \boldsymbol{y}_{t,-i}) = 0 \end{cases} \tag{2.28}$$

where $f(y_{t,i}|\boldsymbol{y}_{t,-i})$ is the distribution of the i^{th} component of $\boldsymbol{y}$ given all the other remaining components. The remaining components are not changed at time t. It can be easily checked that the single-component Metropolis-Hastings algorithm leaves the distribution f invariant.

The proposal distribution can be, for example, a Gaussian distribution centered at the current point y_t, or more generally, any proposal of the form $q(y'|y) = q(|y - y'|)$. A proposal of this form gives rise to a special case of the Metropolis algorithm called random-walk Metropolis.

The best choice for the proposal depends on the equilibrium distribution f. Roberts, Gelman and Gilks (1997) and Gelman, Roberts and Gilks (1995b) provide some theoretical arguments for having an acceptance rate between 0.15 and 0.5 for high-dimensional problems. In general, the proposal step should be small enough to have a reasonably high acceptance rate and large enough to move a reasonable distance.

2.2.2 Gibbs Sampling

Suppose f is the joint distribution for $\boldsymbol{y} = (y_1, \ldots, y_N)$. We may not able to sample directly from f, but we might nevertheless be able to sample from the distribution of one component conditioned on all others. Given that at time t we have state $\boldsymbol{y}_t$, we can generate the state at time $t+1$ as follows:

Generate $y_{t+1,1}$ from the conditional distribution $f(y_1|y_{t,2}, \ldots, y_{t,N})$

Generate $y_{t+1,2}$ from the conditional distribution $f(y_2|y_{t+1,1}, y_{t,3} \ldots, y_{t,N})$

$$\vdots$$

Generate $y_{t+1,i}$ from the conditional distribution

$$f(y_i|y_{t+1,1}, \ldots, y_{t+1,i-1}, y_{t,i+1} \ldots, y_{t,N})$$

$$\vdots$$

Generate $y_{t+1,N}$ from the conditional distribution $f(y_N|y_{t+1,1}, \ldots, y_{t+1,N-1})$

Notice that the update of component i is done by using the new value for previously updated component $i-1$.

These transitions will leave the distribution f invariant if each step of the above transition leaves f invariant. At step i, all $y_j,\ j \neq i$ are unchanged, so the marginal distribution for these components does not change and y_i is generated from the desired conditional distribution. Therefore if the distribution of $\boldsymbol{y}_t$ is f, the joint distribution of $\boldsymbol{y}_{t+1}$, that is the product of the marginal distribution of $y_j,\ j \neq i$ and the conditional distribution of y_i given the other components, does not change. Although the transitions leave the distribution invariant, there is no guarantee that the resulting chain is ergodic. Ergodicity has to be established for each equilibrium distribution.

Conditions for irreducibility and aperiodicity of Gibbs sampler and therefore ergodicity were investigated by, among others, Chan (1993), Roberts and Polson (1994), and Roberts and Smith (1994).

The name of the algorithm, Gibbs sampling, was given by Geman and Geman (1984) and comes from the “Gibbs distributions” used in their Bayesian image restoration application. Gelfand and Smith (1990) applied Gibbs sampling to many Bayesian inference problems.

The Gibbs sampler can be seen as a special case of the single-component Metropolis-Hastings algorithm in which the proposals are the full conditional distributions and the acceptance probability is always one.

For Gaussian target distributions, rates of convergence for Gibbs sampling were established by Amit and Grenander (1991), Amit (1991), and Barone and Frigessi (1990). Roberts and Sahu (1997) obtain exact rates of convergence for the Gibbs sampler used for Gaussian target distributions. They studied the effect of correlation structure, updating, and blocking schemes on the convergence rates.

2.3 Overrelaxation

2.3.1 Markov Chains and Random Walks

Gibbs sampling and simple forms of the Metropolis algorithm typically move in a random walk, as at each step the direction in which the chain moves is randomized.

Gibbs sampling generates randomly at each step a new state from the conditional distribution, $f(y_i|\{y_j\}_{j\neq i})$. For highly-correlated variables, the conditional distribution is much narrower than the marginal distribution, and Gibbs sampling produces a chain that moves very slowly. Consider also the single-component random-walk Metropolis algorithm with a Gaussian proposal, centered at the current state and with standard deviation σ. If

σ is large, the acceptance rate will be small because mostly states of low density will be proposed, causing the chain to stay in the same state for a long time. If σ is small, the acceptance rate will be high, but the chain will move slowly because the proposal step is small.

Because high-dimensional Bayesian problems usually have high dependencies between the variables, the distance moved in one step is small. Moreover, at each iteration the direction of these steps is randomized, causing the chain to have a random walk behaviour. Therefore, on average n^2 steps are needed to move to a point n steps away. Due to the random walk behaviour, the chain converges slowly to the equilibrium distribution, and once it has converged, it will explore the distribution very slowly.

Gibbs sampling is invariant under translation and scaling, but is sensitive to rotation and therefore the preferred solution is to find a transformation that will decorrelate the variables, but for most high-dimensional Bayesian problem this is not feasible. At each step the chain moves a distance proportional to square root of the smallest eigenvalue of the variance-covariance matrix for which the marginal variances were all scaled to be one. To explore the state space the chain needs to move in the long direction that is given by the square root of the largest eigenvalue. If the components are highly correlated, because the step size and because of the random walk behaviour, Gibbs sampling needs to perform a large number of updates to move to a distant state. Global Metropolis is invariant under rotation, therefore a rotation will not improve its performance.

Suppressing random walks can be achieved either by the overrelaxation methods we will present in Section 2.3.2, or by hybrid Monte Carlo methods.

Hybrid Monte Carlo, introduced by Duane et al. (1987), uses Hamiltonian dynamics to generate a trajectory whose end point is proposed as a candidate state, and this state is accepted or rejected based on the change in total energy, as in the Metropolis algorithm.

Hamiltonian dynamics moves in the same direction for many steps, therefore suppressing random walks. Hybrid Monte Carlo is applicable to continuous state variables whose density has first derivatives that can be efficiently computed. The method requires fine tuning of two parameters: the step size used in the discretization of the dynamics, and the number of steps of the dynamics to perform before the accept/reject test.

Overrelaxation, presented in the next section, is another method of suppressing random walks.

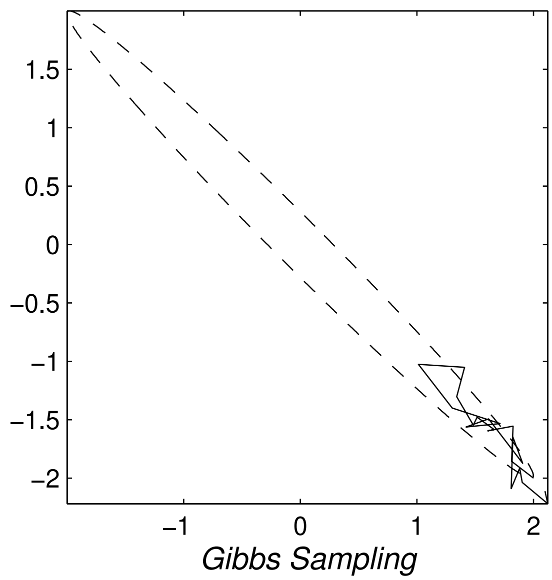
2.3.2 An Overview of Overrelaxation Techniques

Overrelaxation methods have been used for solving system of linear equations iteratively (Young 1971), and in particular for finding numerical solutions of partial differential equations. In this context, the method is known as "successive overrelaxation" (SOR). The first MCMC overrelaxation method was introduced by Adler (1981) in a physics context. Whitmer (1984) later studied different applications of overrelaxation to physics problems.

Adler's overrelaxation method can be applied to distributions f for which all full conditional distributions, $f(y_i|\{y_j\}_{j\neq i})$ are Gaussian. At time $t+1$ we update the i^{th} component of the vector $\boldsymbol{y}$. The other components are kept fixed. A new state $y_{t+1,i}$ is generated as follows:

$$y_{t+1,i} = \mu_i + \alpha(y_{t,i} - \mu_i) + \sigma_i \sqrt{(1-\alpha^2)}\,\epsilon \tag{2.29}$$

where μ_i and σ_i are the conditional mean and standard deviation of y_i given all other components $\{y_{t,j}\}_{j\neq i}$, ϵ is a Gaussian random variable with mean zero and variance one, and α is the parameter that controls the overrelaxation. For the algorithm to leave the distribution invariant, we need $-1 \leq \alpha \leq 1$. For $\alpha = 0$, (2.29) is Gibbs sampling. We illustrate how Adler's overrelaxation suppresses random walks for a bivariate Gaussian in



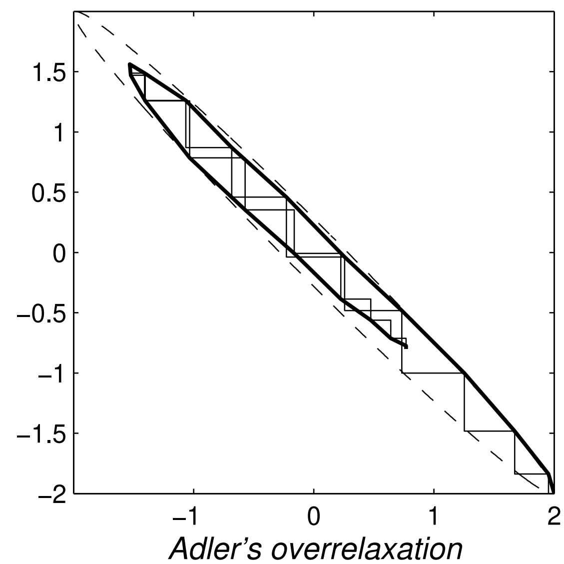


Figure 2.1: Gibbs sampling and Adler's overrelaxation method applied to a bivariate Gaussian distribution with correlation $\rho = -0.99$. The left plot shows twenty Gibbs sampling iterations and the right plot twenty iterations of overrelaxation with $\alpha = -0.98$. The thin line is the path obtained after updating each component. The thick lines are the paths after each iteration. An iteration consists of updating each component once.

Figure 2.1.

The efficiency of Adler's overrelaxation for a bivariate Gaussian distribution was investigated in Neal (1998). He found that with a correlation coefficient ρ of 0.998, Adler's overrelaxation with an $\alpha = -0.89$ is 22 times more efficient than Gibbs sampling in estimating $E(y_1)$ and 16 times more efficient in estimating $E(y_1^2)$. For a bivariate Gaussian distribution, Adler's overrelaxation performs the same for positively or negatively correlated components due to the symmetry of these distributions.

Barone and Frigessi (1990) derived the optimal value for the parameter α for some interesting cases of multivariate Gaussian distributions and showed that overrelaxation can be arbitrarily better than Gibbs sampling in certain cases. They find that the fastest rate of convergence for a multivariate Gaussian distribution with all correlations negative is achieved for $\alpha > 0$. For Gaussian distributions with all positive correlations, the fastest convergence rate is achieved for $\alpha < 0$. Adler's overrelaxation does not sample efficiently for Gaussian distributions with negatively-correlated components of dimension $N > 2$.

Extensions of overrelaxation to non-Gaussian distributions were investigated by Brown and Woch (1987). They propose two different methods. One is to transform the coordinates to bring conditional distributions to a Gaussian form. Overrelaxation is performed in the transformed space and then the state is transformed back to the original space. The second method proposed is an overrelaxed Metropolis algorithm. A new state is proposed corresponding to an $\alpha = -1$ overrelaxed step in which the point of reflection is the mode of the conditional distribution. To produce an ergodic chain, the method is combined with a few conventional Metropolis or Gibbs sampling steps in between.

Green and Han (1992) proposed to find a Gaussian approximation to the conditional distribution, $f(y_i|\{y_j\}_{j\neq i})$, whose mean, μ_i, and standard deviation, σ_i, do not depend on the current component to be updated ($y_{t,i}$). The component $y_{t,i}$ is then updated using a standard overrelaxation step as in (3.1), and the new state is accepted or rejected using a Metropolis acceptance test.

In a physics context, Fodor and Jansen (1994) proposed a method that is useful when the conditional distributions are unimodal. They proposed a new state y_i' on the other side of the mode such that $f(y_i'|\{y_j\}_{j\neq i}) = f(y_i|\{y_j\}_{j\neq i})$. The new state will be accepted or rejected based on the ratio of the derivative of the conditional density evaluated at the old state to the derivative evaluated at the proposed state. To achieve ergodicity one has to mix in some other transitions, such as Metropolis updates.

The drawback of all of these extensions of overrelaxation, except Brown and Woch's (1987) method of transformation to Gaussian, is that the rejection rate might be high.

Even a low rejection rate might diminish the benefits of overrelaxation in suppressing random walks. By rejecting an overrelaxed proposed state, the direction of the movement of the chain along the harder to investigate axis will be reversed. The rejection rate has to be low to allow the chain to move along the length of the distribution for a substantial period of time.

Neal (1998) proposed an overrelaxed Markov chain Monte Carlo method based on order statistics that can be efficiently applied when the full conditional distributions are such that the cumulative distribution and the inverse cumulative distribution functions can be easily computed. This method has the advantage that the proposed state is never rejected, thereby maintaining the feature of suppressing random walks of Adler's overrelaxation.

Neal (2003) discusses one way of doing overrelaxation using slice sampling. Due to the fact that the overrelaxation method is usually applied when the conditional distributions are unimodal, overrelaxed slice sampling could potentially be performed such that the rejection rate of the proposal state will be eliminated. Suppose we want to update component i by sampling from the conditional distribution $p(y_i) = f(y_i|\{y_j\}_{j\neq i})$. Slice sampling consists of three steps: (a) Generate z uniformly random from $(0, p(y_i))$ and define a horizontal 'slice' $S = \{y_i \text{ such that } z < p(y_i)\}$; (b) Use bisection method or some other to accurately approximate the slice with a single interval (L, R); (c) Propose a new state: $y_i' = \frac{L+R}{2} - \left(y_i - \frac{L+R}{2}\right)$. Rejection occurs only if the end points of the slice are not accurately approximated.

None of these methods is a perfect solution for all problems, therefore there is still room for development of new overrelaxation algorithms.

2.4 The Coupling Technique

Two chains are coupled when their transitions are determined by the same random numbers. Suppose we have two distributions, g and f, from which we want to draw samples $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$, respectively. At each iteration we randomly draw v_t from some distribution V and generate the updates for the two chain by $x_t = \phi_g(x_{t-1}, v_t)$ and $y_t = \phi_f(y_{t-1}, v_t)$. The transition functions, ϕ_f and ϕ_g, take two inputs, the state at time $t-1$ and some randomness v_t, and return the state at time t. These transition functions are chosen to keep the target distributions, f and g, invariant. In the applications described below, g and f are the same, but in our method presented in Chapter 5 they are different.

Coupling is used to prove theoretical bounds on the total variation distance between the distribution of the state of the Markov chain after k steps and the equilibrium distribution. We imagine running two chains, one starting from the equilibrium distribution and the other from some arbitrary distribution. The same transition functions and random numbers are used for the two chains. The total variation distance between the probability distributions of the two states after k steps can be bounded by the probability that the chains have not coalesced at step k. For a review of the coupling method in Markov chain theory context see Rosenthal (1995a). The convergence of general state space Markov chains using coupling techniques is analyzed in Rosenthal (1995b).

Coupling has also been used in Propp and Wilson's (1996) coupling from the past method of exact sampling. In this context, chains are started from all possible starting points and are run using the same transition probability function. Under certain assumptions, if we start the chains far enough back in the past, they will all coalesce by time zero. The point obtained in this way can be proved to come from the exact equilibrium distribution.

Chapter 3

Random Sequence Overrelaxation

Adler's overrelaxation moves well through the state space when applied to a Gaussian distribution with positively correlated components, but fails to do so well when the components are negatively correlated. In this chapter we present an overrelaxation algorithm that samples efficiently when applied to Gaussian distributions with negatively correlated components.

3.1 How Standard Overrelaxation behaves for Gaussian Distributions with Negatively Correlated Components

Let $\boldsymbol{y} = (y_1, ..., y_N)$ be a random vector distributed according to a multivariate Gaussian distribution. Adler's overrelaxation algorithm updates one component at a time in a deterministic order as follows:

$$y_{t+1,i} = \mu_i + \alpha(y_{t,i} - \mu_i) + \sigma_i \sqrt{(1 - \alpha^2)}\epsilon_{t+1} \tag{3.1}$$

$$y_{t+1,j} = y_{t,j} \quad \text{for } j \neq i \tag{3.2}$$

where i is the component updated at time t, $y_{t,i}$ is the i^{th} component of $\boldsymbol{y}$ at time t, μ_i and σ_i are the conditional mean and standard deviation of y_i given the other components, y_j for $j \neq i$, and ϵ_{t+1} is Gaussian random variable with mean zero and variance one. For $-1 \leq \alpha \leq 1$, the distribution of $\boldsymbol{y}$ is left invariant under the above update.

For a bivariate distribution, Adler's overrelaxation performs well for both negative and positive correlation between the components, but Adler's overrelaxation becomes inefficient for $N \geq 3$ when the components are negatively correlated.

Consider $\boldsymbol{y} = (y_1, y_2, \ldots, y_N) \sim N(\mathbf{0}, \mathbf{\Sigma})$ where

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & \rho & \ldots & \rho \\ \rho & 1 & \ldots & \rho \\ \vdots & & & \\ \rho & \rho & \ldots & 1 \end{pmatrix} \tag{3.3}$$

The variance-covariance matrix $\mathbf{\Sigma}$ is positive definite for $\rho \in (-1/(N-1), 1)$. For $N \geq 3$ and $\rho < 0$, a variance-covariance matrix of the form (3.3) has $N-1$ large eigenvalues and one small eigenvalue. When ρ is close to its lower limit of $-1/(N-1)$, the ratio of the largest to the smallest eigenvalue is large. The $N-1$ directions given by the $N-1$ eigenvectors corresponding to the large eigenvalues will then be difficult to explore.

For such a Gaussian distribution with $\rho < 0$, Adler's overrelaxation produces a chain that has a circular movement, exploring the states of the space inefficiently, and not providing a good estimate of the expected value of a non-linear function of the state. Figure 3.1 illustrates the movement of the chain for a Gaussian distribution with $N = 3$, $\rho = -0.499$. To produce a good estimate for the expected value of a linear function, the chain will have to move at least once in a circle, which due to the slow movement of the chain, will require a large number of iterations. Green and Han (1992) showed that the asymptotic variance of an estimator of the expectation of a linear function of the states

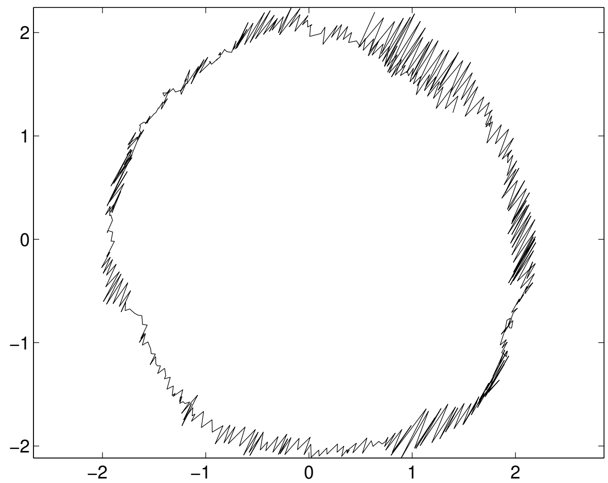


Figure 3.1: The plot shows the states of the chain projected on the space generated by the eigenvectors corresponding to the two large eigenvalues, obtained using Adler's overrelaxation with $\alpha = -0.99$ for a Gaussian distribution of dimension $N = 3$ with the variance-covariance matrix as in (3.3) for which $\rho = -0.499$. There are 500 states plotted, each state is obtained by updating all the components once.

goes to zero as α goes to -1. This is an asymptotic behaviour only, however, and in practice for a fixed length run this result will not provide us any information regarding the accuracy of the estimator. Also, if we were interested only in a linear function of the states, we could simply find the mode of the distribution and evaluate the function at the mode.

Adler's overrelaxation method with α close to -1 samples efficiently for Gaussian distributions $\mathbf{\Sigma}$ as in (3.3) and $\rho > 0$. On the other hand overrelaxation explores the space of the Gaussian distribution with negatively-correlated components very slowly. We propose a method that produces efficient samples for Gaussian distributions with $\rho < 0$. For Gaussian distribution with $\rho > 0$, our algorithm is not as efficient as Adler's overrelaxation, but it is more efficient than Gibbs sampling in most cases. We will call this method *random sequence overrelaxation (RSO)*.

3.2 Random Sequence Overrelaxation

The algorithm proposed eliminates the circular movement of Adler's overrelaxation for Gaussian distributions with negatively correlated components. Instead, movement in different directions is realized by updating the components in a sequence selected randomly and then repeated many times.

The random generation of a sequence is done as follows. A vector of length Ns is obtained by repeating the labels 1 to N for s times. This vector is then randomly permuted with equal probability for all permutations. Avoiding consecutive updating of the same component makes the method more efficient since updating the same component twice will produce almost no movement when α is close to -1. Therefore the algorithm goes through each component of this permuted vector and checks if two consecutive components have the same label. If two consecutive components at positions j and $j+1$ have the same label, the algorithm generates a random position, k, from $j+2$ to $Ns-1$. If the component at position $j+1$ has a different label than the components at positions k and $k+1$, we shift the components from positions $j+2$ to k to positions $j+1$ to $k-1$ and the component that was at position $j+1$ is placed at position k. If the random position k chosen is not acceptable, it is marked as being visited and a new random position is generated and, if not visited before, the admissibility of this new random position is checked. If all the positions from $j+2$ to Ns are visited and no position is acceptable, the algorithm finds an acceptable position from 1 to $j-2$. A random number, k, between 1 and $j-2$ is generated and if the component at position $j+1$ is different from the components at positions k and $k+1$, we shift to the right one position all the components at positions $k+1$ to j

and we place component $j+1$ at position $k+1$. Each two consecutive components of the vector are investigated in this way, and arranged such that no two consecutive components are the same. Note that a slightly bigger improvement could be achieved by generating vectors for which the first and the last component are different as well.

Using RSO overrelaxation with $\alpha = -1$ with a random sequence of length Ns repeated a number of times, r, produces a chain that moves in one direction until the end of the probability contour is reached, at which point the direction reverses. We illustrate how RSO works on a Gaussian distribution with $N = 3$. The variance-covariance matrix, $\mathbf{\Sigma}$, is of the form (3.3), with $\rho = -0.49999$. The degree of difficulty of investigating the space is given by the square root of the ratio of the largest to smallest eigenvalue of $\mathbf{\Sigma}$, which for this ρ is 273. It takes approximately 273 steps to move to a distant state. In Figure 3.2, we generate one random sequence at a time of length 6 and repeat it for 40 times to produce movement in the same direction. In this example there were three random sequences that account for the three directions. The updates are done with $\alpha = -1$.

There are three parameters to be chosen for this method: the number of times we update each component in a random sequence, s, the number of times this sequence is repeated, r, and the value of α used for overrelaxation updates. From the experiments done, we have noticed that efficiency is optimized when the length of the sequence is minimal. Therefore in our experiments we will use $s = 2$ or $s = 3$, producing a minimum length sequence of $2N$ or $3N$. The number of times the sequence is repeated, r, will be chosen such that the product of the length of the sequence and the number of times we repeat the sequence, rsN, is approximately equal to the square root of the ratio of the largest to the smallest eigenvalue. This ensures that one sequence can move the chain to an almost independent point. There are alternative ways of setting the third parameter α. Since it is not known what α is best, we might just set α to -1, but do the first of

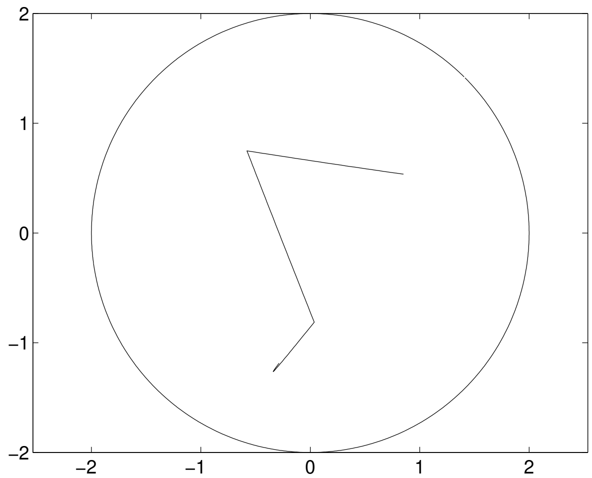


Figure 3.2: The Random Sequence Overrelaxation method with $\rho = -0.49999$, $s = 2$ and $r = 40$. The plot shows the paths produced using three random sequences ([3 1 2 1 2 3], [1 3 2 3 2 1] and [3 2 3 1 2 1]) projected on the plane generated by two eigenvectors of $\mathbf{\Sigma}$ corresponding to the two large eigenvalues. The paths show the states after one full random sequence was applied. The circle is the two standard deviation contour.

the rsN updates with $\alpha = 0$, in order to allow the chain to move from one probability contour to another. In the experiments presented later, however, we instead choose the best α for the RSO method (no $\alpha = 0$ updates), as we compare the method with Adler's overrelaxation for which also the best α is chosen.

3.3 Description of the Experiment

The random sequence overrelaxation method arose from the need to suppress random walks when negatively correlated components make Adler's overrelaxation work poorly. We would like the new method to work well for positively correlated components as well. The tests are done for Gaussian distributions for which the variance-covariance matrix has the form (3.3).

