Advances and Applications in Perfect Sampling

by

Ulrike Schneider

M.S., University of Vienna, Austria, 1999

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2003

Schneider, Ulrike (Ph.D., Applied Mathematics)

Advances and Applications in Perfect Sampling

Thesis directed by Prof. Jem Corcoran

Perfect sampling algorithms are Markov Chain Monte Carlo (MCMC) methods without statistical error. The latter are used when one needs to get samples from certain (non-standard) distributions. This can be accomplished by creating a Markov chain that has the desired distribution as its stationary distribution, and by running sample paths "for a long time", i.e. until the chain is believed to be in equilibrium. The question "how long is long enough?" is generally hard to answer and the assessment of convergence is a major concern when applying MCMC schemes. This issue completely vanishes with the use of perfect sampling algorithms which – if applicable – enable exact simulation from the stationary distribution of a Markov chain.

In this thesis, we give an introduction to the general idea of MCMC methods and perfect sampling. We develop **advances** in this area and highlight **applications** of these advances to two relevant problems.

As advances, we discuss and devise several variants of the well-known Metropolis-Hastings algorithm which address accuracy, applicability, and computational cost of this method. We also describe and extend the idea of slice coupling, a technique which enables one to couple continuous sample paths of Markov chains for use in perfect samplingalgorithms.

As a first application, we consider Bayesian variable selection. The problem of variable selection arises when one wants to model the relationship between a variable of interest and a subset of explanatory variables. In the Bayesian approach one needs to sample from the posterior distribution of the model and this simulation is usually carried out using regular MCMC methods. We significantly expand the use of perfect sampling algorithms within this problem using ideas developed in this thesis. As a second application, we depict the use of these methods for the interacting fermion problem. We employ perfect sampling for computing self energy through Feynman diagrams using Monte Carlo integration techniques.

We conclude by stating two open questions for future research.

Acknowledgements

Well ... it's been great. Was I asked to summarize the past four years in one sentence, I would probably come up with something along the lines of "I never thought that getting a Ph.D. could be so much fun." And I am serious about that!

Being a little lost after my first year, my research interests started to take better shape when I took Anne's Markov chain class in the fall of 2000. It was in this course that I stumbled out of the deterministic world and into a stochastic universe, where I ran into "way cool" things, such as perfect sampling – which became the topic of my thesis. In a run of good luck, along with this exciting research area came the perfect advisor, who guided, supported, and encouraged me throughout the process. Thank you for everything, Jem!

I found that the Applied Math department provided a truly friendly and safe atmosphere for me to grow professionally. I enjoyed sharing offices, discussing homework problems, research topics and many, many other things with great people – including Rian, Scott, Dave, Stefan, Paul, Neil, and Dan. You played a big part in making me feel at home.

Many patiently put up with me during the last month before my defense – especially Jem, my officemates, and Christophe. A big "thank you" also goes to Reinhard for reading through my entire thesis.

And finally, I want to mention "everybody" back home, my family and my friends, who didn't forget me (and even came to visit) as I moved away thousands of miles to go to grad school ... thank you!

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

This thesis is about methods in stochastic simulation, more precisely about **exact** (*perfect*) sampling methods for *Markov chain Monte Carlo (MCMC) algorithms.* So, what is MCMC and how does perfect sampling fit into the picture? The following sections describe some of the history and fundamental ideas behind these techniques.

1.1 Monte Carlo Methods

The initial motivation for the development of MCMC methods was for the use in *Monte Carlo methods* in statistical physics. Monte Carlo methods can be described to "encompass any technique of statistical sampling employed to approximate solutions to quantitative problems" [40]. The idea of exploiting random events for numerical computations has been used long before its use in statistical physics and has been documented, for example, in 1873, when Hall [29] published his attempts to estimate the number π by throwing needles on a sheet of graph paper.

While the Monte Carlo method is credited to John von Neumann and Stanislaw Ulam, it was presented systematically for the first time when Nicolas Metropolis and Stanislaw Ulam published their famous paper "The Monte Carlo Method" [36] in 1953. The name "Monte Carlo" was inspired by the city of Monte Carlo in the principality of Monaco, famous for its gambling houses and source of many random number generators, such as the roulette wheel.

More formally, the idea of Monte Carlo can be illustrated by the following example. Assume we would like to approximate the (deterministic) integral

$$\int_A g(x)\,dx.$$

We rewrite this integral as

$$\int_A \frac{g(x)}{f(x)} f(x)\,dx \quad \text{with} \quad \int_A f(x)\,dx = 1$$

and observe that it can be interpreted as

$$\mathsf{E}[\frac{g(X)}{f(X)}], \quad \text{where } X \sim f(x).$$

Then, by the Central Limit Theorem, we can approximate the desired quantity by

$$\sum_{i=1}^{n} \frac{g(X_i)}{f(X_i)}, \quad \text{where } X_i \overset{iid}{\sim} f(x).$$

(We describe this idea in more detail in Section 6.3.)

There are countless ways to interpret an integral as an expected value and usually the induced distribution from which we need to sample is non-standard. MCMC methods were introduced for the purpose of simulating the required values $X_i \sim f(x)$.