The experiments presented in the following sections consist of two parts. In the first stage, using autocorrelation time, estimated as described in Section 2.1, we chose the best values of α for Adler's overrelaxation and for RSO, when the function to be estimated was considered to be the square of the Euclidean norm. The values of α investigated were:
-0.999 -0.995 -0.99 -0.98 -0.97 -0.96 -0.95 -0.94 -0.93 -0.92 -0.91 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.99.
We considered four different values for the dimensionality, N, and for each N we investigated two positive correlations and two negative correlations. These results are presented in Figure 3.5 through Figure 3.12. The length of Markov chains for this stage of the simulations was such that a good estimate for the autocorrelation times could be obtained for α close to the optimal for each method. For α close to -1 or 1, the autocorrelation times may be poorly estimated in some cases.

In the second stage, we ran very long chains using the optimal values for α found in the first stage. Based on these chains, we obtained accurate estimates for the autocorrelation times, which were used to provide good estimates of the efficiency of RSO versus Adler's overrelaxation and versus Gibbs sampling. We estimated the autocorrelation time for three functions of the state: the sum of the squares of the components, the first component squared, and an indicator function that is 1 if the absolute value of the first component is greater than 1.5 and 0 otherwise. The same chain was used to estimate the autocorrelations for all three functions of the state, using the α that was optimal for the sum of the squares of the components. The optimal α for the sum of the squares of the components is close to the optimal values for α for the other two functions, so the autocorrelation times should be quite close to the optimal values for all functions of the state.

We consider three positive correlations to demonstrate that the efficiency of RSO versus

Adler's overrelaxation, for a fixed N, remains constant as the correlations approach 1. As the positive correlation becomes closer to its upper limit one needs to run very long chains in order to obtain good estimates for the autocorrelation times for the Gibbs sampling. Therefore it is computationally very intensive to investigate all values of α from -1 to 1 for the strongest positive correlation that is found under the double line in Tables 3.2 through 3.13. Therefore we have not found the optimal α for these cases and rather we used either the same α as for the less strong correlation or an α closer to -1.

In Table 3.1 we summarized the lengths of the chain for all methods and the lags at which the autocorrelations are close to zero. The last column in these tables represents the number of iterations for which Gibbs sampling and Adler's overrelaxation were run. One iteration for Gibbs sampling and Adler's overrelaxation consists of updating each component once. For RSO an iteration consists of updating the components by using once the random sequence of length Ns. The number of iterations for RSO is adjusted such that each component will be updated the same number of times as for Gibbs sampling and Adler's overrelaxation. Therefore the number of iterations will be different for the methods. For $\rho < 0$, for Gibbs sampling and Adler's overrelaxation, we adjust the length of chains by taking only the k^{th} state of the chain, where k is chosen such that the chains will have the same length. For $\rho > 0$, taking every k^{th} state is not always possible because the lag at which the autocorrelations for Adler's overrelaxation are close to zero is very small and by taking every k^{th} iteration the remaining states will become independent and therefore not provide a good estimate for the autocorrelation time. When the lengths of the chains are different, we adjust the autocorrelation time by multiplying the autocorrelation time for the RSO with a factor determined by the ratio of the length for Adler's overrelaxation chain to the length of RSO chain.

For the strongest positive correlation for each N, the chains using Gibbs sampling and

$N=3$	GS		AO		RSO - Seq=2		RSO - Seq=3		Total length
ρ	length	lag	length	lag	length	lag	length	lag	for AO and GS
-0.49950	500000	250	500000	250	500000	80	500000	120	3000000
-0.49990	500000	1200	500000	1200	500000	120	500000	600	3000000
0.99750	166667	350	1000000	60	499998	250	333332	120	1000000
0.99950	166667	1500	1000000	150	499996	750	333333	300	1000000
0.99988	166667	8000	1000000	200	500006	1500	333342	600	1000000

$N=4$	GS		AO		RSO - Seq=2		RSO - Seq=3		Total length
ρ	length	lag	length	lag	length	lag	length	lag	for AO and GS
-0.33300	166667	200	166667	200	166667	60	166666	60	1000000
-0.33325	250000	600	250000	500	250000	100	250002	100	1500000
0.99700	1000000	1500	1000000	60	500000	500	333333	200	1000000
0.99940	1000000	6000	1000000	200	500000	1800	333333	900	1000000
0.99985	1000000	50000	1000000	300	500000	3600	333340	1500	1000000

$N=10$	GS		AO		RSO - Seq=2		RSO - Seq=3		Total length
ρ	length	lag	length	lag	length	lag	length	lag	for GS and AO
-0.11109	500000	800	500000	800	500000	120	500000	120	3000000
-0.11110	500000	1800	500000	1800	500001	280	500000	280	3000000
0.99850	250000	1800	250000	40	250000	5000	250001	550	1500000
0.99965	500000	8000	500000	60	500001	12000	500001	1500	3000000
0.99990	500000	20000	500000	150	500000	20000	500000	3000	3000000

$N=11$	GS		AO		RSO - Seq=2		RSO - Seq=3		Total Length
ρ	length	lag	length	lag	length	lag	length	lag	for AO and GS
-0.099985	500000	900	500000	900	500000	120	500000	200	3000000
-0.099996	500000	4000	500000	2500	500000	300	500000	900	3000000
0.998650	250000	2500	250000	40	250000	4000	250001	600	1500000
0.999650	500000	9000	500000	60	500001	30000	500001	1800	3000000
0.999900	500000	9000	500000	200	499999	50000	500000	3000	3000000

Table 3.1: The length of the chains used for each method and the lags at which the autocorrelations are close to zero.

RSO have high autocorrelations up to a large lag comparable to the length of the chain. There may therefore not be enough independent points to produce a reliable estimate for the autocorrelation times.

3.4 Discussion of Simulation Results

The RSO method behaves differently for odd versus even dimensional multivariate Gaussian distributions and for positive versus negative correlations. In the next sections we present the performance of RSO for these situations.

3.4.1 Odd-dimensional Gaussian distribution with negatively-correlated components

For Gaussian distributions of odd dimension, whether the length of the random sequence is odd or even affects the efficiency of random sequence overrelaxation. Figure 3.3 shows the slow movement of the chain using RSO for a Gaussian distribution of dimension $N = 3$, with length of sequence, $3N$. Note that the plot spans a range of only 0.08 on one axis and 0.07 on the other, whereas the standard deviation is one. Compare this with Figure 3.2 where RSO with $s = 2$ was used, for which a range of 1.5 on one axis and 2.5 on the other axis was spanned with a similar computational effort.

A variance-covariance matrix as in (3.3) with ρ close to the low extreme, $-1/(N-1)$, has $N-1$ large eigenvalues and one small eigenvalue. The distribution therefore has a flat ellipsoidal shape. If we start with a point on one side of the flat ellipsoid, by updating the components once using a random sequence of odd length, the chain ends up on the other side of the ellipsoid. When we repeat the random sequence a second time, we almost retrace our steps, rather than accomplishing the desired result of moving consistently in the same direction. An even length random sequence accomplishes the movement in the same direction because after one repetition of the sequence, the state ends up on the same side of the flat ellipsoid, so that by repeating the sequence we continue the movement in the same direction.

We illustrate the performance of RSO on Gaussian distributions with $N = 3$ and $N = 11$. In Figure 3.5 and Figure 3.11 we compare the autocorrelation time for the RSO applied with even length and odd length sequences to the autocorrelation time for Adler's overrelaxation for values of α ranging from -0.995 to 0.9. All three chains have a similar autocorrelation time for α away from -1. The best performance for RSO is achieved for values of α close to -1 and for random sequences of even length. The optimal values of α for Adler's overrelaxation when applied to Gaussian distributions with negatively-correlated components are positive and around 0. This result is similar to Barone and Frigessi's (1990) results. They found that for positively-correlated components Adler's overrelaxation is best if performed with $\alpha < 0$ and with $\alpha > 0$ when applied to negatively-correlated components case.

In Table 3.2 through Table 3.4 and Table 3.11 through 3.13 we presented the results for two odd dimension distributions, $N = 3$ and $N = 11$. For each dimension we demonstrate the efficiency on two variance-covariance matrices as in (3.3) with $\rho < 0$. For all three functions of the components that we investigate, RSO with even length sequence is more efficient than Adler's overrelaxation and than Gibbs sampling. The efficiency of the RSO method with $s = 2$ ranges from 3 to 8 times better than the Adler's overrelaxation and from 4 to 8 times better than Gibbs sampling. As the correlation gets close to its limit, $-1/(N-1)$, the efficiency of the RSO compared to Adler's overrelaxation and Gibbs sampling increases. Since Gibbs sampling performs a random walk, a factor of two increase in the square root of the ratio of the largest to smallest eigenvalue produces a factor of four increase in the autocorrelation time. Since RSO suppresses random walks, a factor of two increase in the square root of the ratio of the largest to smallest eigenvalue will produce a factor of two increase in the efficiency of RSO method compared to Gibbs sampling. For $N = 3$, the efficiency of RSO versus Gibbs sampling increases from 4 to about 11 when the square root of the eigenvalues increases by a factor of 2. For $N = 11$, for a factor of two increase in the square root of the eigenvalues there is a increase in the RSO's efficiency versus Gibbs sampling from around 6 to around 11 for the square norm. Due to the round off errors we see an increase from around 4 to only around 6 for the square of the first component function and for the indicator function.

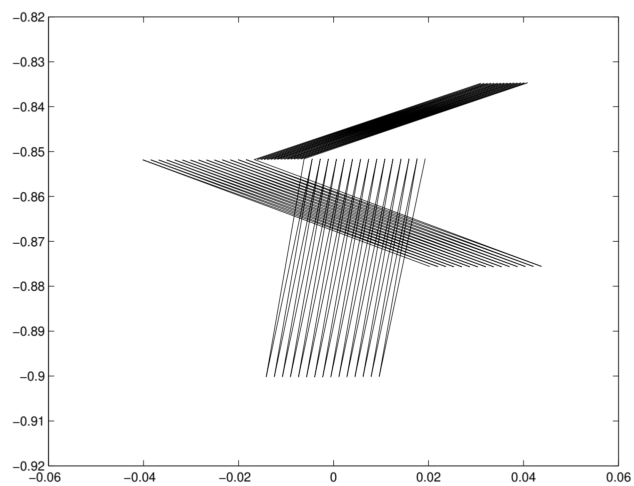


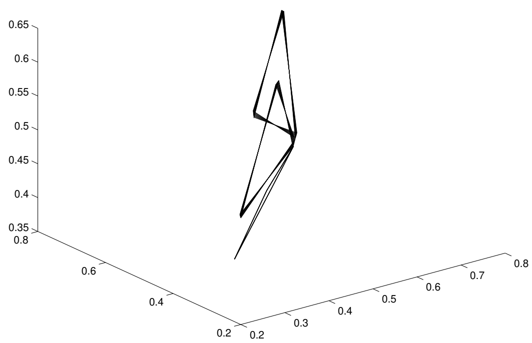
Figure 3.3: Random Sequence Overrelaxation method with $N = 3$, $\rho = -.49999$, $s = 3$ and $r = 30$. The plot shows the paths produced using three random sequences ([1 2 1 3 1 3 2 3 2], [1 2 3 2 1 3 2 3 1] and [1 3 2 3 2 1 2 1 3]) projected on the space generated by two eigenvectors of $\mathbf{\Sigma}$ corresponding to the two large eigenvalues. The paths show the states after one full random sequence was applied.

3.4.2 Even-dimensional Gaussian distribution with negatively-correlated components

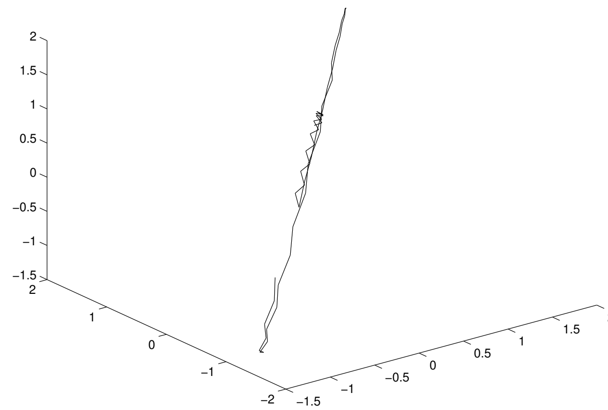
RSO is more efficient than Gibbs sampling and Adler's overrelaxation for even dimensional problems, and the efficiency of the method does not depend on whether the parameter s is odd or even, as the length of the sequence, Ns, is even in both situations. As for odd dimensional distributions, the efficiency of the method as compared to Gibbs sampling and Adler's overrelaxation increases as the correlation approaches its lower limit because RSO moves efficiently in the different directions given by the random sequences. Tables 3.5 through 3.7 and Tables 3.8 through 3.10 and Figures 3.7 and 3.9 show the results for the even dimensional case when $N = 4$ and $N = 10$. RSO is 3 to 8 times more efficient than Gibbs sampling. Since the optimal α for Adler's overrelaxation is near 0, its performance is similar to that of Gibbs sampling.

3.4.3 Gaussian Distribution with Positively Correlated Components

For Gaussian distributions with positive correlations, Adler's overrelaxation with α close to -1 produces efficient estimators. The results for Gaussian distributions with variance-covariance matrix as in (3.3) are shown in Figures 3.6, 3.8, 3.10 and 3.12. For $\rho > 0$ the variance-covariance matrix has one large eigenvalue and $N - 1$ small eigenvalues. Adler's overrelaxation suppresses the random walk in the long direction corresponding to the large eigenvalue. RSO is not as good as Adler's overrelaxation, but the efficiency for RSO versus Adler's overrelaxation for a fixed N does not get worse as the correlation approaches one. For $N = 3$, for example, RSO is 5 to 10 times more inefficient than Adler's overrelaxation, for different values of $\rho > 0$ and different functions of state. For $N = 10$ and $N = 11$, the



$s = 2$, $r = 27$. The plot shows the paths using three random sequences: 1 2 3 1 3 2, 1 3 2 3 1 2, 1 2 3 2 3 1



$s = 3$, $r = 18$. The plot shows the paths using three random sequences: 2 3 2 3 1 3 1 2 1, 2 3 1 2 3 2 1 3 1, 1 2 3 1 2 3 1 2 3.

Figure 3.4: RSO with $N = 3$, $\rho = 0.999$, $\alpha = -1$. The paths show the states after one full random sequence was applied. Illustrates the behaviour of the chains for $s = 2$ versus $s = 3$. Note the different scales for the two plots.

inefficiency of RSO is greater, because the probability of generating a good sequence (e.g. $[1, 2, \ldots, N]$) is low.

We notice that the method is slightly more efficient for $s = 3$ for low dimensional problems, and the discrepancy in efficiency increases as the dimensionality increases. For $N = 10$ and $N = 11$, RSO with $s = 3$ is 10 times more efficient than RSO with $s = 2$. Figure 3.4 illustrates the difference in the behaviour of RSO with $s = 2$ versus $s = 3$. RSO with $s = 2$ moves a range of around 0.3 in each direction, whereas for $s = 3$ the range spanned is around 2.8. However, for low dimensional problems the autocorrelation times are not very different because of the high probability of generating a good sequence such as $[1, 2, \ldots, N]$.

RSO is 2 to 42 times more efficient than Gibbs sampling.

For variance-covariance matrices as in (3.3) with $\rho > 0$, RSO does better than Gibbs sampling, but worse than Adler's overrelaxation, especially for large N.

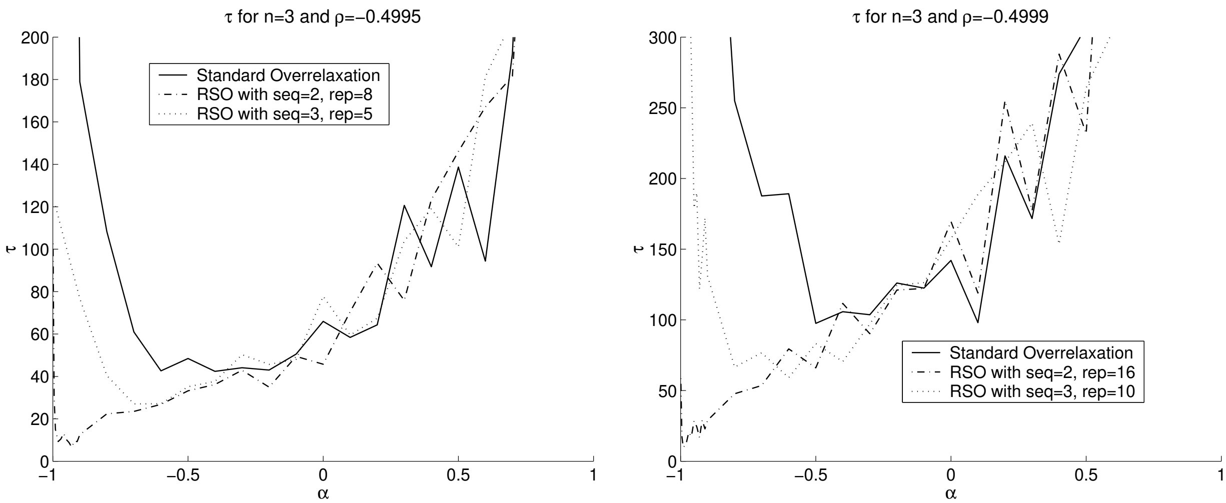


Figure 3.5: $N = 3$, $\rho < 0$. Autocorrelation times of the square of the Euclidean norm of the states for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

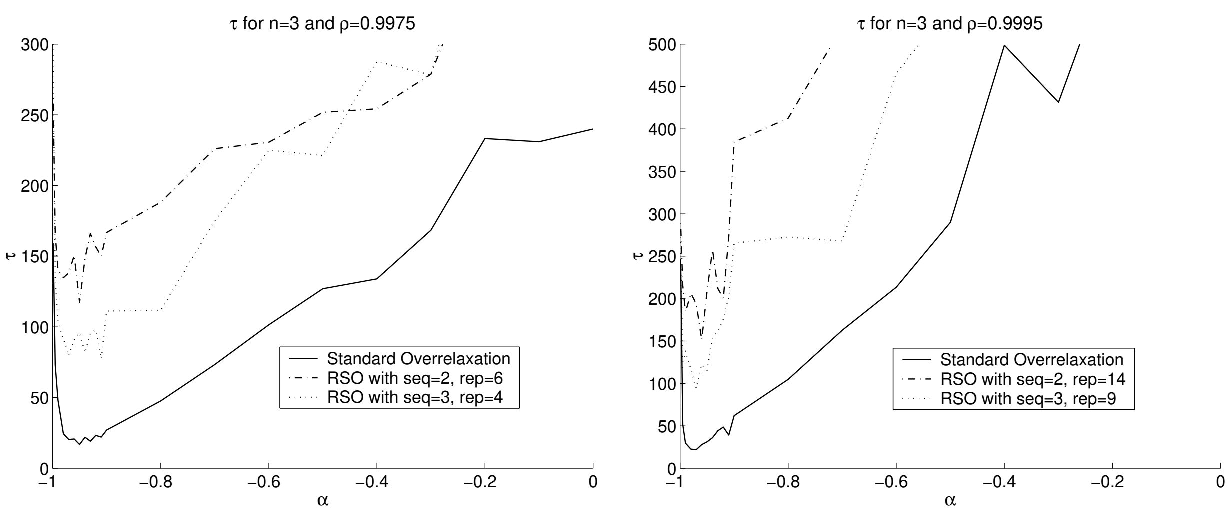


Figure 3.6: $N = 3$, $\rho > 0$. Autocorrelation times of the square of the Euclidean norm of the states for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

$N = 3$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.49950	38.7	54	-0.30	48	-0.94	8	12	-0.60	5	31	4.39	1.75	3.93	1.57
-0.49990	86.6	283	-0.50	222	-0.98	16	25	-0.80	10	125	11.21	2.26	8.79	1.77
0.99750	34.6	447	-0.95	18	-0.95	6	128	-0.91	4	92	3.49	4.86	0.14	0.20
0.99950	77.4	2257	-0.97	45	-0.96	14	436	-0.97	9	221	5.17	10.21	0.10	0.20
0.99988	158.1	14536	-0.99	84	-0.98	26	741	-0.98	18	417	19.62	34.89	0.11	0.20

Table 3.2: Efficiency for the square of the Euclidean norm, $N = 3$.

$N = 3$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.49950	38.7	46	-0.30	34	-0.94	8	11	-0.60	5	29	4.27	1.59	3.14	1.17
-0.49990	86.6	238	-0.50	155	-0.98	16	21	-0.80	10	126	11.06	1.89	7.19	1.23
0.99750	34.6	446	-0.95	18	-0.95	6	128	-0.91	4	92	3.50	4.86	0.14	0.20
0.99950	77.4	2255	-0.97	45	-0.96	14	436	-0.97	9	221	5.17	10.20	0.10	0.20
0.99988	158.1	14536	-0.99	84	-0.98	26	741	-0.98	18	417	19.62	34.88	0.11	0.20

Table 3.3: Efficiency for the square of the first component, $N = 3$.

$N = 3$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.49950	38.7	32	-0.30	24	-0.94	8	8	-0.60	5	20	3.90	1.57	2.94	1.18
-0.49990	86.6	171	-0.50	112	-0.98	16	18	-0.80	10	90	9.60	1.90	6.32	1.25
0.99750	34.6	308	-0.95	14	-0.95	6	98	-0.91	4	68	3.15	4.50	0.14	0.20
0.99950	77.4	1829	-0.97	34	-0.96	14	324	-0.97	9	156	5.65	11.70	0.10	0.22
0.99988	158.1	10539	-0.99	64	-0.98	26	563	-0.98	18	319	18.73	33.00	0.11	0.20

Table 3.4: Efficiency for indicator function of the absolute value of the first component being greater than 1.5, $N = 3$.

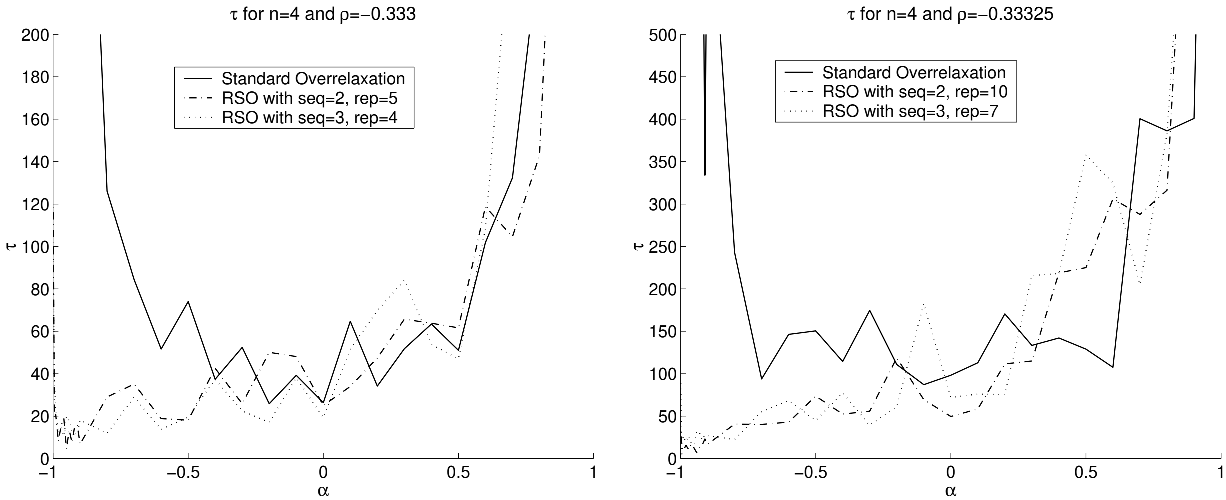


Figure 3.7: $N = 4$, $\rho < 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

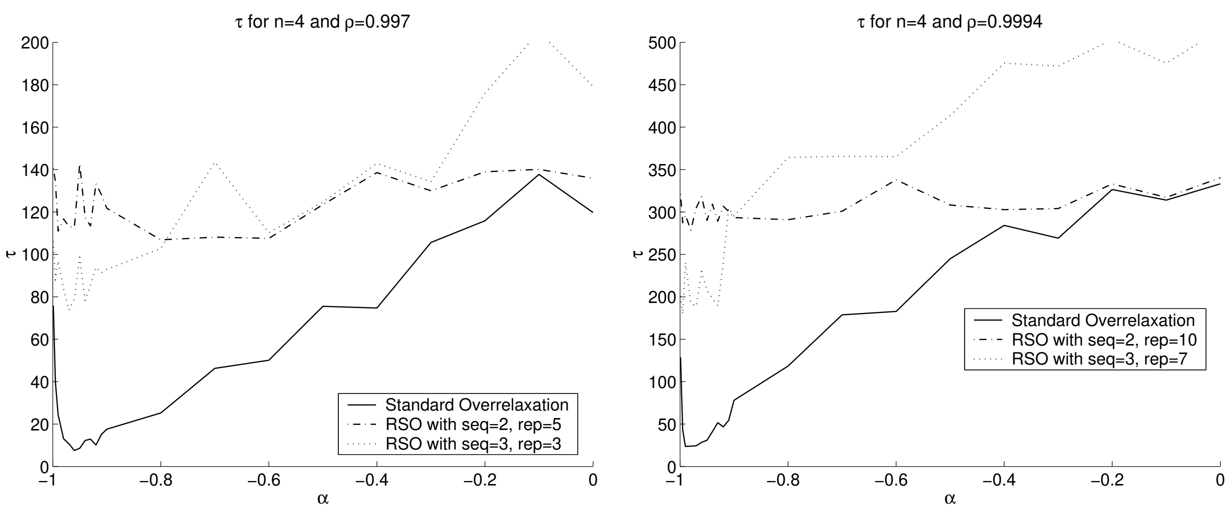


Figure 3.8: $N = 4$, $\rho > 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

$N=4$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.33300	36.5	47	0.00	47	-0.95	5	12	-0.96	4	13	4.10	3.71	4.10	3.71
-0.33325	73.0	177	-0.10	180	-0.96	10	24	-0.98	7	27	7.52	6.64	7.64	6.75
0.99700	36.5	529	-0.96	20	-0.80	5	329	-0.97	3	154	1.61	3.44	0.06	0.13
0.99940	81.6	2487	-0.99	52	-0.98	10	966	-0.99	7	539	2.58	4.62	0.05	0.10
0.99985	163.3	16433	-0.99	90	-0.99	20	1796	-0.99	14	1119	9.15	14.69	0.05	0.08

Table 3.5: Efficiency for the square of the Euclidean norm, $N=4$.

$N=4$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.33300	36.5	35	0.00	35	-0.95	5	9	-0.96	4	10	4.08	3.50	4.08	3.50
-0.33325	73.0	134	-0.10	122	-0.96	10	19	-0.98	7	20	6.89	6.62	6.30	6.05
0.99700	36.5	526	-0.96	20	-0.80	5	328	-0.97	3	153	1.61	3.44	0.06	0.13
0.99940	81.6	2484	-0.99	52	-0.98	10	965	-0.99	7	538	2.57	4.61	0.05	0.10
0.99985	163.3	16431	-0.99	90	-0.99	20	1797	-0.99	14	1119	9.15	14.69	0.05	0.08

Table 3.6: Efficiency for the square of the first component, $N=4$.

$N=4$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.33300	36.5	24	0.00	24	-0.95	5	7	-0.96	4	7	3.66	3.24	3.66	3.24
-0.33325	73.0	91	-0.10	82	-0.96	10	15	-0.98	7	15	6.20	6.05	5.58	5.45
0.99700	36.5	374	-0.96	15	-0.80	5	229	-0.97	3	116	1.63	3.22	0.07	0.13
0.99940	81.6	1721	-0.99	38	-0.98	10	676	-0.99	7	395	2.55	4.36	0.06	0.10
0.99985	163.3	10665	-0.99	69	-0.99	20	1376	-0.99	14	691	7.75	15.42	0.05	0.10

Table 3.7: Efficiency for indicator function of the absolute value of the first component being greater than 1.5, $N=4$.

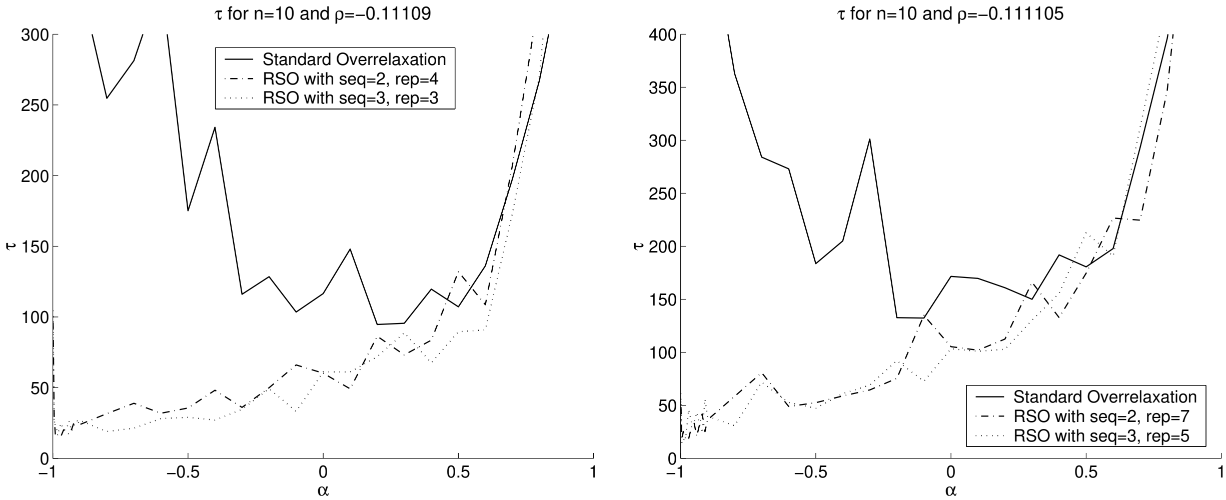


Figure 3.9: $N = 10$, $\rho < 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

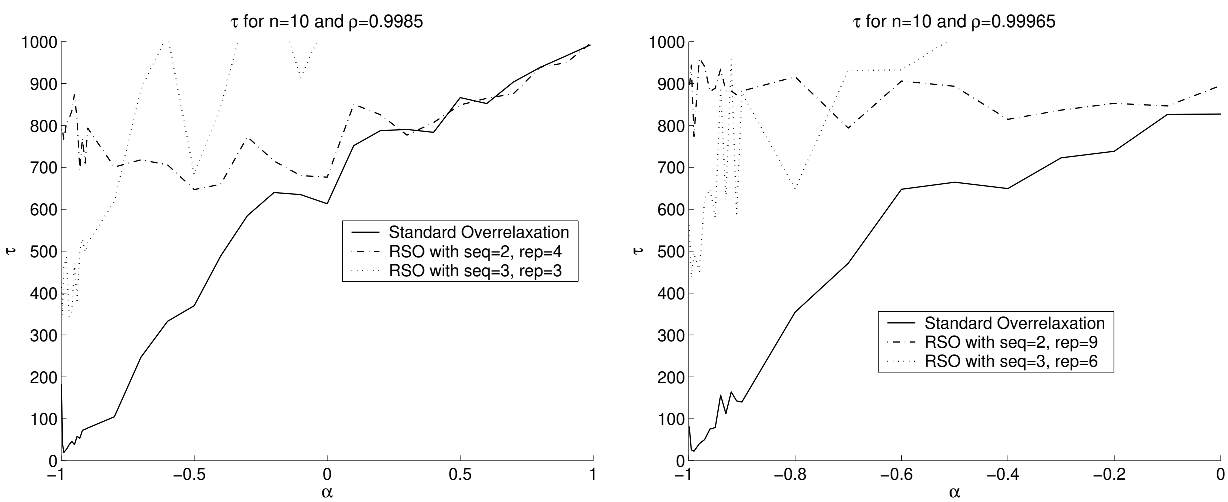


Figure 3.10: $N = 10$, $\rho > 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

$N=10$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.11109	76.5	181	0.20	169	-0.99	4	33	-0.98	3	34	5.51	5.34	5.151	4.99
-0.11110	142.1	481	-0.10	506	-0.98	7	61	-0.99	5	75	7.83	6.38	8.247	6.72
0.99850	81.6	3194	-0.99	8	-0.97	4	1367	-0.97	3	151	2.34	21.11	0.006	0.05
0.99965	169.0	14264	-0.99	16	-0.99	9	2993	-0.99	6	405	4.77	35.25	0.005	0.04
0.99990	316.2	61695	-0.99	44	-0.99	16	11020	-0.99	11	827	5.60	74.58	0.004	0.05

Table 3.8: Efficiency for the square of the Euclidean norm, $N = 10$.

$N=10$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.11109	76.5	87	0.20	108	-0.99	4	17	-0.98	3	26	5.08	3.37	6.323	4.20
-0.11110	142.1	313	-0.10	288	-0.98	7	47	-0.99	5	45	6.70	6.95	6.180	6.41
0.99850	81.6	3185	-0.99	8	-0.97	4	1364	-0.97	3	151	2.34	21.12	0.006	0.05
0.99965	169.0	14257	-0.99	16	-0.99	9	2991	-0.99	6	404	4.77	35.27	0.005	0.04
0.99990	316.2	61687	-0.99	44	-0.99	16	11019	-0.99	11	827	5.60	74.60	0.004	0.05

Table 3.9: Efficiency for the square of the first component, $N = 10$.

$N=10$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.11109	76.5	63	0.20	82	-0.99	4	13	-0.98	3	19	4.94	3.37	6.503	4.43
-0.11110	142.1	220	-0.10	203	-0.98	7	35	-0.99	5	32	6.27	6.79	5.789	6.27
0.99850	81.6	2137	-0.99	6	-0.97	4	1071	-0.97	3	110	1.99	19.46	0.006	0.06
0.99965	169.0	11710	-0.99	12	-0.99	9	1919	-0.99	6	292	6.10	40.15	0.006	0.04
0.99990	316.2	53783	-0.99	33	-0.99	16	10110	-0.99	11	600	5.32	89.66	0.003	0.06

Table 3.10: Efficiency for indicator function of the absolute value of the first component being greater than 1.5, $N = 10$.

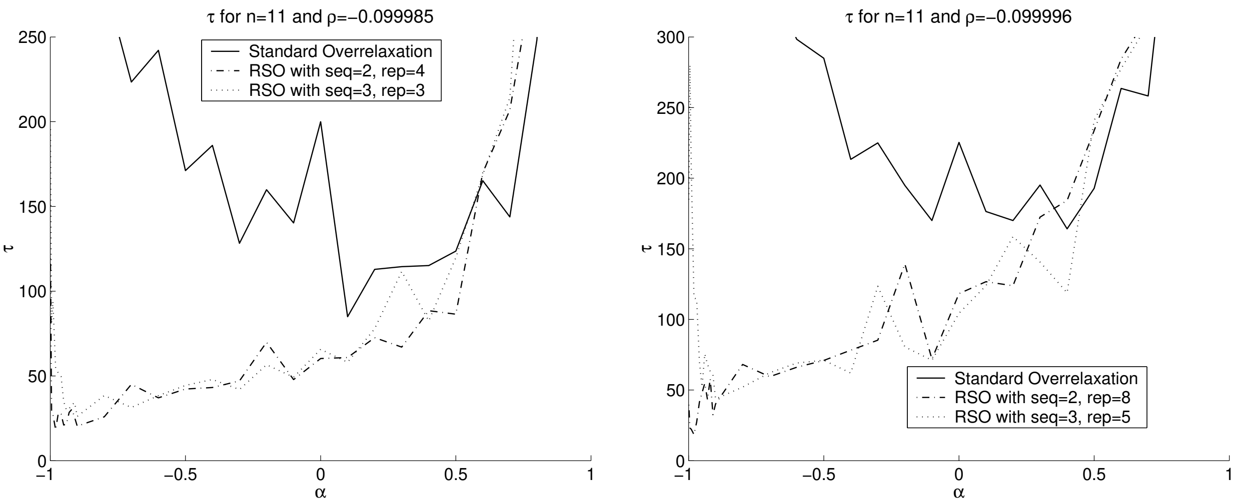


Figure 3.11: $N = 11$, $\rho < 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

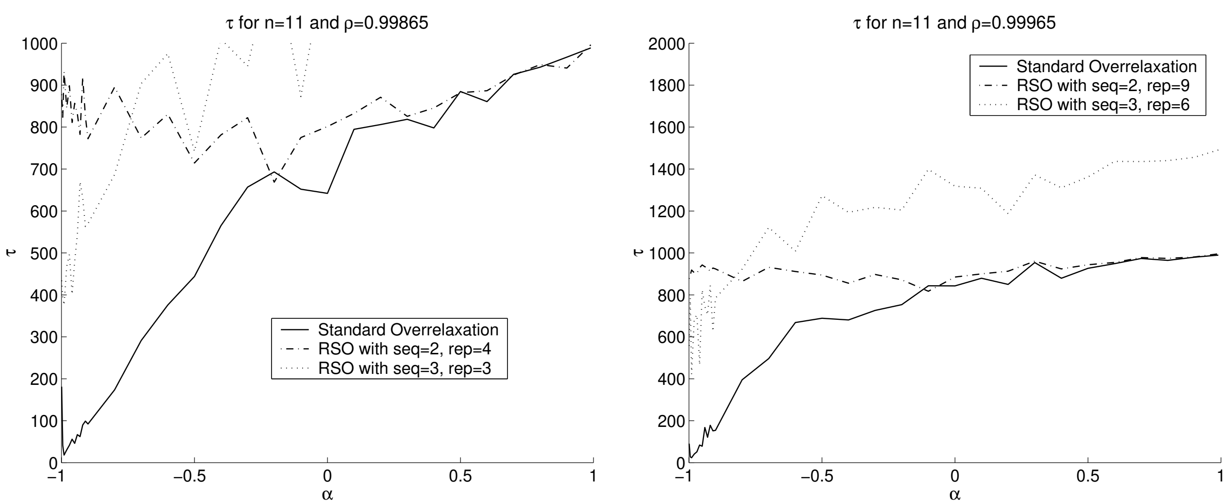


Figure 3.12: $N = 11$, $\rho > 0$. Autocorrelation times of the square of the Euclidean norm of the state for RSO with length of sequence odd, RSO with length of sequence even, and standard overrelaxation, for different values of α.

$N = 11$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.099985	85.6	183	0.10	174	-0.98	4	32	-0.90	3	56	5.77	3.30	5.469	3.13
-0.099996	165.8	832	0.20	659	-0.98	8	78	-0.90	5	225	10.66	3.69	8.441	2.93
0.998650	90.2	3644	-0.99	9	-0.99	4	1254	-0.99	3	176	2.90	20.66	0.007	0.05
0.999650	177.3	16301	-0.99	17	-0.99	9	9084	-0.99	6	475	1.79	34.34	0.002	0.04
0.999900	331.6	50272	-0.99	46	-0.99	17	20838	-0.99	11	787	2.41	63.86	0.002	0.06

Table 3.11: Efficiency for the square of the Euclidean norm, $N = 11$.

$N = 11$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.099985	85.6	103	0.10	113	-0.98	4	25	-0.90	3	50	4.21	2.07	4.604	2.27
-0.099996	165.8	349	0.20	471	-0.98	8	58	-0.90	5	217	6.03	1.61	8.144	2.18
0.998650	90.2	3631	-0.99	9	-0.99	4	1252	-0.99	3	176	2.90	20.62	0.007	0.05
0.999650	177.3	16294	-0.99	17	-0.99	9	9080	-0.99	6	474	1.79	34.35	0.002	0.04
0.999900	331.6	50269	-0.99	46	-0.99	17	20833	-0.99	11	787	2.41	63.88	0.002	0.06

Table 3.12: Efficiency for the square of the first component, $N = 11$.

$N = 11$	$\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$	GS	AO		RSO - Seq=2			RSO - Seq=3			Efficiency			
ρ		τ_1	α	τ_2	α	rep	τ_3	α	rep	τ_4	$\frac{\tau_1}{\tau_3}$	$\frac{\tau_1}{\tau_4}$	$\frac{\tau_2}{\tau_3}$	$\frac{\tau_2}{\tau_4}$
-0.099985	85.6	70	0.10	76	-0.98	4	17	-0.90	3	36	4.07	1.94	4.440	2.12
-0.099996	165.8	253	0.20	340	-0.98	8	43	-0.90	5	154	5.86	1.65	7.865	2.21
0.998650	90.2	2262	-0.99	7	-0.99	4	983	-0.99	3	124	2.30	18.27	0.007	0.05
0.999650	177.3	13578	-0.99	13	-0.99	9	5545	-0.99	6	317	2.45	42.86	0.002	0.04
0.999900	331.6	43492	-0.99	34	-0.99	17	16960	-0.99	11	541	2.56	80.40	0.002	0.06

Table 3.13: Efficiency for indicator function of the absolute value of the first component being greater than 1.5, $N = 11$.

Chapter 4

Jacobian Overrelaxation for Non-Gaussian Distributions

Adler's overrelaxation suppresses random walks for Gaussian distributions with positively-correlated components. However, Adler's overrelaxation is applicable only to distributions for which the full conditional densities are all Gaussian. We propose an algorithm effective in suppressing random walks for distributions with positively-correlated components that can be applied to any distribution as long as we can find the mean or mode or some other approximation of location of the distribution of each component conditioned on all others. We call this method *Jacobian overrelaxation* (JO).

Section 4.1 presents the idea of the algorithm and shows how it works when applied on a bivariate Gaussian distribution. Section 4.2 proves that the Markov chain constructed by Jacobian overrelaxation leaves the desired distribution invariant. In this section the ergodicity of the algorithm is investigated as well. In Section 4.3 we explore the choice of the tuning parameters for simple Gaussian distributions and provide a guide for how these parameters should be chosen for more general distributions. Section 4.4 examines the performance of Jacobian overrelaxation on a Bayesian logistic regression problem.

4.1 The Jacobian Overrelaxation Algorithm

One iteration of the Jacobian overrelaxation algorithm (JO) generates S candidate states using overrelaxation type updates, as in (3.1), but without adding the Gaussian random noise. It then chooses from among these S states, plus the current state, with probabilities proportional to the densities of the states multiplied by Jacobian factors for each state. At each iteration, the algorithm randomly chooses the overrelaxation parameter α, the component, j, to be updated first with α, and the number of updates, r, that will be done using α. The remaining $S-r$ steps will be overrelaxed with $1/\alpha$, updating the components in reverse order, starting with the component $(j-1) \pmod N$. To perform an overrelaxation type step we need an approximation for the mean or the mode for each conditional distribution for the component to be updated conditioned on all others. Note that we do not need the exact mean or mode for the method to be valid. We will denote by $\mu_p(\boldsymbol{z})$ this function of the components of $\boldsymbol{z}$ except the p^{th} component of $\boldsymbol{z}$.

The procedure to generate the next state, y_{t+1}, from the current state, y_t, is:

1) Generate α uniformly at random on $[a, b]$.

2) Generate j, the first component to update, uniformly from $\{0, 1, \ldots, N-1\}$, where N is the dimension of the state.

3) Generate r uniformly from $\{0, 1, \ldots, S\}$.

4) Let $\boldsymbol{z}^{(0)} = \boldsymbol{y}_t$ and $\boldsymbol{z}^{(k)} = \left(z_0^{(k)}, \ldots, z_{N-1}^{(k)}\right)$ and let $p_k = (j+k) \pmod N$. The algorithm generates the states $\boldsymbol{z}^{(-(S-r))}, \ldots, \boldsymbol{z}^{(-1)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(r)}$ as follows:

For $k = 0, \ldots, r-1$

$$\begin{aligned} z_{p_k}^{(k+1)} &= \mu_{p_k}(\boldsymbol{z}^{(k)}) + \alpha\left(z_{p_k}^{(k)} - \mu_{p_k}(\boldsymbol{z}^{(k)})\right) \\ z_i^{(k+1)} &= z_i^{(k)} \text{ for all } i \neq p_k \end{aligned} \tag{4.1}$$

For $k = -1, \ldots, -(S-r)$

$$\begin{aligned} z_{p_k}^{(k)} &= \mu_{p_k}(\boldsymbol{z}^{(k+1)}) + \frac{1}{\alpha}\left(z_{p_k}^{(k+1)} - \mu_{p_k}(\boldsymbol{z}^{(k+1)})\right) \\ z_i^{(k)} &= z_i^{(k+1)} \text{ for all } i \neq p_k \end{aligned} \tag{4.2}$$

Note that either of these loops may be done zero times (when $r = 0$ or $r = S$).

5) Randomly select the next state $\boldsymbol{y}_{t+1}$ from among $\boldsymbol{z}^{(-(S-r))}, \ldots, \boldsymbol{z}^{(-1)}, \boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)} \ldots, \boldsymbol{z}^{(r)}$, where each state, $\boldsymbol{z}^{(k)}$, has probability proportional to $\pi(\boldsymbol{z}^{(k)})|\alpha|^k$.

The algorithm has two parameters: S, the number of new states from among which we chose the next state, and the interval $[a, b]$ in which α lies. The range for α controls the degree of overrelaxation, and S controls the distance the chain moves. In Section 4.3, we derive a relationship between good values for S and α. Therefore the algorithm will require the choice of only one parameter.

To illustrate how the algorithm suppresses random walks we will consider a toy example. Although the algorithm is intended mainly for non-Gaussian distributions, for illustrative purposes we consider a bivariate Gaussian distribution. In Figure 4.1 (a) it is shown how in one Jacobian overrelaxation step, the chain can move to a state in a distant part of the distribution. Figure 4.1 shows the probability of choosing each of the $S+1$ points on the trajectory.

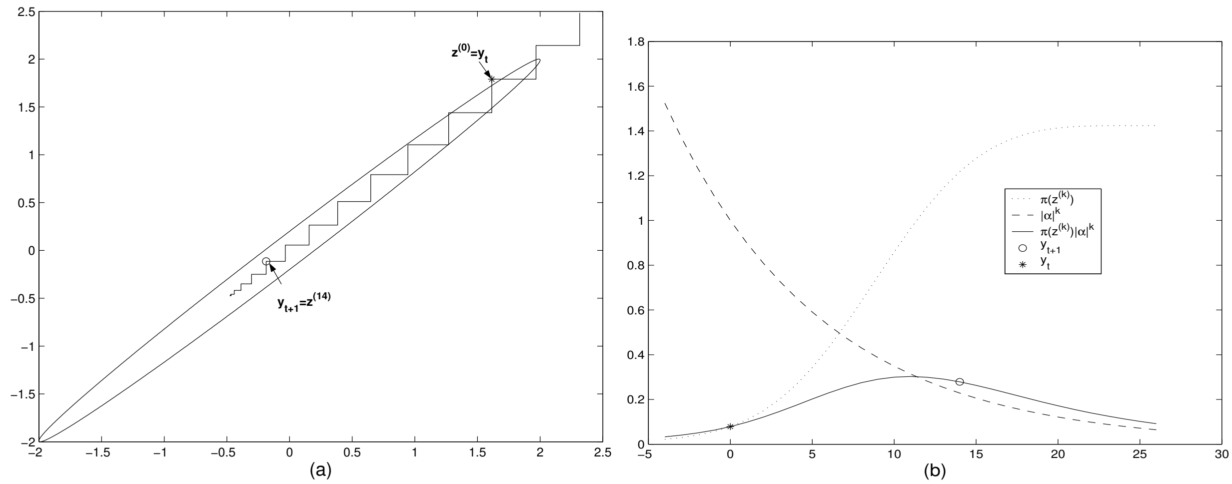


Figure 4.1: **(a)** The path shows the candidate states generated by Jacobian overrelaxation applied to a bivariate Gaussian distribution with correlation 0.995. The parameters used by the Jacobian Overrelaxation are $S = 30$ and $\alpha = -0.9$. In this iteration, $r = 26$ updates were done with α and the first component updated with α was $j = 1$ (here z_1 is on the vertical axis) The point chosen to be the next state, $\boldsymbol{y_{t+1}}$, is 14 steps away from $\boldsymbol{y_t}$. The ellipse represents the two standard deviation probability contour. **(b)** The plot shows the density of the points on the trajectory in part (a) (dotted line), the powers of $|\alpha|$ (dashed line) and the probabilities for each point obtained as a product of the density and the corresponding power of $|\alpha|$ (solid line), with k on the horizontal axis.

4.2 Validity of the Method

The algorithm is valid if the Markov chain leaves the desired distribution *invariant* and is *ergodic*. The invariance can be proved using Theorem 4.1 of Liu and Wu (1999) which can be paraphrased as follows:

Theorem 4.1. *Suppose* (i) *the random variable* $v \sim \nu(v)$*;* (ii) $\{t_r : r \in \mathcal{A}\}$ *is a set of one-to-one differentiable transformations on* v*;* (iii) *a probability measure* $p_0(r)$ *is defined on* $\mathcal{A}$ (iv) $J_r(w) = \det\{\partial t_r(w)/\partial w\}$*. Let* r_0 *be a random draw from* $p_0(r)$ *and let* $w = t_{r_0}^{-1}(v)$*. If*

$$r_1 \sim \nu(r|w) \propto \nu(t_r(w))\,|J_r(w)|\,p_0(r) \tag{4.3}$$

then $v' = t_{r_1}(w)$ *follows the distribution* ν*.*

The above theorem holds in our case with the following choices of parameters: $\mathcal{A} = \{0, 1, \ldots, S\}$, $p_0(r)$ is the uniform distribution on $\mathcal{A}$, $\boldsymbol{v} = (\boldsymbol{y}_t, j)$, where $\boldsymbol{y}_t$ is the state of the chain at time t and j is the component chosen to be first updated with α. The distribution ν is the joint distribution of $\boldsymbol{y}$ and j, where $\boldsymbol{y}$ is distributed according to π and j is uniform on $\{0, 1, \ldots, N-1\}$, independent of $\boldsymbol{y}$, as it is established in step (2) of the algorithm. The number of steps, r, the chain is updated with α is uniformly drawn from $\{0, 1, \ldots, S\}$. Therefore the number of steps, $r_0 = S - r$, used to update the chain with $1/\alpha$ is also uniformly distributed on $\{0, 1, \ldots, S\}$. The state $\boldsymbol{w}$ is obtained by applying formula (4.2) $S - r$ times, so that $\boldsymbol{w} = t_{S-r}^{-1}(\boldsymbol{v}) = \left(\boldsymbol{z}^{(-(S-r))}, (j - (S-r)) \pmod N\right)$. Choosing r_1, the number of steps to move forward starting from $\boldsymbol{w}$ and using an update with α is equivalent to choosing among:

$t_0(\boldsymbol{w}) = \left(\boldsymbol{z}^{(-(S-r))},\ (j - (S-r)) \pmod N\right),$

$\vdots$

$t_{S-r-1}(\boldsymbol{w}) = \left(\boldsymbol{z}^{(-1)},\ (j-1) \pmod N\right),$

$$t_{S-r}(\boldsymbol{w}) = \left(\boldsymbol{z}^{(0)},\ j \pmod N\right),$$
$$t_{S-r+1}(\boldsymbol{w}) = \left(\boldsymbol{z}^{(1)},\ (j+1) \pmod N\right),$$
$$\vdots$$
$$t_S(\boldsymbol{w}) = \left(\boldsymbol{z}^{(r)},\ (j+r) \pmod N\right)$$

with probabilities proportional to $\nu(t_k(\boldsymbol{w}))\,|J_k(\boldsymbol{w})| \cdot (1/S)$ where $J_k(\boldsymbol{w}) = |\alpha|^k$.

The state chosen is $\boldsymbol{v'} = \left(\boldsymbol{z}^{(-(S-r)+r_1)},\ (j-(S-r)+r_1) \pmod N\right) = (\boldsymbol{y_{t+1}},\ (j-(S-r)+r_1) \pmod N)$ and follows the distribution ν. Dropping the second component of $\boldsymbol{v'}$, we obtain that the marginal distribution of $\boldsymbol{y}_{t+1}$ is distributed according to π, and therefore π is left invariant by the JO updates.

The algorithm chooses α at random at each step to ensure ergodicity. If α is fixed, there are cases for which the chain produced by Jacobian overrelaxation will not be ergodic. For a univariate distribution or a multivariate distribution with independent components the conditional mean is the same for a particular component, so if α is kept constant, the chain could move only through a countable number of states. We explore the ergodicity of the Markov chain produced by the Jacobian overrelaxation algorithm for zero mean Gaussian distributions with independent components, very low correlation between components, and strong positive correlation between components.

For a bivariate distribution with independent components and $\alpha < 0$, the chain never reaches points on the main axes, because once it gets there, it will not move away. However, since the chain gets close to the axes, the chain is ergodic as long as it is not started on the main axes as we can see in Figure 4.2.

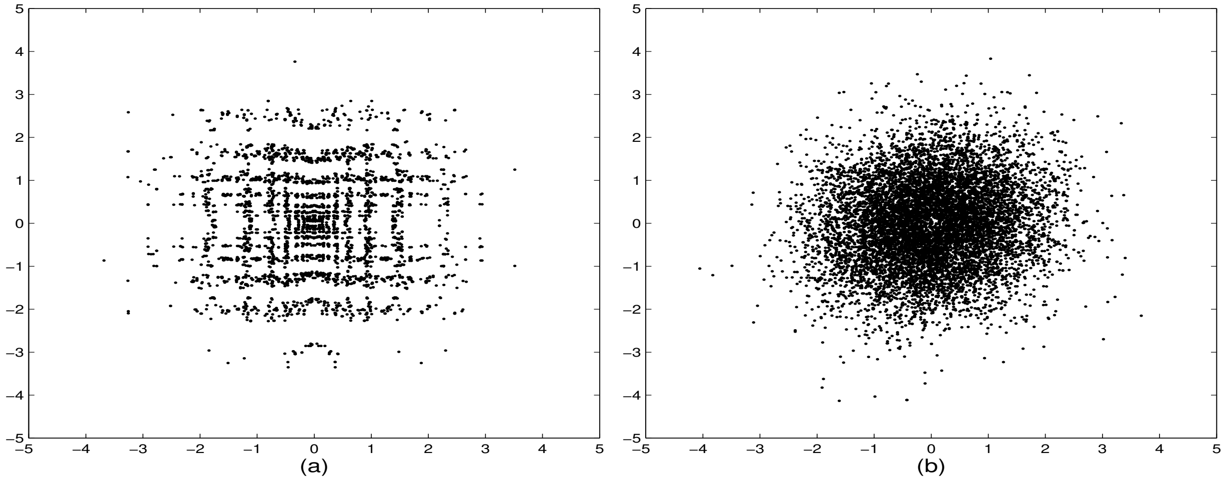
In Figures 4.2 through 4.4, we started with a bivariate independent Gaussian distribution and increased the correlation between the components to 0.1, keeping the same interval for α . We can see that as we move away from the independent case, the chain explores the space more efficiently. We have to note that the choice of the interval for α

is not optimal for these correlations, but it was chosen only for illustration purposes.

We investigate the bivariate Gaussian with independent components with α closer to -1 and $S = 10$. This situation is presented in Figure 4.6 (a)-(d). The chain explores the state space very slowly and it is symmetric with respect to both main axes up to iterations $100,000$ (see (a) and (b)). As the chain moves above $100,000$ iterations, the chain gets closer to the $y-$axis and it spends more time there before it moves towards the $x-$axis.

The case of strongly positively correlated components, for which this method is intended, is presented in Figure 4.7. The chain explores the state space very efficiently.

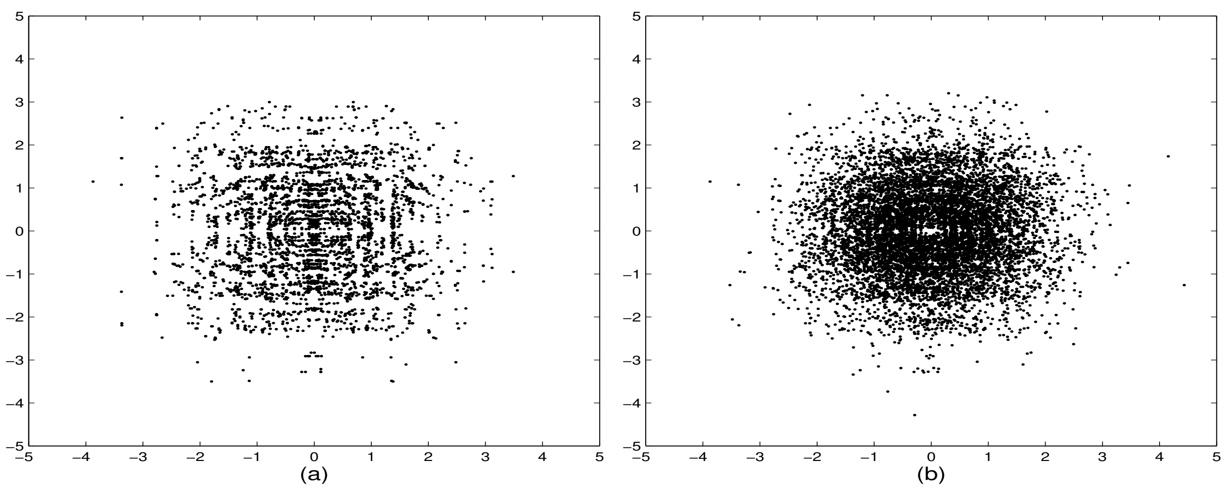
For Gaussian distributions with independent components, performing Jacobian overrelaxation with $\alpha > 0$ will not produce an ergodic chain as the chain stays in the same quadrant as the starting point. This can be easily seen as the conditional means are the two main axes. If the first component of the vector that we are interested in updating is negative, the first component of the next state will be the first component of the previous state multiplied with α, which will also be negative. For $\rho = 0.01$ and $\alpha > 0$, we can see in Figure 4.5 that the chain is ergodic, although it does not sample efficiently, spending a longer time around the main axes. However, we are not interested in using this algorithm with $\alpha > 0$.



10,000 iterations.

There are 100,000 iterations, but we kept only every $10th$ iteration.

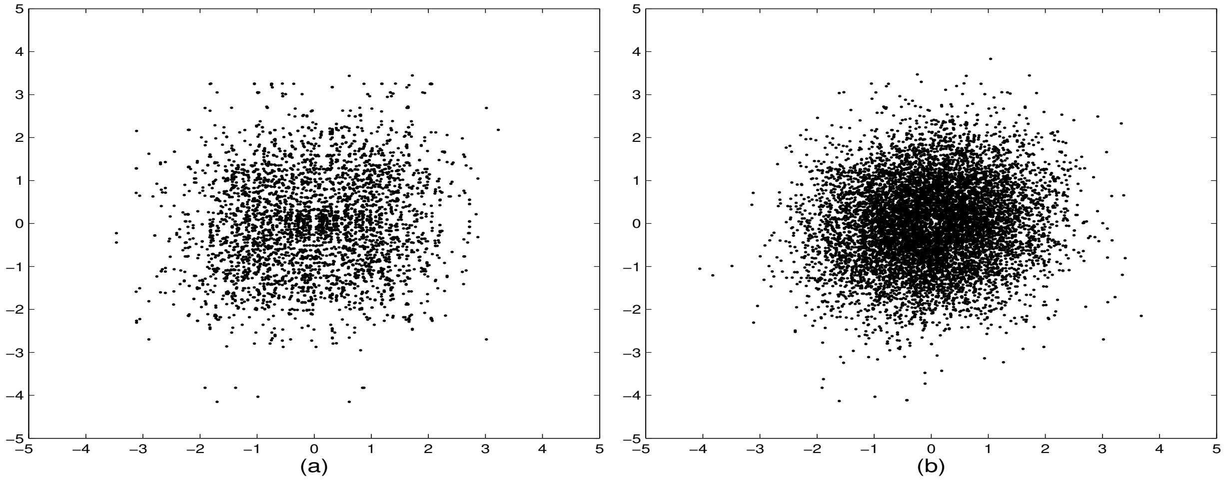
Figure 4.2: $\rho = 0$, $S = 1$, $\alpha \in (-0.81, -0.79)$



10,000 iterations.

There are 100,000 iterations, but we kept only every $10th$ iteration.

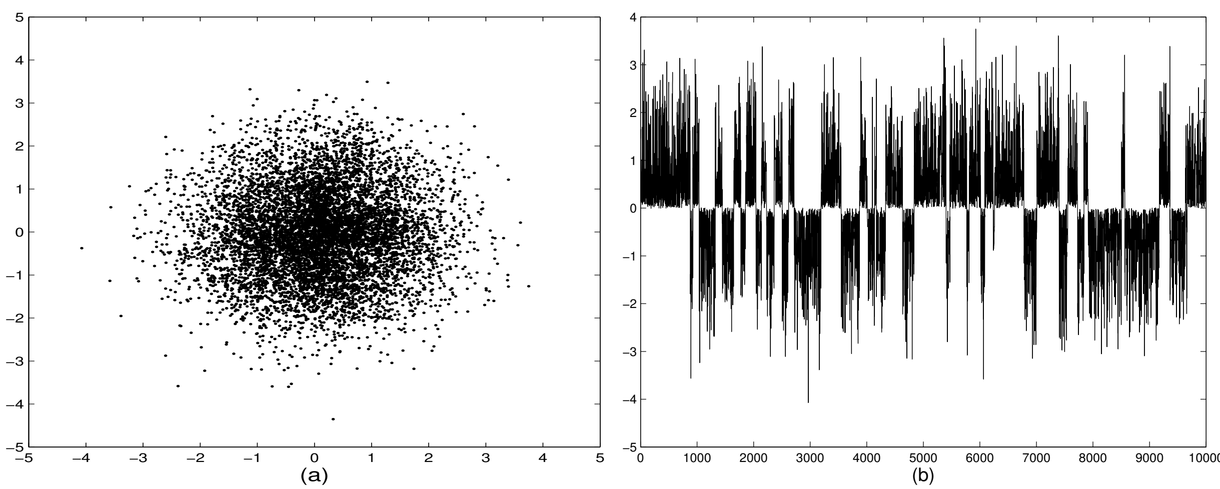
Figure 4.3: $\rho = 0.01$, $S = 1$, $\alpha \in (-0.81, -0.79)$



10,000 iterations.

There are 100,000 iterations, but we kept only every $10th$ iteration.

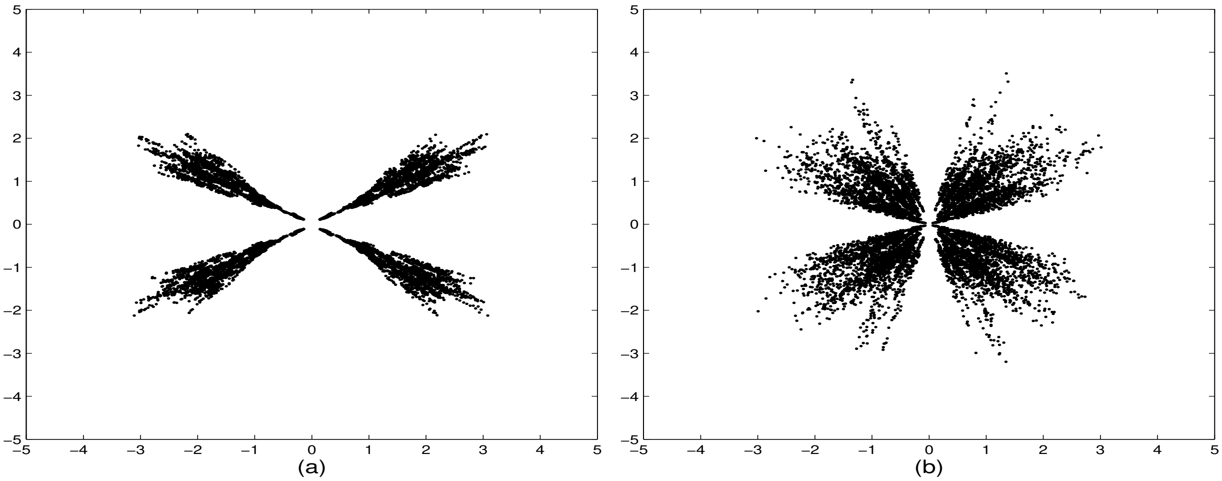
Figure 4.4: $\rho = 0.1$, $S = 1$, $\alpha \in (-0.81, -0.79)$



There are 100,000 iterations, but we kept only every $10th$ iteration.

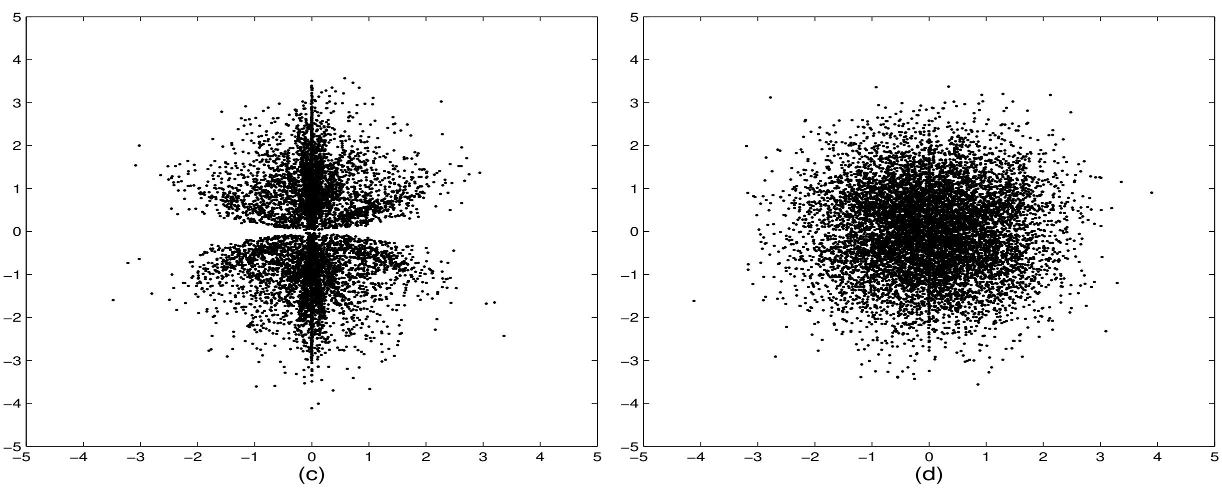
There are 100,000 iterations, but we kept only every $10th$ - First component of a).

Figure 4.5: $\rho = 0.01$, $S = 5$, $\alpha \in (0.05, 0.15)$



$10,000$ iterations.

There are $100,000$ iterations, but we kept only every $10th$ iteration.



There are $1,000,000$ iterations, but we kept only every $100th$ iteration.

There are $10,000,000$ iterations, but we kept only every $1000th$ iteration.

Figure 4.6: $\rho = 0$, $S = 10$ $\alpha \in (-0.995, -0.985)$

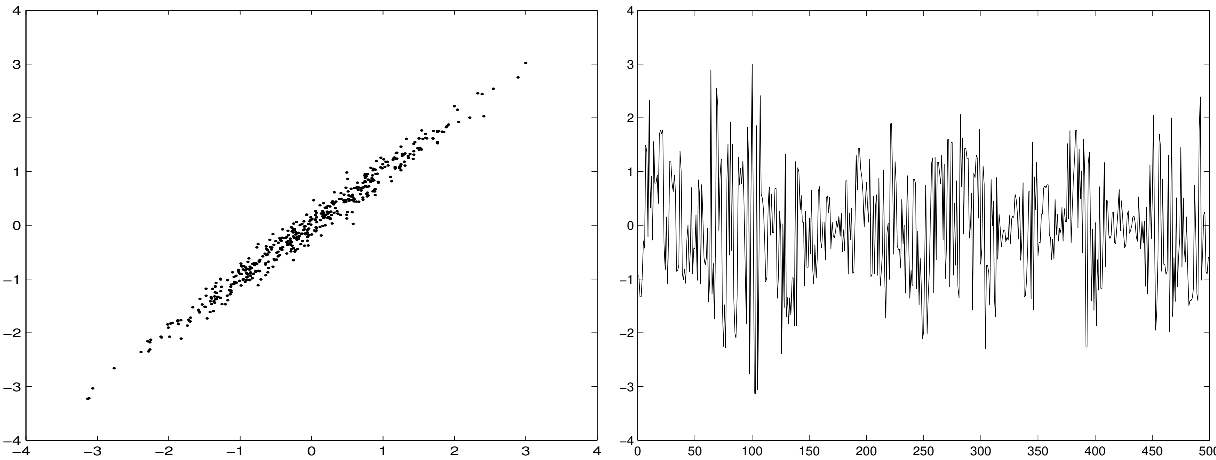


Figure 4.7: **(a)** 500 Jacobian overrelaxation iterations for a bivariate Gaussian distribution with correlation between the components of $\rho = 0.99$ (the square root of the ratio of the largest to the smallest eigenvalue is approximately 14). The parameter used in the Jacobian overrelaxation method are $S = 40$ and $\alpha \in (-0.97, -0.95)$ **(b)** The first component of the chain. The autocorrelations for the first component are close to 0 at lag 3. The autocorrelations for the square of the first component are close to 0 at lag 15.

The following theorem states the conditions under which we prove that for a bivariate distribution it is possible to move from any state to any other state using Jacobian overrelaxation updates. Note that in what follows we talk about approximate conditional mean functions, but they can instead be the functions giving the conditional mode or any other measure of location for the distribution of one component conditioned on the others.

Theorem 4.2. *Suppose the bivariate distribution with non-independent components, π, is such that $\pi(\boldsymbol{x}) > 0$, the approximate conditional mean functions are continuous everywhere except a set of measure 0, and the number of regions generated by the approximate* *conditional mean functions is finite within a bounded region. It is possible to move from any state to any other state using Jacobian overrelaxation type updates with* $\alpha \in (-a, -b)$ *or* $\alpha \in (b, a)$ *with* $a, b > 0$*, except for a set of points of* π *probability* 0*. The set of points of* π *probability* 0 *consists of the intersection points of the conditional mean functions.*

For an independent bivariate distribution, Jacobian overrelaxation with $\alpha > 0$ *does not produce a chain that explores all states of the space. For independent bivariate distribution Jacobian overrelaxation with* $\alpha < 0$*, the conditional mean points cannot be reached.*

Proof. For the special situation when the bivariate distributions has independent components by performing Jacobian overrelaxation with $\alpha > 0$ will not explore states from the entire space since the chain can move anywhere in the region of the starting point, but it cannot cross the conditional mean functions and move in another region.

For bivariate distributions with independent components, Jacobian overrelaxation with $\alpha < 0$, it is not possible to move to the points that lie on the approximate conditional mean lines because if the chain reaches states on the approximate conditional mean lines it cannot move away from there. For the same reason, for any distribution, it is not possible to move to points found at the intersection of the approximate conditional mean functions.

The proof is done for simplicity with $S = 1$, but it is valid for $S > 1$, since there is a non-zero probability of moving to the same states as in the case $S = 1$.

The two approximate conditional mean functions divide up the space into regions. We need to show that it is possible to move from any point in a region to any other point in the same region and to prove that it is possible to move from a point in a region to a point in any another region.

Note that we can move from $\boldsymbol{X}_0$ to any point since $\pi(x) > 0$.

First we want to prove that we can get from any point $\boldsymbol{X} = \boldsymbol{X}_0 = (X_0^{(1)}, X_0^{(2)})$ to any point $\boldsymbol{Y}$ that are in the same region as determined by the approximate conditional mean functions. We will show how to do this by performing two overrelaxation updates. Assume the updates are done for the first component of the two dimensional vector and that $X_0^{(1)} = \mu_1 + d$ where $d > 0$ and μ_1 is the approximate conditional mean or conditional mode or some other measurement of location for the distribution of the first component conditioned on the second. We prove that from $X_0^{(1)}$ it is possible to move to any point in (μ_1, ∞).

We show that it is possible to move from $X_0^{(1)}$ to any point in $(\mu_1 + (b/a)d, \mu_1 + (a/b)d)$ in two overrelaxation updates. We have to prove that for any point $X_2^{(1)} \in (\mu_1 + (b/a)d, \mu_1 + (a/b)d)$, there exist $\alpha_1,\ \alpha_2 \in (-a, -b) \cup (-1/b, -1/a)$ or $\alpha_1,\ \alpha_2 \in (b, a) \cup (1/a, 1/b)$ if $\alpha > 0$, such that:

$$X_1^{(1)} = \mu_1 + \alpha_1 (X_0^{(1)} - \mu_1) \tag{4.4}$$

and

$$\begin{aligned} X_2^{(1)} &= \mu_1 + \alpha_2 (X_1^{(1)} - \mu_1) && (4.5) \\ &= \mu_1 + \alpha_2 \Big(\mu_1 + \alpha_1 (X_0^{(1)} - \mu_1) - \mu_1 \Big) && (4.6) \\ &= \mu_1 + \alpha_1 \alpha_2 (X_0^{(1)} - \mu_1) && (4.7) \end{aligned}$$

This is equivalent to showing that there exist $\alpha_1,\ \alpha_2 \in (-a, -b) \cup (-1/b, -1/a)$ or $\alpha_1,\ \alpha_2 \in (b, a) \cup (1/a, 1/b)$ if $\alpha > 0$ such that

$$\alpha_1 \alpha_2 = \frac{(X_2^{(1)} - \mu_1)}{(X_0^{(1)} - \mu_1)} \tag{4.8}$$

We know that $X_0^{(1)} - \mu_1 = d$ and $X_2^{(1)}$ is such that $X_2^{(1)} - \mu_1 \in ((b/a)d, (a/b)d)$. Therefore if $\alpha < 0$ we have to show that for any $p \in (b/a, a/b)$ there exist $\alpha_1,\ \alpha_2 \in (-a, -b) \cup (-1/b, -1/a)$ such that $\alpha_1 \alpha_2 = p$. We can choose any $\alpha_1 \in (-a, -b)$ or $\alpha_1 \in (-1/b, -1/a)$, and then let $\alpha_2 = p/\alpha_1$, which will be in the desired range. To see this, assume $\alpha_1 \in (-a, -b)$, therefore $1/\alpha_1 \in (-1/b, -1/a)$ and combined with the fact that $p \in (b/a, a/b)$ we obtain that $\alpha_2 = p(1/\alpha_1) \in (-1/b, -1/a)$. Similarly if $\alpha_1 \in (-1/b, -1/a)$, $\alpha_2 = p/\alpha_1 \in (-a, -b)$.

Similarly if $\alpha > 0$, we can choose $\alpha_1 \in (b, a)$ and let $\alpha_2 = p/\alpha_1$, which will be in the desired range because $p \in (b/a, a/b)$.

Therefore we found an interval around $X_0^{(1)}$ such that there is a non-zero probability density of moving from $X_0^{(1)}$ to any point in this interval. The distance from the lower end of the interval to the conditional mean after two overrelaxation updates is $(b/a)d$ and after $2n$ overrelaxation updates it is $(b/a)^n d$. Therefore as the number of steps increases we can move to any point arbitrarily close to the conditional mean. Similarly we can prove that we can move to any point in $(\mu_1 + d, \infty)$, since after $2n$ overrelaxation type updates we can move to a distance $(a/b)^n d$ ($(a/b) > 1$), which converges to ∞ as n increases.

A similar proof is valid when the starting point is on the other side of the conditional mean, *i.e.* $X_0^{(1)} = \mu_1 - d$, $d > 0$.

Therefore it is possible to move to any point on the same side of the approximate conditional mean as our starting point, on the horizontal axis that passes through the starting point. Similarly we can move to any point on the vertical axis that is on same side of the conditional mean as the starting point. Therefore by performing successive horizontal and vertical updates we can move from any point $\boldsymbol{X}$ in a region to any point $\boldsymbol{Y}$ in the same region.

The conditional mean functions are continuous than the requirement that the approximate conditional mean functions of being continuous everywhere except a set of measure 0 will be met. Because the approximate conditional mean functions are continuous almost everywhere, the approximate conditional mean function with respect to which we perform the horizontal updates is not parallel with the x-axis and similarly the approximate conditional mean function with respect to which we do the vertical updates is not parallel to the y-axis. Therefore for any state we can perform updates both on the horizontal and vertical directions.

We can move from a region to another region if the two regions have a non-zero length boundary. Then with $\alpha < 0$ we can either flip on the other side of one of the approximate conditional means or by using the other approximate conditional mean we move through this conditional mean until we obtain a state in the new region. If the boundary between the two region has non-zero length and the boundary consists of the two conditional mean functions which coincide for the boundary, we can move in the adjacent region by flipping on the other side of the conditional mean either vertically or horizontally. If the boundary between the two regions, A and B has zero length (either they are not neighbouring boundaries or they have only one point in common, the intersection of the conditional mean functions) then as described before we have to move from A to an adjacent region, R_1 with which it has non-zero length boundaries. We can move from R_1 to B if the two regions are adjacent and have a boundary of non-zero length. If that is not the case the chain has to pass through a series of regions $R_1, \ldots, R_n$ such that two consecutive regions have a boundary of non-zero length and R_n has a boundary of zero length with B. A sequence of such region $R_1, \ldots R_n$ exists. Indeed, assume A and B are enclosed a bounded section and we added all the regions that have a non-zero length boundaries with A and all the regions that have non-zero length boundaries with the new regions and so on. Assume there is no region that has a non-zero length boundary with B, after adding all the regions enclosed in the bounded section. However this is not possible, as the last regions added have to have a non-zero length boundary with some other region and therefore we did not add all the possible regions. Once we moved in the region B, as proved at the previous step, it is possible to move to any state in the same region.

For $\alpha > 0$ it is essential as mentioned before that the conditional means are not the horizontal and the vertical lines. To move from a region A to a region B with $\alpha > 0$, the two region need to have a non-zero length boundary or the chain has to move through a series of neighbouring regions with non-zero boundaries to move to a state in B. Therefore it is sufficient to show that we can move between two region that have a non-zero length boundary. Let us denote the boundary conditional mean with C_1 and let us assume that it corresponds to vertical updates, and the other conditional mean function with C_2 for horizontal updates. We cannot use C_1 directly to move in the other region as we cannot flip on the other side of the conditional mean. We need instead to get to a point in region A for which the horizontal line that passes through this point and it is on the same side of the conditional mean, C_2, contains also points in region B. Therefore, because it is possible to move to any point on this line, it is possible to move to a point in region B. If we obtain a point in region B from this point it is possible to move to the desired point $\boldsymbol{Y}$, as was previously shown. $\square$

4.3 How to choose the parameters for Jacobian Overrelaxation

In this section we derive a relationship between good values of α and S, such that only one parameter of the method need be chosen. The relationship between the parameters should be such that the ratio of the probability of staying in the initial state to the probability of moving k steps away by updating with α or $1/\alpha$ is such that we are reasonably likely to move k steps away.

The relationship between α and S is first derived for a univariate Gaussian distribution. Later in this section we will argue how a similar relationship between α and S holds for a bivariate Gaussian distribution with strongly correlated components.

Consider the univariate Gaussian distribution $N(\mu, \sigma^2)$ and assume the chain at time t is in the state $y_t = z^{(0)}$. By performing k overrelaxation type updates with α we obtain the following states:

$z^{(1)} = \mu + \alpha(z^{(0)} - \mu)$

$z^{(2)} = \mu + \alpha(z^{(1)} - \mu) = \mu + \alpha^2(z^{(0)} - \mu)$

$\vdots$

$z^{(k)} = \mu + \alpha(z^{(k-1)} - \mu) = \mu + \alpha^k(z^{(0)} - \mu)$

The probabilities of choosing the states $z^{(0)}, z^{(1)}, \ldots, z^{(k)}$ are proportional to $\pi(z^{(0)})$, $\pi(z^{(1)})|\alpha|$,...,$\pi(z^{(k)})|\alpha|^k$. The ratio between the probability of choosing state k and the probability of staying in the same state, $z^{(0)}$ is:

$$\frac{P(\text{of moving to state } z^{(k)})}{P(\text{of staying in state } z^{(0)})} = \frac{\pi(z^{(k)})|\alpha|^k}{\pi(z^{(0)})} \tag{4.9}$$

$$= |\alpha|^k \left\{ e^{-\alpha^{2k}(z^{(0)}-\mu)^2/2\sigma^2} \right\} \Big/ \left\{ e^{-(z^{(0)}-\mu)^2/2\sigma^2} \right\} \tag{4.10}$$

$$= \quad |\alpha|^k e^{-(z^{(0)}-\mu)^2(\alpha^{2k}-1)/2\sigma^2} \tag{4.11}$$

If this ratio is too small the state k steps away will very rarely be accepted and we therefore performed too many overrelaxation type steps. On the other hand, if the ratio is close to 1, we performed too few overrelaxation type steps. If in fact this was the right amount of suppression of random walks we could have used an α further from -1, and moved to a different contour line. An appropriate ratio might be around $c = 0.1$. Therefore by setting the ratio of the probabilities above to be equal to c we obtain:

$$|\alpha|^k e^{-\left((z^{(0)}-\mu)^2(\alpha^{2k}-1)\right)/(2\sigma^2)} = c \tag{4.12}$$

If we take the logarithm of equation (4.12) and let $A = -(z^{(0)} - \mu)^2/(2\sigma^2)$, we obtain the following equation in α and k:

$$A(\alpha^{2k} - 1) + k \log |\alpha| - \log(c) = 0 \tag{4.13}$$

Let us write the relationship between α and k in the form $\alpha = -e^{-B/k}$. Plugging this in (4.13), we obtain the following equation in B:

$$g(B) = A(e^{-2B} - 1) - B - \log(c) = 0 \tag{4.14}$$

which does not involve k. We studied the above function and found that it achieves its maximum for $B_{max} = -1/2 \log(-1/2A)$ and that the function g is increasing for B in $(-\infty, B_{max})$ and decreasing for B in (B_{max}, ∞). Also, the limit of g as B goes to $\pm\infty$ is $-\infty$. Therefore the function g has two zeros. If updates with $1/\alpha$ are done and we denote $\beta = 1/\alpha$ we obtain the same equation as (4.13) in β. Therefore the positive zero of the function g correspond to updates done with α and the negative zero to updates done with $1/\alpha$.

The solutions of equation (4.14) depend on A, which is a function of the starting point, and on c, which is the ratio of the probability of staying in the initial state to the probability of moving to the state k steps away. To investigate the possible solutions for equation (4.14), we consider the range of values A and c can take. If $z^{(0)}$ is distributed as $N(\mu, \sigma)$, then $(z^{(0)} - \mu)^2/\sigma^2$ is χ^2_1. The 5% and 95% quantiles of χ^2_1 are 0.0039 and 3.84. The values of A corresponding to these quantiles are -0.00196 and -1.92. In the table 4.1 we present the solutions for the equation (4.14) for some combinations of A and c. The values for $|B|$ are important because $B > 0$ corresponds to updates done with α and $B < 0$ corresponds to updates done with $1/\alpha$, but once we choose a B we do updates both with α and $1/\alpha$. The values for $|B|$ are between 0.43 and 6.61. For starting points close to the mode, corresponding to $A = -0.002$ in Table 4.1, the values of the two solutions are close in magnitude; for those starting points the same relationship between α and k holds for updates with α and $1/\alpha$. For starting points far away from the mode, corresponding to $A = -2$ in the Table 4.1, for updates with α, the relationship between α and the number of steps requires values of B that are similar to those when the starting points are close to the mode. This is not the case for the updates done with $1/\alpha$, for which the relationship between α and k is based on very different values of B. This is due to the fact that when the starting point is far away from the mode, the chain has to move to states closer to the mode and therefore we have values of $B = -0.43$ that corresponds to values for $1/\alpha$ closer to 1 that allow the chain to move closer to the mode or at least not go further away from the mode.

In the bivariate Gaussian case with strongly-correlated components, if the components have different variances, after rescaling them the probability contour lines are nearly parallel to the line at 45° angle and the two conditional mean lines nearly coincide with the 45° line. Assume we are in the state $\boldsymbol{z}^{(k)} = (z_0^{(k)}, z_1^{(k)})$ and the first component is to be updated next. If $z_0^{(k)}$ is at a distance d from the conditional mean, after an update with α, $z_0^{(k+1)}$ is on the other side of the conditional mean at a distance $|\alpha|d$. The next

A	c	B_1	B_2	α_1	α_2
-0.002	0.1	2.30	-4.03	-0.977	-0.961
-0.002	0.01	4.61	-4.19	-0.955	-0.959
-2	0.1	4.30	-0.43	-0.958	-0.996
-2	0.01	6.61	-0.64	-0.936	-0.994

Table 4.1: Values that B can take depending on the starting point ($A = -(z^{(0)} - \mu)/(2\sigma^2)$) and on c, the ratio of the probability of choosing $z^{(k)}$, to the probability of choosing $z^{(0)}$ as the next state. The values of α are for $S = 100$.

component to be updated, $z_1^{(k+1)} = z_1^{(k)}$ is at the same distance $|\alpha|d$ from its conditional mean because the conditional mean is almost along the 45° line. Therefore $z_1^{(k+2)}$, obtained by one update with α, is at a distance $\alpha^2 d$ from the conditional mean. Therefore in two overrelaxation type steps we moved the same distance from the conditional mean, we would have moved in two overrelaxation steps in the univariate Gaussian case. If we take into consideration that α has magnitude around 1, than the distance we move along the diagonal in one step is $2d/\sqrt{2}$. For a N dimensional problem the distance moved in one step is $2d/\sqrt{N}$ along the long diagonal. The difficulty of the problem is measured by the square root of the ratio of the largest to the smallest eigenvalue, $\sqrt{\lambda_{max}/\lambda_{min}}$. The distance, d, is approximately equal to $\sqrt{\lambda_{min}}$ and therefore if we want to move from minus two standard deviations away from the mean to two standard deviations away it is equivalent with moving a distance $4\sqrt{\lambda_{max}}$. The number of overrelaxation steps needed to move this distance is proportional to $2\sqrt{\lambda_{max}/\lambda_{min}}\sqrt{N}$.

The value for B would ideally be chosen based on the performance of the estimator, measured by the autocorrelation time, as described in Section 2.1. Therefore we will run an experiment for a bivariate Gaussian distribution with correlation $\rho = 0.999$ for which we will estimate the autocorrelation time, τ, for different values of B. We will fix the number of steps, S, to twice the square root of the ratio of the largest to the smallest eigenvalue, and we vary B from 0.5 to 10 with step 0.5. For each value of B, α will be in the interval $\left(-e^{-(B+0.1)/S}, -e^{-(B-0.1)/S}\right)$. For each B we generate a chain of length 10000 and we calculate the autocorrelation time, τ, for the first component of the chain and for the first component squared. The lag at which the autocorrelations are close to zero for most values of B is 200 and the autocorrelation times are obtained by summing the autocorrelations up to this lag. The autocorrelation times as a function of B for the two functions of state are plotted in Figure 4.8. The best B for the first component of the state based on our experiment is 1 or less. This is expected since for linear functions of state the optimal B would correspond to an α close to -1, corresponding to B being close to 0. The best B for the square of the first component is around 3.5, but any value between 2 and 8, corresponding to α between -0.98 and -0.92, will give a close-to-optimal value for the autocorrelation times.

The relationship between α and S is different for the Gaussian distribution case with independent components than the relationship for Gaussian distribution with strong correlated components. Consider the bivariate Gaussian distribution with independent components and assume the mean of each component is zero, and hence the conditional means are always zero. If we start in the state $\boldsymbol{z}^{(0)} = (z_0^{(0)}, z_1^{(0)})$, and begin by updating the first component, after k overrelaxation type updates we have:

$$\boldsymbol{z}^{(k)} = \begin{cases} (\alpha^{k/2} z_0^{(0)}, \alpha^{k/2} z_1^{(0)}) & \text{if } k \text{ is even} \\ (\alpha^{(k+1)/2} z_0^{(0)}, \alpha^{(k-1)/2} z_1^{(0)}) & \text{if } k \text{ is odd} \end{cases} \tag{4.15}$$

As we need only an approximation for the relationship between α and k we can consider

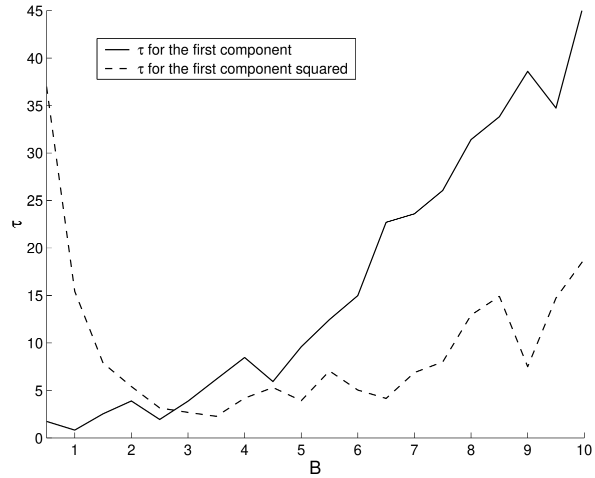


Figure 4.8: The autocorrelation times versus B are plotted for two functions of the components: the first component of the states (solid line) and for the square of the first component of the states (dashed line). The number of steps, S is fixed to 90 (twice the square root of the ratio of the largest to smallest eigenvalue) and the relationship between α and S is given by $\alpha = -\exp(-B/S)$.

that $\boldsymbol{z}^{(k)} = (|\alpha|^{k/2} z_0^{(0)}, |\alpha|^{k/2} z_1^{(0)})$. Therefore,

$$\frac{\pi(\boldsymbol{z}^{(k)})}{\pi(\boldsymbol{z}^{(0)})} = \left\{ e^{-\alpha^k \left((z_0^{(0)})^2 + (z_1^{(0)})^2 \right)/2} \right\} \Big/ \left\{ e^{-\left((z_0^{(0)})^2 + (z_1^{(0)})^2 \right)/2} \right\} \tag{4.16}$$

$$= e^{-\left((z_0^{(0)})^2 + (z_1^{(0)})^2 \right)(\alpha^k - 1)/2} \tag{4.17}$$

and

$$\frac{P(\text{of moving to state } \boldsymbol{z}^{(k)})}{P(\text{of staying in state } \boldsymbol{z}^{(0)})} = \frac{\pi(\boldsymbol{z}^{(k)})|\alpha|^k}{\pi(\boldsymbol{z}^{(0)})} \tag{4.18}$$

$$= |\alpha|^k e^{-\left((z_0^{(0)})^2 + (z_1^{(0)})^2 \right)(\alpha^k - 1)/2} \tag{4.19}$$

Taking the logarithm of the equation below

$$|\alpha|^k e^{\frac{-\left((z_0^{(0)})^2 + (z_1^{(0)})^2 \right)(\alpha^k - 1)}{2}} = c \tag{4.20}$$

and letting $A = -\left((z_0^{(0)})^2 + (z_1^{(0)})^2\right)/2$, we obtain the following equation in α and k:

$$A(\alpha^k - 1) + k \log|\alpha| - \log(c) = 0 \tag{4.21}$$

Note that this equation is very similar to the equation (4.13). To investigate the relationship between α and k we will proceed as in the Gaussian univariate case by assuming the relationship between α and k is of the form $\alpha = -e^{-B/k}$. Plugging this in (4.21) we obtain the following equation in B:

$$g(B) = A((-1)^k e^{-B} - 1) - B - \log(c) = 0 \tag{4.22}$$

Because $(z_0^{(0)})^2 + (z_1^{(0)})^2$ is distributed according to a χ_2^2, we choose the 5% and 95% quantiles for χ_2^2, that correspond to values of A of -0.05 and -3 respectively. From Table 4.2 we can see that the range of values for $|B|$ for the bivariate case with independent components is between 0.69 and 7.6 and it is comparable to the range of $|B|$ for the univariate case that was from 0.43 to 6.61. For independent Gaussian distributions, as N increases, the values for $|B|$ increase. For example for $N = 100$, $A = -62$ corresponding to the 95% quantile for the χ_{100}^2 and $c = 0.1$, the corresponding $B's$ are 30.85 and -6.97.

In summary, for higher dimension distributions with strongly positive correlated components, for which JO is meant to work well, we are expecting, as we discussed earlier, that the relationship between α and S will be similar to the case for the univariate Gaussian distribution. To assess the difficulty of the problem we scale the components of the distribution such that the variances for all components are the same, obtaining a new scaled variance-covariance matrix. A good value for S is about twice the square root of the ratio of the largest eigenvalue to the smallest eigenvalue of the scaled variance-covariance matrix times the square root of the dimension of the distribution. This will ensure that the chain can move in one Jacobian overrelaxation to a distant state. This is true for Gaussian distributions, but it probably works for other distributions that are close

A	c	B_1	B_2
-0.05	0.1	2.35	-4.99
-0.05	0.01	4.65	-5.29
-3	0.1	5.29	-0.69
-3	0.01	7.60	-1.06

Table 4.2: Values that B can take depending on the starting point ($A = -\left(\left(z_0^{(0)}\right)^2 + \left(z_1^{(0)}\right)^2\right)/2$) and on c, the ratio of the probability of choosing $\boldsymbol{z}^{(k)}$, to the probability of choosing $\boldsymbol{z}^{(0)}$ as the next state.

to Gaussian. This rule can probably not be used in practice as the variance-covariance matrix is not available. The parameter α is then derived based on the relationship established, $\alpha = -e^{-B/S}$, where the value of B can be between 2 and 8. This rule of thumb is valid for bivariate Gaussian distribution with highly positively-correlated components. The validity of this recommendation is verified in the next section on a logistic regression problem.

4.4 Logistic Regression Example

In this section we evaluate the performance of Jacobian overrelaxation on a logistic regression example. We compare Jacobian overrelaxation with the Metropolis algorithm.

4.4.1 The Bayesian Logistic Regression Model

To illustrate the performance of Jacobian overrelaxation we will consider its use for a logistic regression model. We are given $(y_1, x_{11}, \ldots x_{1p}), \ldots, (y_i, x_{i1}, \ldots x_{ip})$, $\ldots, (y_n, x_{n1}, \ldots x_{np})$. The response variable, y_i, is assumed to be $Bernoulli(p_i)$ and the logistic regression model is:

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}, \text{ for all } i = 1, \ldots, n \tag{4.23}$$

In a Bayesian context we are interested in the posterior distribution of $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)'$. If we denote $\boldsymbol{x}_i = (1, x_{i1}, \ldots, x_{ip})$ then

$$p_i = P(y_i = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}_i \boldsymbol{\beta}}} \tag{4.24}$$

Denote by $\boldsymbol{X}$ the design matrix,

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1p} \\ \vdots & & & \\ 1 & x_{n1} & \ldots & x_{np} \end{pmatrix} \tag{4.25}$$

we consider the prior distribution for $\boldsymbol{\beta}$ to be $N(\boldsymbol{\mu}_\beta, \Sigma_\beta)$. The posterior distribution for $\boldsymbol{\beta}$ can be written as:

$$P(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{X}) \propto \prod_{i=1}^{n} \left(\frac{1}{1 + e^{-\boldsymbol{x}_i \boldsymbol{\beta}}}\right)^{y_i} \left(\frac{e^{-\boldsymbol{x}_i \boldsymbol{\beta}}}{1 + e^{-\boldsymbol{x}_i \boldsymbol{\beta}}}\right)^{1-y_i} e^{-(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)' \Sigma_\beta^{-1} (\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)/2} \tag{4.26}$$

The Jacobian method requires the calculation of the mode of the distribution or an approximation to it for each component of $\boldsymbol{\beta}$ conditioned on all the other components. It is more convenient to work with the energy function, E, that is defined as minus the logarithm of the joint density. The Illinois method as described in Section 4.2.5 of Thisted (1988) is used to find the minimum of the energy function by finding the zero of the derivative of the energy. The Illinois method is a bracketing method that requires two starting points $a < b$ such that $E'(a)E'(b) < 0$, implying that E' has a zero in the interval $[a, b]$. To locate the starting points a and b, we start by evaluating the first derivative of the energy at the mode of the prior distribution. If the first derivative is positive, the minimum is found at the left of this point; if it is negative the minimum is found at the right of this point. We step left or right using a step size, ψ, of magnitude 1, and doubling the step size each time until we find two points a and b for which the signs of the first derivative evaluated at these points are opposite. We stop the Illinois algorithm when two consecutive points are within a distance less than 0.01. To reduce the computational effort we have experimented by increasing this distance to 0.1, but then, due to the fact that the mode is very inaccurately determined, the chain obtained by the Jacobian overrelaxation explores the space less efficiently.

The energy function for the logistic problem considered above is:

$$
\begin{aligned}
E(\boldsymbol{\beta}) &\propto -\log P\left(\boldsymbol{\beta}|\boldsymbol{y},\boldsymbol{X}\right) && (4.27)\\
&= \log\left(\prod_{i=1}^{n}(1+e^{-\boldsymbol{x_i}\boldsymbol{\beta}})\right) - \log\left(e^{-\sum_{i=1}^{n}\boldsymbol{x_i}\boldsymbol{\beta}(1-y_i)}\right) && (4.28)\\
&\quad + \frac{(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)'\Sigma_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)}{2}\\
&= \sum_{i=1}^{n}\left(\log(1+e^{-\boldsymbol{x}_i\boldsymbol{\beta}})\right) + \sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{\beta}(1-y_i) && (4.29)\\
&\quad + \frac{(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)'\Sigma_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)}{2}\\
&= \sum_{i=1}^{n}\left(\log(1+e^{-\boldsymbol{x}_i\boldsymbol{\beta}})\right) + (\boldsymbol{e}-\boldsymbol{y}')\boldsymbol{X}\boldsymbol{\beta} + \frac{(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)'\Sigma_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)}{2} && (4.30)
\end{aligned}
$$

where $\boldsymbol{e} = (1, 1, \ldots, 1)$ of dimension n. The first derivative of the energy function is:

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{\beta}} &= -\boldsymbol{X}'\left(\frac{e^{-\boldsymbol{x}_1\boldsymbol{\beta}}}{1+e^{-\boldsymbol{x}_1\boldsymbol{\beta}}}, \ldots, \frac{e^{-\boldsymbol{x}_n\boldsymbol{\beta}}}{1+e^{-\boldsymbol{x}_n\boldsymbol{\beta}}}\right) + \boldsymbol{X}'(\boldsymbol{e}-\boldsymbol{y}) + \Sigma_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta) && (4.31)\\
&= -\boldsymbol{X}'\left(\frac{1}{1+e^{\boldsymbol{x}_1\boldsymbol{\beta}}}, \ldots, \frac{1}{1+e^{\boldsymbol{x}_n\boldsymbol{\beta}}}\right) + \boldsymbol{X}'(\boldsymbol{e}-\boldsymbol{y}) + \Sigma_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta) && (4.32)
\end{aligned}
$$

The Jacobian overrelaxation method requires repeated calculation of the first derivative of the energy, both for finding the starting points a and b for the Illinois method, and for finding the minimum using this method. Only one component of $\boldsymbol{\beta}$ changes during these operations. We can take advantage of this to reduce the computation time. Assume the chain is in the state $\boldsymbol{\beta}^{old}$ with energy $E(\boldsymbol{\beta}^{old})$ and the next state $\boldsymbol{\beta}^{new}$ is the same as $\boldsymbol{\beta}^{old}$ except for the l^{th} component. The energy and the first derivative for $\boldsymbol{\beta}^{new}$ can be obtained from the energy and first derivative of $\boldsymbol{\beta}^{old}$. Using the following notation:

$$T(\boldsymbol{\beta}) = \boldsymbol{X}\boldsymbol{\beta} \tag{4.33}$$

$$V(\boldsymbol{\beta}) = (\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)'\Sigma_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta) \tag{4.34}$$

$$U(\boldsymbol{\beta}) = \Sigma_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta) \tag{4.35}$$

$$Q = \Sigma_\beta^{-1} \tag{4.36}$$

$$Q_{.,l} = \text{the } l^{th} \text{ column of } Q \tag{4.37}$$

$$\delta = \beta_l^{new} - \beta_l^{old} \tag{4.38}$$

$$\boldsymbol{X}_{.,l} = \text{the } l^{th} \text{ column of } \boldsymbol{X} \tag{4.39}$$

we can rewrite

$$T(\boldsymbol{\beta}^{new}) = T(\boldsymbol{\beta}^{old}) + \boldsymbol{X}_{.,l}\delta \tag{4.40}$$

$$V(\boldsymbol{\beta}^{new}) = V(\boldsymbol{\beta}^{old}) + \delta\left((\boldsymbol{\beta}^{old} - \boldsymbol{\mu})'Q_{.,l} + (1/2)\delta q_{ll}\right) \tag{4.41}$$

$$U(\boldsymbol{\beta}^{new}) = U(\boldsymbol{\beta}^{old}) + \delta(\sum_{i=1}^{p} q_{il}) \tag{4.42}$$

Recalculating the energy function requires $n(p+2)$ multiplications when evaluating $(\boldsymbol{e} - \boldsymbol{y})\boldsymbol{X}\boldsymbol{\beta}$. By writing the product $\boldsymbol{X}\boldsymbol{\beta}^{new}$ as a function of $\boldsymbol{X}\boldsymbol{\beta}^{old}$ as in (4.40), we can reduce the computational effort for computing $(\boldsymbol{e} - \boldsymbol{y})\boldsymbol{X}\boldsymbol{\beta}^{new}$ to n multiplications. Similarly for $(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)'\Sigma_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)$ the computational effort per evaluation of energy function can be reduced from $(p+1)(p+2)$ multiplications to just $p+1$ multiplications by using the expression (4.41). For the first derivative of the energy function, the evaluation of $\Sigma_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)$ requires $p+1$ multiplications, which can be reduced to one summation and one multiplication by using (4.42).

4.4.2 Description of the Experiment

The purpose of this experiment is to compare the Jacobian overrelaxation method with the Metropolis algorithm, and to check how the formula relating S and α, derived in Section 4.3 works for the logistic regression problem.

The data for the logistic regression model was generated as follows:

- The number of predictors p, is 15. There is also an intercept.

- The prior distribution for $\boldsymbol{\beta}$ is Gaussian with mean $\boldsymbol{\mu}_\beta = \mathbf{0}$ and $\Sigma_\beta = 1000 I_{p+1}$ (i.e., the standard deviation for each β_i is approximately 30).

- The logistic regression coefficients are:
  $\boldsymbol{b} = (1.1, 1.2, 1, 1, 1, 0.9, 1.1, 1.1, 1.4, 1.3, 1.2, 1.2, 1.5, 0.7, 0.9, 1.1)$

- The columns 1 to p of matrix $\boldsymbol{X}$ were generated randomly from a Gaussian distribution, with mean $\mathbf{0}$ and variance-covariance matrix of form:

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \vdots & & & \\ \rho & \rho & \dots & 1 \end{pmatrix} \tag{4.43}$$

  Because we are trying to produce a posterior distribution for which the components are positively correlated, we set the correlation between the predictors to be negative, specifically $\rho = -1/(p-1) + 0.0001$.

- Y_i is 1 with probability $p_i = 1/(1 + e^{-\boldsymbol{x}_i \boldsymbol{\beta}})$ and 0 with probability $1 - p_i$.

- There were $n = 100$ cases generated.

Based on a sample drawn from the Bayesian logistic regression posterior distribution using the Metropolis algorithm, we computed the sample variance-covariance matrix for this problem. Although the posterior distribution is not Gaussian, and therefore might not be of an ellipsoidal shape, the square root of the ratio of the largest to the smallest eigenvalue of the scaled sample variance-covariance matrix is still a measure of the difficulty of the problem. For this logistic regression problem the ratio is $\sqrt{\lambda_{max}/\lambda_{min}} \approx 40$. In Section 4.3 we recommended to take the number of steps, S, to be approximately $2\sqrt{\lambda_{max}/\lambda_{min}}\sqrt{N}$, where N is the dimension of the target distribution. For our logistic regression problem, S according to this recommendation should be around 320, as the dimension of our problem is 16, 15 predictors plus the intercept. We consider values for S of 100, 200, 300, 400 and 600 to determine the best value of S empirically. For each value of S, we investigate different values for α, set according to the formula $\alpha_f = -\exp(-B/S)$, where we take $B = 4$. To check the validity of the formula we move α_f closer to -1 by dividing the distance from -1 to α_f by 2 and for $S = 100$ and $S = 200$ even by 4, and we move it further from -1 by doubling the distance, obtaining this way the "middle" of the intervals. Because we do not have an idea of the width of the interval in which α should lie, we consider different lengths of the interval for each "middle" of the interval. Assume the "middle" of the interval is at a distance ϵ from -1, then we chose the ends of the intervals such that the lengths of the intervals on a logarithmic scale are 1.5, 2 and 3. Therefore the interval of "length" 1.5ϵ interval is $(-1+\epsilon/\sqrt{1.5}, -1+\epsilon\sqrt{1.5})$, of "length" 2ϵ is $(-1+\epsilon/\sqrt{2}, -1+\epsilon\sqrt{2}))$, of "length" 3ϵ is $(-1+\epsilon/\sqrt{3}, -1+\epsilon\sqrt{3})$, and for $S = 400$ and $S = 600$ we try even 4ϵ "length" interval. Table 4.3 summarizes the values of S and α that we use in our experiments. The first column in each table represents the distance from the "middle" of the interval to -1, and the rest of the columns contain the interval from which α is randomly chosen by the Jacobian algorithm. L is the logarithm of the ratio of the distances from the ends of the interval to -1. L should be the same for intervals of the same "length", no matter where the position of the "middle" of the interval is with respect to -1. We can see that not all L's in one column are the same; this is due to the fact that the ends of intervals were rounded to 3 decimal places. We rounded to 5 decimal places all ends of the intervals for $S = 600$ and the ones in the first row of the table for $S = 400$, as for these cases α_f, as derived by the formula, is closer to -1 and 3 decimal places were not enough to give us accurate intervals.

For each set of the parameters we ran a chain a number of iterations, it, such that $S \times it = 2,400,000$. We then determined τ, the autocorrelation time (as described in Section 2.1), for β_1^2, and T the CPU time per iteration.

We chose the best combination of parameters as being the ones that achieve the smallest value of $\tau \times T$. For this S and interval for α, we ran 5 more chains to assess the variability of the estimator for τ.

We also assessed the performance of the estimators by finding the lag by which the autocorrelation dropped to approximately 0.2, which was then multiplied by the CPU time per iteration. This method has the advantage that it eliminates any bias introduced by manually finding the lags at which the autocorrelations are close zero, but it comes at the expense of having to assume that the shapes of the autocorrelation functions are the same for all choices of parameters. For example the autocorrelation function for autoregressive processes is exponentially decreasing. Based on a visual inspection of the autocorrelations we concluded that a similar assumption can be made in our case too. For each set of parameters, we found the lag at which the autocorrelation is just below 0.2 and the one at which is just above 0.2 and we did a linear interpolation for these two points in order to find the (fractional) lag at which the autocorrelation is 0.2.

ϵ	1.5 ϵ		2ϵ		3 ϵ	
$S = 100$	Interval	L	Interval	L	Interval	L
0.010	(-0.992,-0.988)	-0.405	(-0.993,-0.986)	-0.693	(-0.994,-0.983)	-1.041
0.020	(-0.984,-0.976)	-0.405	(-0.986,-0.972)	-0.693	(-0.988,-0.965)	-1.070
0.039	(-0.968,-0.952)	-0.405	(-0.972,-0.945)	-0.675	(-0.977,-0.932)	-1.084

ϵ	1.5 ϵ		2ϵ		3 ϵ	
$S = 200$	Interval	L	Interval	L	Interval	L
0.005	(-0.996,-0.994)	-0.405	(-0.996,-0.993)	-0.560	(-0.997,-0.991)	-1.099
0.010	(-0.992,-0.988)	-0.405	(-0.993,-0.986)	-0.693	(-0.994,-0.983)	-1.041
0.020	(-0.984,-0.976)	-0.405	(-0.986,-0.972)	-0.693	(-0.988,-0.965)	-1.070
0.040	(-0.967,-0.951)	-0.395	(-0.972,-0.943)	-0.711	(-0.977,-0.931)	-1.099

ϵ	1.5 ϵ		2ϵ		3 ϵ	
$S = 300$	Interval	L	Interval	L	Interval	L
0.007	(-0.994,-0.991)	-0.405	(-0.995,-0.990)	-0.693	(-0.996,-0.988)	-1.099
0.013	(-0.989,-0.984)	-0.375	(-0.991,-0.982)	-0.693	(-0.992,-0.977)	-1.056
0.026	(-0.979,-0.968)	-0.421	(-0.982,-0.963)	-0.721	(-0.985,-0.955)	-1.099

ϵ	1.5 ϵ		2ϵ		3 ϵ		4ϵ	
$S = 400$	Interval	L	Interval	L	Interval	L	Interval	L
0.002	(-0.99797,-0.99695)	-0.407	(-0.99824,-0.99648)	-0.693	(-0.99856,-0.99569)	-1.096	(-0.99876,-0.99502)	-1.390
0.005	(-0.996,-0.994)	-0.405	(-0.996,-0.993)	-0.560	(-0.997,-0.991)	-1.099	(-0.998,-0.99)	-1.609
0.010	(-0.992,-0.988)	-0.405	(-0.993,-0.986)	-0.693	(-0.994,-0.983)	-1.041	(-0.995,-0.98)	-1.386
0.020	(-0.984,-0.976)	-0.405	(-0.986,-0.972)	-0.693	(-0.988,-0.965)	-1.070	(-0.99,-0.96)	-1.386

ϵ	1.5 ϵ		2ϵ		3 ϵ		4ϵ	
$S = 600$	Interval	L	Interval	L	Interval	L	Interval	L
0.002	(-0.99864,-0.99797)	-0.401	(-0.99883,-0.99765)	-0.697	(-0.99904,-0.99712)	-1.099	(-0.99917,-0.99668)	-1.386
0.003	(-0.99729,-0.99593)	-0.407	(-0.99765,-0.99530)	-0.693	(-0.99808,-0.99425)	-1.097	(-0.99834,-0.99336)	-1.386
0.007	(-0.99458,-0.99187)	-0.405	(-0.99530,-0.99061)	-0.692	(-0.99617,-0.98850)	-1.099	(-0.99668,-0.98672)	-1.386
0.013	(-0.98916,-0.98374)	-0.405	(-0.99061,-0.98122)	-0.693	(-0.99233,-0.97700)	-1.098	(-0.99336,-0.97344)	-1.386

Table 4.3: The intervals for α, for $S = 100$ to $S = 600$ and for various distances ϵ from -1.

4.4.3 Performance of Jacobian overrelaxation for the Logistic Regression Problem

For all combinations of parameters as presented in Table 4.3 we simulated Markov chains using Jacobian overrelaxation. In Table 4.4 we summarize for each combination of parameters the lags at which the autocorrelations are close to zero, the autocorrelation time for β_1^2, and the CPU time per iteration. In Table 4.5 we summarize the performance of the method measured as the product of τ, the autocorrelation time, and T, the CPU time per iteration.

For the apparently best combination of $S = 400$, $\epsilon = 0.010$ and an interval for α of "length" 3ϵ, we assessed the variability of the estimator by running 5 chains with different random seeds. From Table 4.7 we can see that there is variability in $\tau's$ from 7.0 to 9.6 as well as in the CPU time per iteration from 0.58 to 0.60 producing a variability in the performance between 4.0 to 5.8 with an average of 4.9. The performance as measured by lag needed to achieve an autocorrelation of approximately 0.2, multiplied by the CPU time per iteration, ranges between 4.0 to 5.3, as seen in Table 4.7, with an average of 4.8.

The variability was also assessed for a large value of the product $\tau \times T$, 11.7, achieved for $S = 200$, $\epsilon = 0.040$, length of the interval 2ϵ, found in Table 4.5. The 5 extra chains produce values of $\tau \times T$ between 8.9 to 10.5 as seen in Table 4.8.

There is a large variability in the estimates of the performance, that could raise the question regarding the conclusion for best α and S for this problem. From the tables, one can see that the estimates of the performance for $S = 100$ are all less than the estimates of the performance for $S = 200$, therefore we can decided that $S = 100$ is not the optimal parameter. The fact that the performance is not very different for different values of S is good, as in practice one will not be able to choose the best S.

For $S = 200$, the smallest value of $\tau \times T$ is 5.6, which is obtained for a "length" of the interval 2ϵ and the "middle"of the interval at half of the distance from -1 that the formula would recommend. For $S = 300$ the minimum is attained at 5.7 for an interval of "length" 2ϵ and the "middle" of the interval given by our formula.

Overall, the smallest value for $\tau \times T$ is 4.5, obtained for $S = 400$, an interval of "length" 3ϵ and the "middle" of the interval as given by the formula derived in Section 4.3.

For $S = 600$, the values for $\tau \times T$ are higher, so we can stop our experiment here, by choosing as the best value $S = 400$.

The performance as assessed by multiplying the lag at which the autocorrelation is 0.2 with the CPU time per iteration is summarized in Table 4.6. The results are qualitatively similar with the previous way of assessing performance, based on the autocorrelation time. For $S = 400$ and the "middle" of the interval at a distance 0.010, the performance is similar for any interval of "length" 1.5ϵ, 2ϵ and 3ϵ.

ϵ	1.5 ϵ			2ϵ			3 ϵ		
it=24000, $S = 100$	lag	τ	T	lag	τ	T	lag	τ	T
0.010	180	80.6	0.15	140	70.7	0.15	140	63.2	0.15
0.020	150	79.4	0.15	160	69.1	0.15	140	76.9	0.15
0.039	120	69.9	0.15	110	64.1	0.15	150	89.3	0.14

ϵ	1.5 ϵ			2ϵ			3 ϵ		
it=12000, $S = 200$	lag	τ	T	lag	τ	T	lag	τ	T
0.005	40	19.7	0.29	60	32.7	0.29	60	24.5	0.31
0.010	90	30.5	0.29	40	18.3	0.31	70	27.4	0.29
0.020	60	26.9	0.30	60	27.9	0.28	40	20.5	0.30
0.040	60	35.2	0.29	80	40.0	0.29	80	36.3	0.31

ϵ	1.5 ϵ			2ϵ			3 ϵ		
it=8000, $S = 300$	lag	τ	T	lag	τ	T	lag	τ	T
0.007	35	13.7	0.46	40	14.4	0.45	35	13.9	0.43
0.013	30	14.7	0.43	25	13.1	0.44	60	25.1	0.44
0.026	50	20.8	0.44	40	14.9	0.44	40	19.3	0.43

ϵ	1.5 ϵ			2ϵ			3 ϵ			4 ϵ		
it=6000, $S = 400$	lag	τ	T	lag	τ	T	lag	τ	T	lag	τ	T
0.002	150	18.5	0.61	25	9.0	0.62	25	11.6	0.63	25	9.4	0.61
0.005	30	16.2	0.59	30	10.7	0.58	25	9.7	0.57	25	10.3	0.58
0.010	15	8.0	0.59	20	9.3	0.58	20	7.9	0.57	30	12.5	0.58
0.020	20	10.5	0.57	25	10.3	0.58	20	9.7	0.58	25	10.6	0.59

ϵ	1.5 ϵ			2ϵ			3 ϵ			4 ϵ		
it=6000, $S = 600$	lag	τ	T	lag	τ	T	lag	τ	T	lag	τ	T
0.002	25	7.6	0.87	80	20.6	0.88	50	13.2	0.88	20	7.4	0.89
0.003	25	7.7	0.86	25	10.0	0.88	20	7.8	0.88	15	6.2	0.87
0.007	15	7.3	0.90	25	8.8	0.89	20	6.5	0.93	15	6.5	0.89
0.013	20	7.6	0.90	15	6.0	0.89	20	7.0	0.89	15	7.5	0.91

Table 4.4: The lags at which the autocorrelations are close to zero, the autocorrelation time, τ, and the time per iteration, T, for each combination of S and length and positioning of the interval around α.

$S = 100$, ϵ	1.5 ϵ	2ϵ	3 ϵ
0.010	12.0	10.9	9.3
0.020	12.0	10.1	11.7
0.039	10.5	9.5	12.6

$S = 200$, ϵ	1.5 ϵ	2ϵ	3 ϵ
0.005	5.7	9.6	7.6
0.010	8.8	5.6	7.9
0.020	8.0	7.7	6.1
0.040	10.3	11.7	11.3

$S = 300$, ϵ	1.5 ϵ	2ϵ	3 ϵ
0.007	6.3	6.4	6.0
0.013	6.4	5.7	11.0
0.026	9.1	6.6	8.4

$S = 400$, ϵ	1.5 ϵ	2ϵ	3 ϵ	4ϵ
0.002	11.3	5.6	7.3	5.8
0.005	9.5	6.2	5.6	6.0
0.010	4.7	5.4	4.5	7.3
0.020	6.0	6.0	5.6	6.3

$S = 600$, ϵ	1.5 ϵ	2ϵ	3 ϵ	4ϵ
0.002	6.6	18.0	11.6	6.6
0.003	6.6	8.8	6.9	5.4
0.007	6.6	7.8	6.1	5.8
0.013	6.8	5.4	6.2	6.8

Table 4.5: The performance of Jacobian overrelaxation as assessed by the product of τ, the autocorrelation time, and T, the CPU time per iteration. The results above are for all combinations of S and length of interval and position of α with respect to -1.

$S = 100$, ϵ	$1.5\ \epsilon$	2ϵ	$3\ \epsilon$
0.010	9.6	7.9	8.3
0.020	10.2	7.3	10.9
0.039	9.7	8.5	11.3

$S = 200$, ϵ	$1.5\ \epsilon$	2ϵ	$3\ \epsilon$
0.005	5.3	10.5	6.2
0.010	7.0	5.2	6.9
0.020	7.7	7.0	5.5
0.040	9.5	8.9	9.8

$S = 300$, ϵ	$1.5\ \epsilon$,	2ϵ	$3\ \epsilon$
0.007	5.4	4.6	5.1
0.013	6.1	5.4	10.0
0.026	7.6	6.5	7.0

$S = 400$, ϵ	$1.5\ \epsilon$	2ϵ	$3\ \epsilon$	4ϵ
0.002	6.0	4.6	6.5	4.9
0.005	9.1	5.1	4.9	5.3
0.010	4.3	5.0	4.5	6.5
0.020	5.6	5.7	5.2	5.7

$S = 600$, ϵ	$1.5\ \epsilon$	2ϵ	$3\ \epsilon$	4ϵ
0.002	5.5	7.4	7.6	5.6
0.003	5.4	8.4	6.4	5.1
0.007	6.4	6.9	6.2	5.1
0.013	6.1	5.3	6.7	6.7

Table 4.6: The performance as assessed by the lag at which autocorrelations are 0.2 multiplied by the CPU time per iteration, T.

Run	lag	τ	T	$\tau \times T$	$lag_{0.2} \times T$
1	15	7.7	0.59	4.6	4.4
2	20	8.8	0.58	5.1	5.1
3	15	7.0	0.58	4.0	4.0
4	20	8.8	0.59	5.2	5.2
5	20	9.6	0.60	5.8	5.3

Table 4.7: Variability for the performance of JO for $S = 400$, $it = 6,000$ and $\alpha \in (-0.994, -0.983)$. The 5 extra runs are summarized by lag, lag at which the autocorrelations appear to be zero, τ, the autocorrelation time, T, time per iteration, $\tau \times T$, performance, $lag_{0.2} \times T$, performance as assessed by multiplying the (fractional) lag at which the autocorrelation is 0.2.

Run	lag	τ	T	$\tau \times T$	$lag_{0.2} \times T$
1	70	34.0	0.31	10.5	10.1
2	70	34.5	0.30	10.4	8.6
3	60	30.8	0.31	9.5	9.0
4	60	34.4	0.30	10.5	9.5
5	60	29.5	0.30	8.9	7.8

Table 4.8: Variability for the performance of JO for $S = 200$, $it = 12,000$ and $\alpha \in (-0.972, -0.943)$. The 5 extra runs are summarized by lag, lag at which the autocorrelations appear to be zero, τ, the autocorrelation time, T, time per iteration, $\tau \times T$, performance, $lag_{0.2} \times T$, performance as assessed by multiplying the (fractional) lag at which the autocorrelation is 0.2.

4.4.4 Performance of the Metropolis Algorithm for the Bayesian Logistic Regression Problem

We chose for comparison the Metropolis algorithm, because it is easy to implement, it is widely applicable, and it is widely used as well. Simple Gibbs sampling is not easy to implement as we cannot obtain an analytical form for the full conditional distributions. One could use adaptive rejection sampling as presented Gilks and Wild (1992), but this is applicable only to problems (such as this one) for which the full-conditional densities are log-concave. We aim here to evaluate JO for a wider class of problems.

The global Metropolis algorithm updates all components at once. For our logistic regression problem, we used a Gaussian distribution proposal, centered at the current point and with standard deviation of 0.17. This proposal produced a rejection rate of 75%, which is close to the value of 76.6% that Roberts et al. (1997) proved that produces an optimal chain as the dimensionality of the problem increases. We ran a chain $6,000,000$ iterations long, where an iteration corresponds to updating all components once (i.e., one global Metropolis update), and we kept every $100th$ iteration. For this thinned chain, the autocorrelations are close to zero at lag 600, the autocorrelation time for the second component squared is 259.6, and the CPU time per 100 iterations is 0.022. Therefore the performance as given by the product of the autocorrelation time and the time per 100 iteration is 5.6. We assessed the variability of this estimator by running five more chains with the same parameters, but with different random seeds. The autocorrelation times multiplied by the CPU times per 100 iterations are presented in Table 4.9, and they vary from 7.3 to 9.4 with an average of 8.1. The other performance measure, based on the product of the (fractional) lag at which the autocorrelations are 0.2 and the time per iteration, gives us an estimate of 4.7 for the first chain and the variability of this estimate is presented in Table 4.9, with an average of 7.1.

If we consider that Jacobian overrelaxation can take advantage of the fact that the energy of a state can be written as a function of the energy of the previous state when only one component is changed, then Metropolis should take advantage of this as well, by using Metropolis in which one component is updated after which the new state is accepted or rejected. In this case the rejection rate cannot be justified as in the global Metropolis algorithm as the result used there was based on an asymptotic result. The increase in the dimensionality of the problem doesn't have an impact on the local Metropolis algorithm which updates one component at a time. The rejection rate in this case should be lowered and as a rule of thumb a optimal rejection rate is around 50%. A standard deviation of 0.57 for the Gaussian proposal centered at the current state achieves a rejection rate of about 50%. A chain $1,600,000$ iterations long, where each iteration consists of updating all the components once, was simulated and every $100th$ iterations was kept. For this chain the lag at which the autocorrelation for β_1^2 is close to zero is 80, the autocorrelation time is 36.7, and the time per 100 iteration is 0.32, giving a performance of 11.8. The performance for local Metropolis seems to be worse than the performance for the global Metropolis because the time per iteration was calculated without taking advantage of the fact that the energy when you update one component at a time can be written as the energy of the previous state as discussed in Section 4.4.1. The MATLAB program used for all our simulations was not able to take advantage of this property because the overhead operations seem to take longer than multiplications of matrices. By using another programming language, for example C, and taking advantage of this property, for our logistic regression problem of dimension 16, we could probably improve the performance of local Metropolis and Jacobian overrelaxation 10 times. Local Metropolis would then be more desirable to use than global Metropolis and it is therefore the appropriate algorithm to compare with JO.

Five additional local Metropolis chains of the same length are simulated and their variability in performance ranges from 10.0 to 14.3 as measured by the product of τ and CPU time per 100 iterations with an average of 12.2 and between 8.5 to 11.1 as measured by the product between the number of iterations needed to achieve a autocorrelation of about 0.2 and CPU time per 100 iterations with an average of 9.8. These results can be found in Table 4.10.

Run	lag	τ	T	$\tau \times T$	$lag_{0.2} \times T$
1	600	324.6	0.022	7.3	6.4
2	800	366.8	0.023	8.3	7.3
3	600	343.5	0.023	7.9	6.4
4	650	336.0	0.023	7.7	6.7
5	850	419.4	0.023	9.4	8.9

Table 4.9: Five extra chains' performance of global Metropolis to assess the variability of the performance. Performance is measured by $\tau \times T$, the product between the autocorrelation time and the time per iteration, and by the (fractional) lag at which the autocorrelation is 0.2 multiplied by the time per iteration.

Run	lag	τ	T	$\tau \times T$	$lag_{0.2} \times T$
1	90	38.5	0.33	12.6	10.5
2	100	43.1	0.33	14.3	11.1
3	70	31.2	0.32	10.0	8.5
4	80	33.6	0.33	11.1	9.1
5	100	40.8	0.32	13.0	9.7

Table 4.10: Five extra chains' performance of local Metropolis to assess the variability of the performance. Performance is measured by $\tau \times T$, the product between the autocorrelation time and the time per iteration, and by the (fractional) lag at which the autocorrelation is 0.2 multiplied by the time per iteration.

4.4.5 Efficiency of Jacobian Overrelaxation versus Metropolis

The best performance for Jacobian overrelaxation was accomplished for $S = 400$, 3ϵ "length" interval, where $\epsilon = 0.01$. The average value of $\tau \times T$, based on five chains simulated using the same parameters, but different random seeds, is 4.9. Jacobian overrelaxation performance is therefore slightly superior to global Metropolis performance. The ratio of the average performance, measured as the autocorrelation times multiplied by the CPU time, for global Metropolis versus Jacobian overrelaxation, is 8.1 : 4.9, that makes Jacobian overrelaxation 1.7 times more efficient. We calculated the standard error for the ratio estimators as in Murthy (1967). The standard error for the above estimator is 0.1. The other measure of performance gives us a ratio of efficiency of 1.5 ± 0.1. However we can take advantage of the fact that we can improve the computational time by writing the energy of one state as a function of the energy for the previous state as described in Section 4.4.1. Both Jacobian overrelaxation and Metropolis for which we update one component at a time can take advantage of this property.

For local Metropolis the average value of $\tau \times T$, based also on five chains is 12.2. Therefore Jacobian overrelaxation is 2.5 ± 0.2 times more efficient than Metropolis in which you update one component at a time.

A similar result is obtained if you compare the performances as given by the number of iterations needed by the autocorrelations to reach 0.2 multiplied by the CPU time per iteration. The best performance for Jacobian overrelaxation obtained by averaging 5 different values is 4.8 and for Metropolis one component at a time is 9.8, making Jacobian overrelaxation 2.0 ± 0.1 times more efficient.

As the correlation between the components increases, Jacobian overrelaxation performs better and better as compared to Metropolis algorithm. This is due to the fact that

Jacobian overrelaxation suppresses random walks. As the ratio of the square root of the largest to the smallest eigenvalue increases k times, Jacobian overrelaxation becomes only k times worse, but the Metropolis algorithm, which performs a random walk, becomes k^2 times worse.

Chapter 5

Coupled Markov Chain Monte Carlo Estimators

In this chapter, we show how large improvements in the accuracy of MCMC estimates for posterior expectations can sometimes be obtained by coupling a Markov chain that samples from the posterior distribution with a chain that samples from a Gaussian approximation to the posterior. Use of this method requires a coupling scheme that produces high correlation between the two chains. An efficient estimator can then be constructed that exploits this correlation, provided an accurate value for the expectation under the Gaussian approximation can be found, which for simple functions can be done analytically. Good coupling schemes are available for many Markov chain samplers, including Gibbs sampling with standard conditional distributions. For many moderate-dimensional problems, the improvement in accuracy using this method will be much greater than the overhead from simulating a second chain.

5.1 Introduction

One of the early solutions for drawing samples from a complicated posterior distribution f, was to find a Gaussian approximation, g, and use $E_g(a(y))$ as an approximation to $E_f(a(y))$. This reduces the problem to calculating the expected value of the function a with respect to a Gaussian distribution, which, depending on the function a, may be doable analytically, or by Gaussian quadrature (Thisted 1988, Section 5.3), or by efficient Monte Carlo techniques. Another possible solution is importance sampling (Tanner 1993, Section 3.3.3), perhaps using the Gaussian approximation to the posterior distribution. A sample from the Gaussian distribution is drawn and the points of the sample are reweighted to account for the fact that the sample is not from the correct distribution.

In many problems, the Gaussian approximation will be close to the posterior distribution, but not close enough to provide sufficiently accurate estimates. If the posterior distribution has heavier tails than the Gaussian approximation, even importance sampling will not provide good estimates, as in this case the importance sampling weights will be highly variable, and only a few points from the sample drawn from the Gaussian approximation will contribute to the estimate. For this reason, Markov chain Monte Carlo techniques are now commonly used to estimate expected values with respect to complex or high or even moderate-dimensional posterior distributions.

In this chapter, we will use the Gaussian approximation to the posterior distribution to improve the accuracy of Markov chain Monte Carlo estimates. The mean of the Gaussian approximation is taken to be the mode of the posterior distribution. The mode can be found using the Newton-Raphson algorithm, for example, perhaps using as an initial value the sample mean, $\overline{y} = (1/n)\sum_{i=1}^{n} y_i$, where $y_1, \ldots, y_n$ is a sample generated by simulating a Markov chain that converges to f. The variance-covariance matrix for the

Gaussian approximation is chosen to be minus the inverse of the Hessian (matrix of second derivatives) of the logarithm of the posterior density, evaluated at the mode of the posterior distribution.

The Markov chain used to generate the sample $y_1, \ldots, y_n$ from f will be coupled with a chain that converges to the Gaussian approximation, g, producing a sample $x_1, \ldots, x_n$. We hope that these two samples will be highly correlated. To take advantage of this correlation, we construct new estimators for $E_f(a(y))$ that depend on both the y's and the x's and which make use of $E_g(a(x))$, which is assumed to be accurately known.

One such estimator is

$$\overline{a}_y - \alpha(\overline{a}_x - E_g(a(x))), \tag{5.1}$$

where $\overline{a}_x = (1/n)\sum_{i=1}^{n} a(x_i)$. For $\alpha = Cov\,(\overline{a}_y, \overline{a}_x)/Var\,(\overline{a}_x)$, this is the best unbiased linear estimator. In practice, α will have to be determined from the data, introducing some small bias. This new estimator is sometimes much more accurate than $\overline{a}_y$, due to the information provided by the sample drawn from the Gaussian approximation, which for problems of moderate dimensionality can be found with little computational effort.

In the context of simple Monte Carlo estimation from simulation data, a similar technique has been used to reduce variance using control variates (Ripley 1987; Kelijnen 1974, Section III.4). Cheng (1978) investigates the properties of estimators of type (5.1) when the joint distribution for x and y is Gaussian, an assumption that seems to be reasonable in their queuing system context, but perhaps not for our application. Lavenberg, Moeller and Welch (1982) discuss the loss of variance reduction due to estimating α from the data. Another use of coupling to improve estimation is due to Frigessi, Gåsemyr and Rue (2000), who use antithetic coupling of two chains sampling from the same distribution.

Schmeiser and Chen (1991) introduced the basic idea of coupling a Markov chain that samples from the desired distribution with an approximating chain. The same idea is mentioned in Chen, Shao and Ibrahim (2000), Section 3.4, as well. They proposed an estimator of type (5.1) and proved that it is consistent. They did not demonstrate the efficiency of this estimator on any example. They did not introduced the coupled estimators based on regression models that we present in Section 5.4.

In Section 5.2, we show how the coupling procedure can produce correlation between chains. Section 5.3 presents the estimator (5.1) and discusses its properties and efficiency. In Section 5.4, this estimator is seen as the simplest of a larger class of estimators that can also model more complex relationships between the two chains. Section 5.5 presents an example based on the data on pump failures from Gelfand and Smith (1990). We conclude, in Section 5.6, by discussing possible further extensions and applications.

5.2 Coupling to an Approximating Chain - Toy Example

We couple two chains, as described in Section 2.4, using different transition functions, sampling from similar but different distributions. We start both chains from the same initial state and hope the chains will stay close together for the whole run, producing high correlation between the states of the two chains. The success of coupling in producing chains that move together depends on the Markov chain Monte Carlo methods used to sample from the distributions, and on the way they are expressed in terms of ϕ functions.

To illustrate how coupling works, and later the properties of the estimators we introduce, we consider a toy example in which a gamma distribution is approximated with a Gaussian distribution. In Figure 5.1, we can see the effect of coupling a chain sampling from the Gamma(10,5) distribution, denoted by f, with a chain sampling from a Gaussian approximation to it, denoted by g, whose mean is the mode of f, and whose variance is

$-\left[d^2 \log f(x)/dx^2\right]^{-1}$ evaluated at the mode. We used the Metropolis algorithm to sample from these distributions, with a proposal that was a Gaussian distribution centered at the current point and with standard deviation three. The coupling for this example is done by using the same Gaussian random numbers for the proposals and the same uniform random numbers for the accept-reject decisions. The random noise, $v_t = (n_t, u_t)$, therefore has two components, $n_t \sim N(0, 3^2)$ and $u_t \sim$ Uniform(0,1), and the two deterministic functions ϕ_f and ϕ_g are defined by

$$\phi_f(y_{t-1}, (n_t, u_t)) = \begin{cases} y_{t-1} + n_t & \text{if } u_t < f(y_{t-1} + n_t)/f(y_{t-1}) \\ y_{t-1} & \text{otherwise} \end{cases} \tag{5.2}$$

and similarly

$$\phi_g(x_{t-1}, (n_t, u_t)) = \begin{cases} x_{t-1} + n_t & \text{if } u_t < g(x_{t-1} + n_t)/g(x_{t-1}) \\ x_{t-1} & \text{otherwise} \end{cases} \tag{5.3}$$

where f is the gamma density $f(x|\alpha, \beta) = (x^{\alpha-1} e^{-x/\beta})/(\Gamma(\alpha)\beta^\alpha)$ with parameters $\alpha = 10$ and $\beta = 5$, and g is the Gaussian density with mean $\beta(\alpha - 1) = 45$ and variance $\beta^2(\alpha - 1) = 225$. Note that the mean used for the Gaussian approximation is the mode of Gamma distribution and the variance used is based on the curvature at the mode, and is not the same as the actual variance of the gamma distribution.

How high the correlation between chains will be depends on the coupling technique and on how close the Gaussian approximation is to the target distribution. In this example, the coupling is good because the step size is small — smaller than would be optimal if only one chain were used for estimation. For realistic problems, methods that produce good coupling at less cost are needed, as illustrated in the example of Section 5.5.

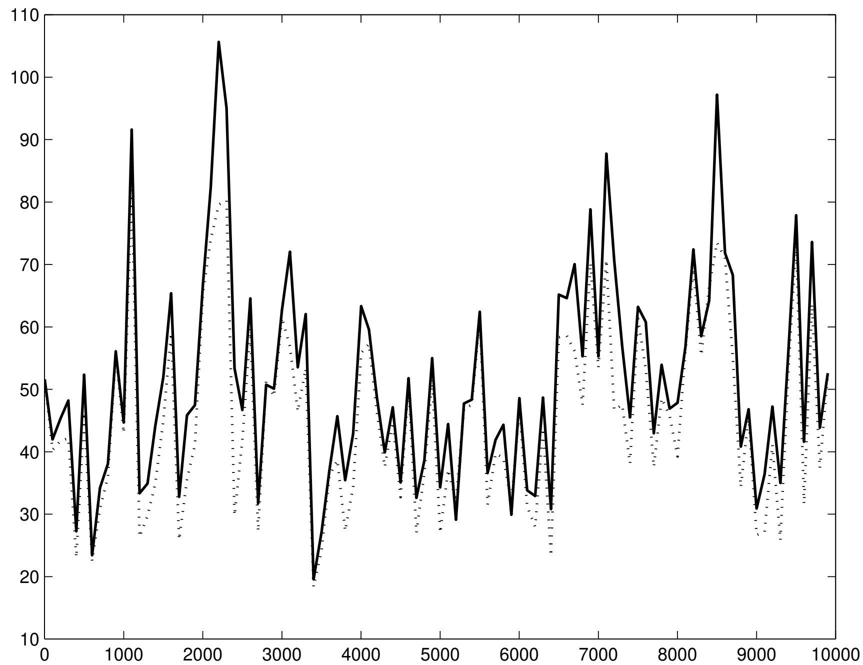


Figure 5.1: Coupling of chains sampling from Gamma(α, β) and the Gaussian approximation with mean $\beta(\alpha-1)$ and variance $\beta^2(\alpha-1)$. Here $\alpha = 10$ and $\beta = 5$. Every hundredth point of a long run of the Metropolis chains is plotted for each distribution. The solid line is the sample from the Gamma distribution and the dotted line is the Gaussian approximation.

5.3 A Simple Estimator Exploiting Coupling between Two Chains

Assume we have samples from two coupled chains, $(y_1, \ldots, y_n)$ from the distribution f, and $(x_1, \ldots, x_n)$ from g, the Gaussian approximation to f. The usual estimator for $E_f(a(y))$ would be $\overline{a}_y = (1/n)\sum_{i=1}^n a(y_i)$. To exploit the correlation of the two chains, we can construct a new estimator for $E_f(a(y))$ of the form:

$$\overline{a}_y - \alpha(\overline{a}_x - E_g(a(x))), \tag{5.4}$$

where $\overline{a}_x = (1/n)\sum_{i=1}^n a(x_i)$. This estimator is unbiased for any fixed α. It has minimum variance (Ripley 1987, Section 5.3), when

$$\alpha = \frac{Cov(\overline{a}_y, \overline{a}_x)}{Var(\overline{a}_x)} \tag{5.5}$$

$$= Corr(\overline{a}_y, \overline{a}_x)\frac{Std(\overline{a}_y)}{Std(\overline{a}_x)}. \tag{5.6}$$

If the pairs of points from the two chains were independent, the appropriate estimate for α would be

$$\widehat{\alpha} = \frac{\sum_{i=1}^n (a(y_i) - \overline{a}_y)(a(x_i) - \overline{a}_x)}{\sum_{i=1}^n (a(x_i) - \overline{a}_x)^2}. \tag{5.7}$$

For samples of dependent pairs obtained using Markov chains, we will still use estimator (5.7), since it is close to optimal.

For notational simplicity, assume we are interested in $\mu_f = E_f(y)$ and we know $\mu_g = E_g(x)$. The first estimate for μ_f we will look at is:

$$\widehat{\mu}_f^{(1)} = \overline{y} - \widehat{\alpha}(\overline{x} - \mu_g) \tag{5.8}$$

where, applying (5.7),

$$\widehat{\alpha} = \frac{\sum_{i=1}^n (y_i - \overline{y})(x_i - \overline{x})}{\sum_{i=1}^n (x_i - \overline{x})^2}. \tag{5.9}$$

The estimation of α can introduce bias, but we can still show that estimator (5.8) is consistent. If the two chains are ergodic, the ergodic theorem helps us establish the following:

$$\overline{y} \to_p \mu_f \tag{5.10}$$

$$\overline{x} \to_p \mu_g \tag{5.11}$$

$$\frac{\sum_{i=1}^n (x_i - \overline{x})^2}{n} \to_p E_g((x - \mu_g)^2) \tag{5.12}$$

$$\frac{\sum_{i=1}^n x_i y_i}{n} \to_p E_{fg}(xy) \tag{5.13}$$

The last statement is justified if the joint coupled chain (x, y) is ergodic. However, for the purpose of this proof, all that is required is that the joint coupled chain converges to some distribution, such that (5.13) converges to some constant. From (5.10) to (5.13) it follows that α converges to a constant and that

$$\widehat{\mu}_f^{(1)} \quad \rightarrow_p \quad \mu_f. \tag{5.14}$$

In Schmeiser and Chen (1991), it is shown that $\widehat{\alpha}(\overline{x} - \mu_g) \rightarrow 0$ as long as the variances of x and y are finite.

The asymptotic efficiency of this estimator can be investigated by considering α to be constant. If we write the estimator (5.8) as $\widehat{\mu}_f^{(1)} = (1/n) \sum_{i=1}^n z_i$, where $z_i = y_i - \widehat{\alpha}(x_i - \mu_g)$, we can estimate its variance by using (2.23) as described in Section 2.1:

$$\widehat{Var}\left(\widehat{\mu}_f^{(1)}\right) \quad = \quad \frac{\sum_{i=1}^n \left(z_i - \widehat{\mu}_f^{(1)}\right)^2}{n - \widehat{\tau}} \frac{\widehat{\tau}}{n}, \tag{5.15}$$

where $\widehat{\tau}$ is the estimated autocorrelation time, which is obtained by summing the estimated autocorrelations of $(z_1, \ldots, z_n)$ at all positive and negative lags up to the lag past which the autocorrelations seem to be nearly zero.

For the example presented in Section 5.2, we ran two coupled chains 100,000 iterations long and found that the autocorrelations were close to 0 past lag 350. The correlation between the two chains is 0.9466 and $\widehat{\alpha} = 0.9926$. The parameter estimated here is the mean of Gamma(10,5), which is 50. The estimate using (5.8) is 50.21 with standard error of 0.22, whereas the traditional estimate is 49.34 with standard error of 0.63. These results show that exploiting the correlations between chains improves the efficiency of the estimator by a factor of $(0.63/0.22)^2 \approx 8$, if we ignore the increase in computation time.

5.4 Coupled estimators based on regression models

The relationship between $y_1, \ldots, y_n$ and $x_1, \ldots, x_n$ could be modeled by a simple linear regression:

$$y_i = \beta_0 + \beta_1(x_i - \mu_g) + \epsilon_i \tag{5.16}$$

The estimator (5.8) is exactly the least square estimator for the intercept in this simple linear regression model. This observation leads us to consider new estimators for μ_f based on higher-order regression models. The samples based on the two coupled chains in Section 5.2 will not be linearly related because the upper tail of the Gamma distribution is heavier than that of the Gaussian distribution. In Figure 5.2 we can see that a third-order regression model fits better than the linear model.

Suppose we fit the following model for how $(y_1, \ldots, y_n)$ relates to $(x_1, \ldots, x_n)$:

$$y_i = \beta_0 + \beta_1(x_i - \mu_g) + \beta_2(x_i - \mu_g)^2 + \beta_3(x_i - \mu_g)^3 + \epsilon_i \tag{5.17}$$

If $\widehat{\beta}_0, \widehat{\beta}_1, \widehat{\beta}_2, \widehat{\beta}_3$ are the least square estimates for the regression coefficients, we propose the following estimator for μ_f:

$$\widehat{\mu}_f^{(3)} = \overline{y} - \widehat{\beta}_1\overline{(x-\mu_g)} + \widehat{\beta}_2\sigma_g^2 - \widehat{\beta}_2\overline{(x-\mu_g)^2} - \widehat{\beta}_3\overline{(x-\mu_g)^3} \tag{5.18}$$

where we denote by $\sigma_g^2 = E\left((x-\mu_g)^2\right)$, $\overline{(x-\mu_g)} = (1/n)\sum_{i=1}^n (x_i - \mu_g)$, $\overline{(x-\mu_g)^2} = (1/n)\sum_{i=1}^n (x_i - \mu_g)^2$, and $\overline{(x-\mu_g)^3} = (1/n)\sum_{i=1}^n (x_i - \mu_g)^3$. By using the ergodic theorem, we can establish that the coefficients, $\widehat{\beta}_i$, converge to some constants, and that

$$\overline{(x-\mu_g)} \to_p 0 \tag{5.19}$$

$$\overline{(x-\mu_g)^2} \to_p \sigma_g^2 \tag{5.20}$$

$$\overline{(x-\mu_g)^3} \to_p 0 \tag{5.21}$$

$$\overline{y} \to_p \mu_f \tag{5.22}$$

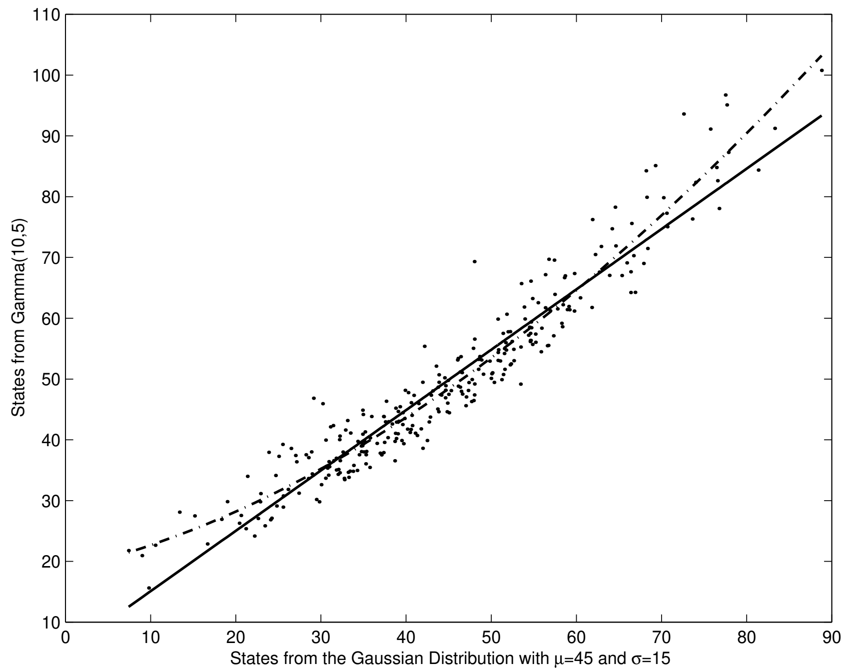


Figure 5.2: Every 350th point of the chains from Gamma(10,5) and the Gaussian approximation with mean 45 and standard deviation 15, along with the regression lines for first and third order models.

It follows that $\widehat{\mu}_f^{(3)} \to_p \mu_f$.

Letting $z_i = y_i - \widehat{\beta}_1(x_i - \mu_g) + \widehat{\beta}_2\sigma_g^2 - \widehat{\beta}_2(x_i - \mu_g)^2 - \widehat{\beta}_3(x_i - \mu_g)^3$, we can write $\widehat{\mu}_f^{(3)} = (1/n)\sum_{i=1}^n z_i$, and estimate its variance by:

$$\widehat{Var}\left(\widehat{\mu}_f^{(3)}\right) = \frac{\sum_{i=1}^n \left(z_i - \widehat{\mu}_f^{(3)}\right)^2}{n - \widehat{\tau}} \frac{\widehat{\tau}}{n} \tag{5.23}$$

where $\widehat{\tau}$ is the estimated autocorrelation time for the z_i.

We expect this estimator to be better than $\widehat{\mu}_f^{(1)}$ because the z_i's are the residuals of the model plus the constant $\widehat{\beta}_0 + \widehat{\beta}_2\sigma_g^2$. The better the model we fit, the smaller the variance of the residuals and hence the smaller the variance of $\widehat{\mu}_f^{(3)}$. Note, however, that the regression-based estimators are valid even when the regression model is not correct.

For the gamma example presented in Section 5.2, this third-order regression estimator, $\widehat{\mu}_f^{(3)}$, gives an estimate for μ_f of 50.20 with standard error 0.18. This estimator is about 1.5 times better than the linear regression estimate, $\widehat{\mu}_f^{(1)}$, and about 12 times more efficient than the standard estimator based on one chain.

5.5 Pumps Data Example

To illustrate the performance of the new coupled estimators we will consider the model for data on pump failures from Gelfand and Smith (1990). The data consists of p counts, $s_1, \ldots, s_p$, that represent the number of failures for p pumps over known periods of time, $t_1, \ldots, t_p$. It is assumed that conditional on unknown failure rates, $\lambda_1, \ldots, \lambda_p$, the counts $s_1, \ldots, s_p$ are independent and Poisson distributed with means $\lambda_i t_i$. The distribution of the unknown λ_i's is gamma with a known shape parameter α and unknown scale factor β and the λ_i's are independent given β. The prior for the hyperparameter β is inverse gamma with a known shape parameter γ and a known scale parameter δ.

The densities for this hierarchical Bayesian model are given by:

$$P(s_i|\lambda_i) = e^{-\lambda_i t_i}\frac{(\lambda_i t_i)^{s_i}}{s_i!}, \; i = 1, \ldots, p \tag{5.24}$$

$$P(\lambda_i|\alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha}\lambda_i^{\alpha-1}e^{-\lambda_i/\beta}, \; i = 1, \ldots, p \tag{5.25}$$

$$P(\beta|\gamma, \delta) = \frac{\delta^\gamma}{\Gamma(\gamma)}\frac{1}{\beta^{\gamma+1}}e^{-\delta/\beta} \tag{5.26}$$

The joint conditional distribution of the parameters of interest, $\beta, \lambda_1, \ldots, \lambda$, given the observed $s_1, \ldots, s_p$ and $t_1, \ldots, t_p$ is:

$$P(\beta, \lambda_1, \ldots, \lambda_p|s_1, \ldots, s_p) \propto P(\beta, \lambda_1, \ldots, \lambda_p, s_1, \ldots, s_p) \tag{5.27}$$

$$\propto \; P(s_1, \ldots, s_p|\lambda_1, \ldots \lambda_p)P(\lambda_1, \ldots, \lambda_p|\beta)P(\beta) \tag{5.28}$$

$$\propto \prod_{i=1}^{p}\left(e^{-\lambda_i t_i}\frac{(\lambda_i t_i)^{s_i}}{s_i!}\right)\prod_{i=1}^{p}\left(\frac{1}{\beta^{\alpha}}\lambda_i^{\alpha-1}e^{-\lambda_i/\beta}\right)\frac{1}{\beta^{\gamma+1}}e^{-\delta/\beta} \tag{5.29}$$

It is more convenient to work in terms of $\theta = 1/\beta$. The joint conditional density for $\theta, \lambda_1, \ldots, \lambda_p$, which we will denote by f, is:

$$f(\theta, \lambda_1, \ldots, \lambda_p | s_1, \ldots, s_p) \propto$$
$$\prod_{i=1}^{p}\left(e^{-\lambda_i t_i}(\lambda_i t_i)^{s_i}\right)\prod_{i=1}^{p}\left(\theta^{\alpha}\lambda_i^{\alpha-1}e^{-\lambda_i\theta}\right)\theta^{\gamma-1}e^{-\delta\theta} \tag{5.30}$$

We will use Gibbs sampling to sample from the posterior distribution of $\theta, \lambda_1, \ldots, \lambda_p$. The conditional distribution for θ given the λ_i's is

Gamma$(p\alpha+\gamma, 1/\left(\delta+\sum_{i=1}^{p}\lambda_i\right))$. The conditional distribution for each λ_i given θ and all $\{\lambda_j\}_{j\neq i}$ is Gamma$(s_i+\alpha, 1/\left(t_i+\theta\right))$. One step of Gibbs sampling at time t is done by generating $u_0^{(t)}, \ldots, u_p^{(t)}$ from the Uniform(0,1) distribution and applying the following deterministic transition functions:

$$\theta^{(t)} = F^{-1}\left(u_0^{(t)};\ p\alpha+\gamma,\ \frac{1}{\delta+\sum_{i=1}^{p}\lambda_i^{(t-1)}}\right) \tag{5.31}$$

$$\lambda_i^{(t)} = F^{-1}\left(u_i^{(t)};\ s_i+\alpha,\ \frac{1}{t_i+\theta^{(t)}}\right),\quad i=1,\ldots,p, \tag{5.32}$$

where F^{-1} is the inverse cumulative distribution function for the gamma distribution with shape and scale parameters as specified. This method of generating gamma variates is not the fastest; it is used because it produces good coupling.

The Gaussian approximation, g, for the joint posterior distribution (5.30) has mean $\boldsymbol{\mu}$ equal to the mode of the posterior distribution and variance-covariance matrix $\boldsymbol{\Sigma}$ equal to minus the inverse of the Hessian of the logarithm of the posterior density evaluated at $\boldsymbol{\mu}$. To do Gibbs sampling for the Gaussian approximation, consider $\boldsymbol{x} = (x_0, x_1, \ldots, , x_p)$, a $p+1$ dimensional Gaussian random vector with mean $\boldsymbol{\mu}$ and variance-covariance matrix

$\mathbf{\Sigma}$. Denote by $\mathbf{\Omega}$ the inverse of the variance-covariance matrix $\mathbf{\Sigma}$, with elements ω_{ij}, $i, j = 0, 1, \ldots, p$. The conditional distribution of x_i given the other components $\{x_j\}_{j \neq i}$ is Gaussian with mean $\mu_i - (1/\omega_{ii}) \sum_{j \neq i} \omega_{ij}(x_j - \mu_j)$ and variance $1/\omega_{ii}$. The coupling of the two chains in this example is done by using at each step the same Uniform(0,1) random numbers. One Gibbs sampling step at time t for the Gaussian approximation is

$$x_i^{(t)} = G^{-1}\left(u_i^{(t)};\ \mu_i - \frac{1}{\omega_{ii}}\left(\sum_{j<i}\omega_{ij}(x_j^{(t)} - \mu_j) + \sum_{j>i}\omega_{ij}(x_j^{(t-1)} - \mu_j)\right),\ \frac{1}{\omega_{ii}}\right)$$

$$i = 0, \ldots, p \tag{5.33}$$

where G^{-1} is the inverse cumulative distribution function for the Gaussian distribution with mean and variance as specified.

Following Gelfand and Smith (1990), we set the known parameters to $p = 10$, $\delta = 1$, $\gamma = 0.1$, and $\alpha = \overline{\rho}^2/(s_\rho^2 - p^{-1}\overline{\rho}\sum_{i=1}^p t_i^{-1})$, where $\rho_i = s_i/t_i$, $\overline{\rho} = (1/p)\sum_{i=1}^p \rho_i$, and $s_\rho^2 = (1/p)\sum(\rho_i - \overline{\rho})^2$. We drew a sample from the posterior distribution using 1000 iterations of Gibbs sampling, starting with $\theta^{(0)} = 1$ and $\lambda_i^{(0)} = s_i/t_i$. Based on this sample we found Monte Carlo estimates for means of the parameters and used them as starting values for the Newton-Raphson method to find the mode, which we used as the mean of the Gaussian approximation. Estimating the parameters of the Gaussian approximation is computationally inexpensive in this example; Newton-Raphson converges to the mode of the posterior distribution in 11 steps using the starting value mentioned. Once the Gaussian approximation was found, we simulated the coupled Gibbs sampling chain sampling from it for the same 1000 iterations as the chain sampling from the posterior distribution.

We are interested in estimating the expected value of each of the parameters $\theta, \lambda_1, \ldots, \lambda_p$ with respect to the posterior distribution. We discarded the first 100 states of each chain as burn-in, which is more than adequate for this problem, for which the autocorrelations are close to zero by lag 5. The results for three estimators and their standard errors are presented in Table 5.1. The estimators evaluated are $\overline{y}$, the traditional Markov chain estimator based on only one chain, estimator $\widehat{\mu}^{(1)}$ presented in Section 5.3, and estimator $\widehat{\mu}^{(3)}$ introduced in Section 5.4. The last three columns of Table 5.1 present the relative efficiencies of the estimators as ratios of their estimated variances. For this problem the estimates based on third-order regression are all much more efficient than the estimates based on one chain, or even those based on simple linear regression. Particularly striking is the estimate for λ_1 based on third-order regression, which is approximately 24000 times more efficient than the estimate based on one chain. This is because the relationship between the two chains is very tight, with little variation unexplained in the third-order model, as seen in Figure 5.3. From Table 5.2 we can see that the correlations between λ_1 and all 10 other components in the Gaussian approximation are close to zero, which explains why the relationship of λ_1 in the two chains is so close to being deterministic. For the parameter θ, the third-order model still provides the most efficient estimate, but as we can see in Figure 5.4, the relationship between the coupled chains is not as tight as for λ_1.

Table 5.1 shows one more estimate labeled “Precise Estimate”, which was obtained by running 200 pairs of coupled chains for 1000 iterations and discarding the first 100 states of each. We fit a third order model for the first pair of chains and with these fixed coefficients we found 199 estimates based on the other chains using (5.18). Due to the fact that the coefficients are fixed, these 199 estimates are unbiased. The precise estimate is taken to be the average of these 199 unbiased estimates, and the standard error for this estimate is found using their sample variance. The result is much more accurate than any of the three estimates, and hence can be used to evaluate their accuracy. We illustrate in

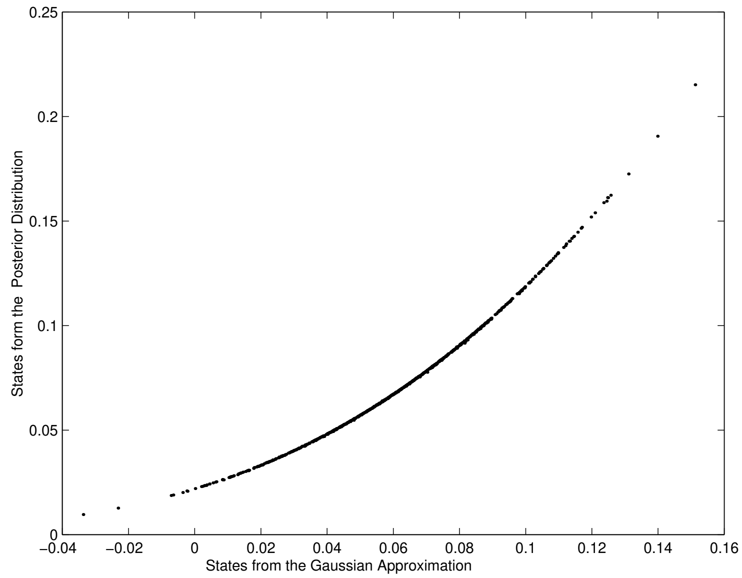


Figure 5.3: Plot showing the relationship between λ_1 values for the two coupled chains

Figure 5.5 the performance of the regression-based estimates for four of the parameters $\theta, \lambda_1, \lambda_2, \lambda_8$. As previously noted, the estimates based on the third order regression provide much tighter confidence intervals than the other two estimates. All intervals contain the precise estimate which is evidence that the standard errors for the estimates are correct.

For the 200 pairs of coupled chains we also calculated the estimates and standard errors for all the parameters based on simple linear regression and on third-order regression. For each parameter, we constructed 95% and 90% confidence intervals around these estimates by taking as margin of error the estimated standard error multiplied by the corresponding quantile for the standard Gaussian distribution. We then found the coverage probabilities for these confidence intervals, as the proportion of intervals that contain the precise estimate. As seen in Figure 5.6, these coverage probabilities are close to the desired values of 95% and 90% and mostly within the 95% confidence intervals based on a binomial distribution. This confirms that the estimators are close to being unbiased and that their standard errors are close to being correct.

		Estimate Based on			Relative Efficiency		
Parameter	Precise Estimate	One Chain ($\overline{y}$)	First-order Model ($\widehat{\mu}^{(1)}$)	Third-Order Model ($\widehat{\mu}^{(3)}$)	$\overline{y}$ vs. $\widehat{\mu}^{(1)}$	$\overline{y}$ vs. $\widehat{\mu}^{(3)}$	$\widehat{\mu}^{(1)}$ vs. $\widehat{\mu}^{(3)}$
θ	2.4895321	2.4674625	2.4874835	2.4884441			
	0.0002776	0.0303696	0.0064195	0.0042179	22	52	2.3
λ_1	0.0702695	0.0708782	0.0701613	0.0702695			
	0.0000003	0.0008800	0.0001644	0.0000056	29	24000	850
λ_2	0.1541290	0.1573713	0.1545877	0.1542453			
	0.0000057	0.0028258	0.0007910	0.0000655	13	1900	150
λ_3	0.1040727	0.1052354	0.1039408	0.1040741			
	0.0000007	0.0013133	0.0002370	0.0000120	31	12000	390
λ_4	0.1232198	0.1233812	0.1232257	0.1232186			
	0.0000005	0.0010237	0.0001227	0.0000071	70	21000	300
λ_5	0.6264700	0.6309189	0.6243196	0.6260541			
	0.0000333	0.0098097	0.0021989	0.0004975	20	390	20
λ_6	0.6133804	0.6089641	0.6134456	0.6133659			
	0.0000084	0.0042671	0.0004532	0.0001238	89	1200	13
λ_7	0.8240495	0.8280188	0.8229350	0.8264169			
	0.0001540	0.0191607	0.0054209	0.0019331	12	98	7.9
λ_8	0.8242431	0.8505378	0.8292412	0.8247778			
	0.0001599	0.0193552	0.0071622	0.0021677	7.3	80	11
λ_9	1.2951942	1.3148071	1.2966271	1.2939680			
	0.0000974	0.0224054	0.0046176	0.0014573	24	240	10
λ_{10}	1.8407347	1.8321169	1.8416890	1.8401319			
	0.0000594	0.0143102	0.0017195	0.0008827	69	260	3.8

Table 5.1: The estimates based on 900 states of the coupled chains and their standard errors, along with the relative efficiencies rounded to two significant digits.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_0	1.0000	-0.0205	-0.0596	-0.0302	-0.0247	-0.1952	-0.1065	-0.2737	-0.2737	-0.3437	-0.2836
x_1	-0.0205	1.0000	0.0012	0.0006	0.0005	0.0040	0.0022	0.0056	0.0056	0.0070	0.0058

Table 5.2: Correlations between x_0 (θ) and x_1 (λ_1) and all the other components of the Gaussian approximation.

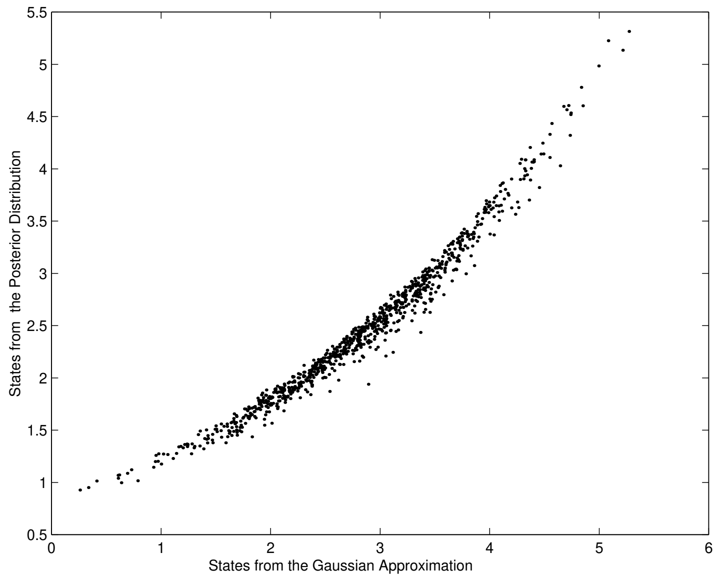


Figure 5.4: Plot showing the relationship between the θ values for the two coupled chains.

As we reduce the length of the chains, we would expect that bias may be present. Also, since the procedure for estimating the standard errors doesn't take into account the variability of the regression coefficients, the standard errors will be underestimated for short chains. Since these problems are expected to be worse when there are many regression coefficients, we recommend using the estimates based on simple linear regression when the chains are short.

5.6 Discussion

We have shown that estimators based on coupling to a chain that samples from a Gaussian approximation can be much more precise than the traditional Markov chain Monte Carlo estimators based on one chain. This method is applicable to those Bayesian problems for which the posterior distribution can be approximated reasonably well with a Gaussian distribution. The success of this method is dependent on use of a good coupling tech-

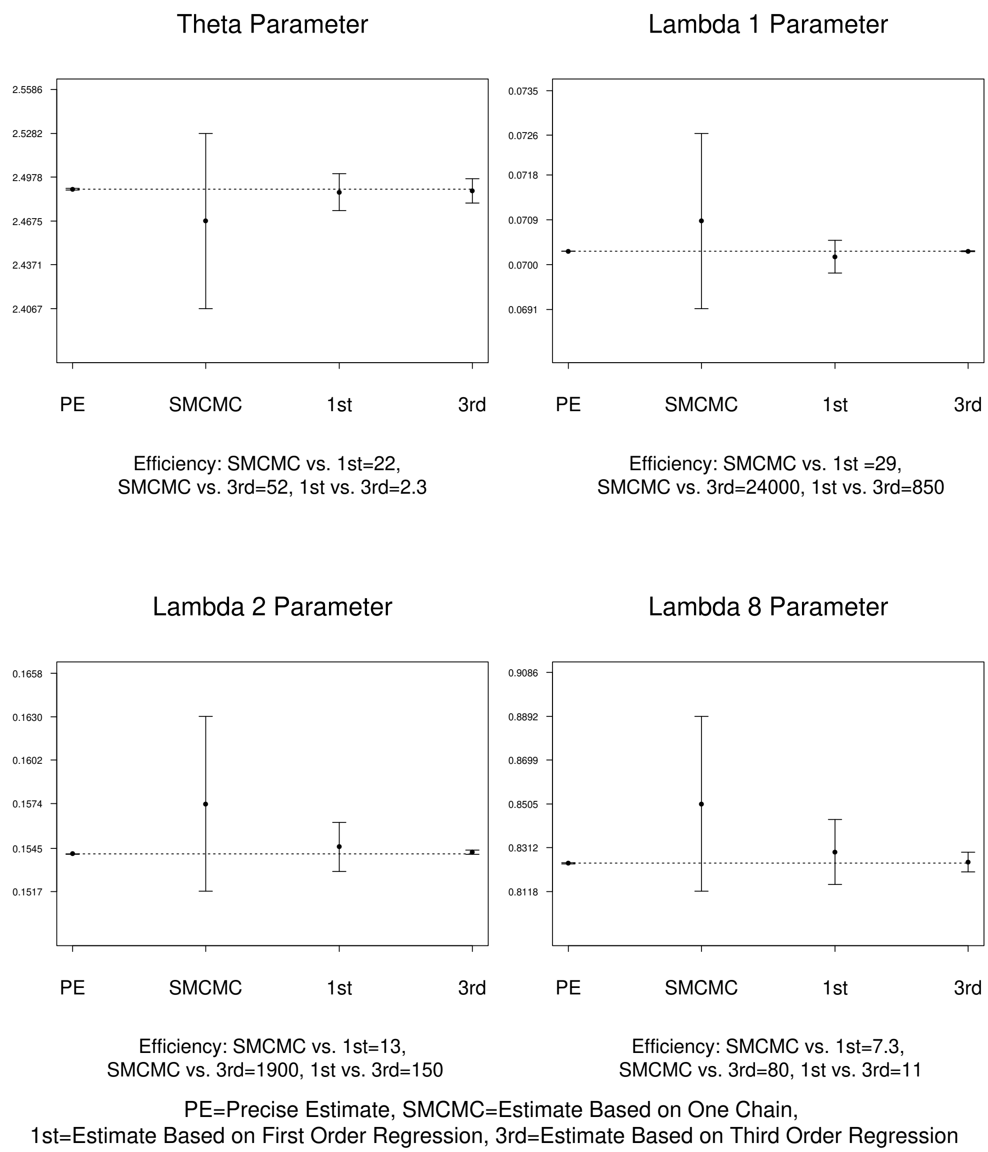


Figure 5.5: Estimates for the posterior means with 95% C.I. obtained by the MCMC methods.

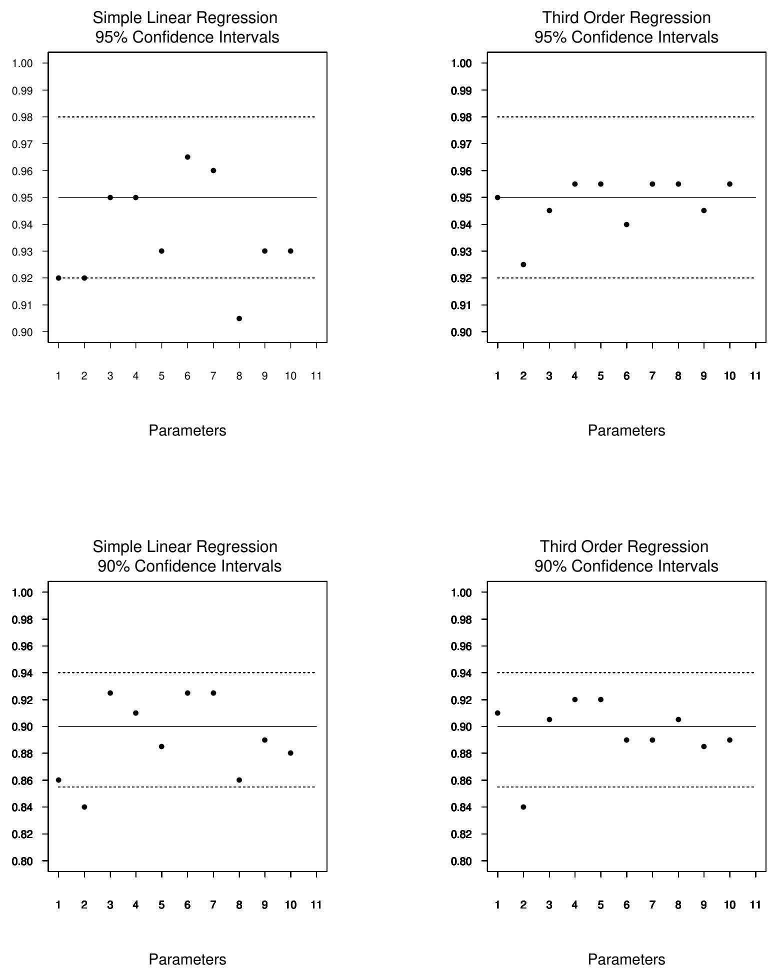


Figure 5.6: Estimates of the coverage probabilities for the 95% and 90% confidence intervals obtained as the fraction of the two hundred 95% and 90% confidence intervals that contain the precise estimate. Each confidence interval was determined from an estimate based on a pair of chains 900 iterations long. The dotted line represents the limits of a 95% confidence interval based on the binomial distribution.

nique. The two coupled sampling techniques used in this chapter, Gibbs sampling and the Metropolis algorithm, both have computational drawbacks. Gibbs sampling seems to produce samples that are highly correlated, but at the expense of having to compute inverse cumulative distribution functions, which for some conditional distributions might be expensive. Moreover, for more difficult problems, the conditional distributions will not be available, and therefore Gibbs sampling will not be applicable. For the Metropolis algorithm, the inefficiency is introduced by the small step size needed to keep the rejection rate small and therefore the correlation between chains high. These inefficiencies can be avoided by using other sampling techniques, such as higher-order Langevin methods, which can produce very low rejection rates using reasonable step sizes.

Finding the mean and the variance-covariance matrix for the Gaussian approximation requires time of order m^3, where m is the number of parameters, and one full Gibbs sampler scan for the Gaussian approximation requires time of order m^2. The methods of this chapter are therefore probably not useful when m is more than a few hundred.

Chapter 6

Conclusions

The methods developed in this thesis have been shown to provide significant improvements in MCMC estimators. Further research could lead to additional improvements.

The random sequence overrelaxation technique and Jacobian overrelaxation could potentially be combined, such that non-Gaussian distributions with highly negatively-correlated components could benefit from it. The parameter S in Jacobian overrelaxation and the product of parameters rsN in random sequence overrelaxation have the same role of updating the chain such that we will move a large distance in the long direction. Therefore one way of choosing S for JO is to take it equal to rsN.

One method for coupling JO that we have tried has not achieved high correlations between the two samples. If Jacobian overrelaxation proposes different states in the two chains, the chains will not move together from that point on. There is also the chance that the two chains will move in different directions due to the fact that the initial states of the one step JO for the two chains although close to each other, maybe on the opposite side of the conditional mean.

We have tried coupling by starting two chains in two states far apart and using the

Metropolis algorithm for some steps followed by one Jacobian overrelaxation step. We assess the coupling performance of this procedure by running a few chains using different starting points and calculating the correlation between the end points produced by each pair of chains. Metropolis updates were used in order to bring the states together locally. We have test this on a Gaussian distribution with highly positively-correlated components, but the correlation between the two samples was not high and it becomes smaller as we increase the dimensionality of the problem. We also tried to couple the chains better by projecting the points on the Jacobian trajectory on the long axis and ordering and then select the next point. This approach was also not very successful because if the starting points in the Jacobian step are on the different side of the conditional mean the chains move in different directions. Future work could be carried out to fine tune the way the next state is chosen in the Jacobian overrelaxation that will help the two chains to move to states closer together.

Methods similar to those we have presented in Chapter 5, can be applied to problems where samples from several similar distributions are needed. These problems arise when assessing the effect on the posterior distribution of deleting observations (Peruggia 1997) or changing the prior or likelihood (Gelman, Carlin, Stern and Rubin 1995a, Chapter 12). These authors use importance sampling to obtain estimates for expected values with respect to all distributions by drawing a sample from one of the distributions, and then reweighting these sample points to reflect the other distributions. Unfortunately, the importance weights can vary wildly when the distributions are too different. We propose the following strategy for estimating expectations with respect to many different, but similar, distributions. Simulate a long Markov chain converging to one of the distributions, from which a precise estimate for the expected values of the parameters with respect to this distribution can be found. For the other distributions, run short chains coupled with the first part of the long chain, and then use the methods presented in Sections 5.3 and 5.4 to find accurate estimates of the expected values of the parameters with respect to these other distributions, taking advantage of the precise estimates from the long chain.

Bibliography

Adler, S. L. (1981). Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions, *Physical Review D* **23**: 2901–2904.

Amit, Y. (1991). On rates of convergence of stochastic relaxation for Gaussian and non-Gaussian distributions, *Journal of the Mutlivariate Analysis* **38**: 82–99.

Amit, Y. and Grenander, U. (1991). Comparing sweep strategies for stochastic relaxation, *Journal of the Mutlivariate Analysis* **37**: 197–222.

Barone, P. and Frigessi, A. (1990). Improving stochastic relaxation for Gaussian random fields, *Probability in the Engineering and Informational Sciences* **4**: 369–389.

Brown, F. R. and Woch, T. J. (1987). Overrelaxed heat-bath and Metropolis algosrithms for accelerating pure gauge Monte Carlo calculations, *Physical Review Letters* **58**: 2394–2396.

Chan, K. S. (1993). Asymptotic behaviour of the Gibbs sampler, *Journal of the American Statistical Association* **88**(421): 320–326.

Chen, M.-H., Shao, Q.-M. and Ibrahim, J. G. (2000). *Monte Carlo Methods in Bayesian Computation*, Springer-Verlag.

Cheng, R. (1978). Analysis of simulation experiments under normality assumptions, *Journal of Operational Research Society* **29**: 493–497.

Duane, S., Kennedy, A., Pendleton, B. and Roweth, D. (1987). Hybrid Monte Carlo, *Physics Letters B* **195**: 216–222.

Fodor, Z. and Jansen, K. (1994). Overrelaxation algorithm for coupled gauge-higgs systems, *Physics Letters B* **331**: 119–123.

Frigessi, A., Gåsemyr, J. and Rue, H. (2000). Antithetic coupling of two Gibbs sampler chains, *The Annals of Statistics* **28**: 1128–1149.

Gelfand, A. E. . and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities, *Journal of the American Statistical Association* **85**: 398–409.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. . (1995a). *Bayesian Data Analysis*, London: Chapman & Hall.

Gelman, A., Roberts, G. O. and Gilks, W. R. (1995b). Efficient Metropolis jumping rules, *in* J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith (eds), *Bayesian Statistics 5*, Oxford: Oxford University Press.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images, *IEEE Transactions of Pattern Analysis and Machine Intelligence* **6**: 721–741.

Gilks, W. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling, *Applied Statistics* **41**: 337–348.

Green, P. J. and Han, X.-L. (1992). Metropolis methods, Gaussian proposals and antithetic variables, *in* P. Barone, A. Frigessi and M. Piccioni (eds), *Stochastic Models,* *Statistical Methods, and Algorithms in Image Analysis*, Lecture Notes in Statistics, Berlin: Springer-Verlag.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications, *Biometrika* **57**: 97–109.

Lavenberg, S. S., Moeller, T. L. and Welch, P. D. (1982). Statistical results on control variables with application to queueing network simulation, *Operations Research* **30**: 182–202.

Liu, J. S. and Wu, Y. N. (1999). Parameter expansion for data augmentation, *Journal of the American Statistical Association* **94**: 1264–1274.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines, *The Journal of Chemical Physics* **21**: 1087–1092.

Murthy, M. N. (1967). *Sampling Theory and Methods*, Statistical Publishing Society.

Neal, R. M. (1998). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation, *in* M. I. Jordan (ed.), *Learning in Graphical Models*, Dordrecht: Kluwer Academic Publisher, pp. 205–228.

Neal, R. M. (2003). Slice sampling, *Annals of Statistics (to appear).*

Nummelin, E. (1984). *General Irreducible Markov Chains and Non-Negative Operators*, Cambridge Univ. Press.

Peruggia, M. (1997). On the variability of case-deletion importance sampling weights in the bayesian linear model, *Journal of the American Statistical Association* **92**: 199–207.

Propp, J. C. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics, *Random Structures and Algorithms* **9**: 223–252.

Ripley, B. D. (1987). *Stochastic Simulation*, New York: Wiley.

Roberts, G. O. and Polson, N. G. (1994). On the geometric convergence of the Gibbs sampler, *Journal of the Royal Statistical Society B* **56**: 377–384.

Roberts, G. O. and Sahu, S. (1997). Updating schemes, correlation structure, blocking and parametrization for the Gibbs sampler, *Journal of the Royal Statistical Society B* **59**: 291–317.

Roberts, G. O. and Smith, A. F. M. (1994). Simple conditions for the convergence of Gibbs sampler and Hastings-Metropolis algorithm, *Stochastic Process and Their Applications* **49**: 207–216.

Roberts, G. O., Gelman, A. and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms, *Annals of Applied Probability* **7**: 110–120.

Rosenthal, J. S. (1995a). Minorization conditions and convergence rates for Markov chain Monte Carlo, *ournal of the American Statistical Association* **90**: 558–566.

Rosenthal, J. S. (1995b). Convergence rates for Markov chains, *SIAM Review* **37**: 387–405.

Schmeiser, B. and Chen, M.-H. (1991). On random-direction Monte Carlo sampling for evaluating multidimensional integrals, *Technical Report SMS91-1*, Dept. of Statistics, School of Industrial Engineering, Purdue University.

Tanner, M. A. (1993). *Tools for Statistical Inference*, second edn, New York: Springer Verlag.

Thisted, R. A. (1988). *Elements of Statistical Computing*, Chapman and Hall/CRC.

Tierney, L. (1994). Markov chains for exploring posterior distributions, *The Annals of Statistics* **22**: 1701–1762.

Whitmer, C. (1984). Over-relaxation methods for Monte Carlo simulations of quadratic and multiquadratic actions, *Physical Review D* **29**: 306–311.

Young, D. M. (1971). *Iterative Solution for Large Linear Systems*, New York: Academic Press.