1.2 Markov Chain Monte Carlo Methods

Robert and Casella [48] give the following definition:

> A Markov chain Monte Carlo (MCMC) method for the simulation of a distribution f is any method producing an ergodic Markov Chain $(X^{(t)})$ whose stationary distribution is f.

MCMC methods essentially have two different applications. One is to assume a given distribution from which we wish to simulate and to "artificially" create a Markov chain that is designed to have the desired distribution as its stationary measure.

The other implementation concerns starting with a Markovian process given by a transition law and wishing to obtain samples from the (unknown) stationary distribution. In this thesis, we focus on the first approach, although we give an example in Section 4.5.3 where we use the second implementation.

1.2.1 Everything is Metropolis-Hastings!

As mentioned above, MCMC evolved with the object of sampling from a given distribution. The main MCMC chain for this purpose is the *Metropolis-Hastings chain* with cleverly designed transition updates which ensure that the target distribution is indeed the stationary distribution of the process. This algorithm was first introduced by Metropolis and Ulam [36] in the above mentioned paper and was later generalized by Hastings [30] in 1970. Another commonly used algorithm is the *Gibbs sampler* due to Geman and Geman [21] in 1984 and Gelfand and Smith [19] in 1990. While there are many ways to construct MCMC chains that will converge to the desired distribution, all of them (including the Gibbs sampler) are special cases of the general Metropolis-Hastings framework.

MCMC methods have had an enormous practical impact since the 1970's as they provide a large scope for statistical modeling. They have proven to be effective in areas such as spatial statistics, image analysis, Bayesian statistics, operations research, economics, and many others.

1.2.2 The Drawback

Implementing MCMC is based on the following concept. Since we have a Markov chain with a given transition law and would like to simulate from its equilibrium distribution, say, π, a natural procedure is to start the chain at some value at time $t = 0$, to move forward according to the transition probabilities, and to stop after a "reasonable" number of time steps. Of course, if we wanted to make sure that the samples are distributed according to π, we would have to move forward to time $t = \infty$! Thus the simulator has to settle for approximate samples, stopping the chain after a certain amount of time steps which is assumed to be "sufficient".

But ... how long is "long enough"? What is a "reasonable" number of time steps? When is the sample "close enough" to π? These questions are generally hard to answer and the assessment of convergence can be a major concern in the use of MCMC methods.

1.3 Perfect Sampling

The issue of convergence that appear in regular MCMC methods vanishes completely with the use of perfect simulation algorithms.

Perfect sampling is MCMC without statistical error!

Perfect sampling (*perfect simulation*, *exact simulation*, *backward coupling*, *coupling from the past* or *CFTP*) algorithms enable exact simulation of the invariant (or stationary) measure π of a Markov chain, either exactly (that is, by drawing a random sample known to be from π) or approximately, but with computable order of accuracy. These were sparked by the seminal paper of Propp and Wilson [46] in 1996, and several variations and extensions of this idea have appeared since.

The essential idea of these approaches is to find a random epoch $-T$ in the past such that, if we construct sample paths (according to a transition law of the chain that is invariant for π) from every point in the state space starting at $-T$, then all paths will have coupled successfully by time zero. The common value of the paths at time zero is a draw from π. Intuitively, it is clear why this result holds with such a random time T. Consider a chain starting at $-\infty$ with the stationary distribution π. At every iteration it maintains the distribution π. At time $-T$ it must pick **some** value x, and from then on it follows the trajectory from that value. But of course it arrives at the same place at time zero no matter what value x is picked at time $-T$, so that the value returned by the algorithm at time zero must itself be a draw from π. This idea will be discussed in further detail in Chapter 2.

1.4 About This Thesis

In this thesis we deal with advances of perfect sampling methods and present two relevant problems where we apply these techniques.

While there was a large boom in the development of perfect simulation algorithms after Propp and Wilson's article [46] in 1996, the progression of theory in this area has slowed down since then and the attention has been turned to applications of these schemes. Applying perfect sampling procedures is a non-trivial task and the current challenge has shifted towards depicting how these methods may be used in specific problems.

We present our **advances** in Chapter 3 and Chapter 4 and describe the two **applications** in Chapter 5 and Chapter 6. More specifically, this thesis is outlined as follows.

Chapter 2 is a tutorial chapter on perfect simulation with a brief review on some of the existing literature.

We discuss the Metropolis-Hastings chain in detail in **Chapter 3** and present an existing perfect version, the *perfect IMH algorithm.* We provide some new observations that allow to apply this method to the problems described in Chapter 5 and Chapter 6. We also specify a (non-perfect) *adaptive Metropolis-Hastings scheme* that, due to its adaptive nature, appears to converge very rapidly to the desired distribution. Also see [7].

In **Chapter 4** we describe and develop a variation of the *layered multishift coupler* due to Wilson that allows one to obtain potentially common draws from two different distributions. Our main application is coupling sample paths of Markov chains for use in perfect sampling algorithms and we apply this technique in Chapter 5. We present several variants of this idea, discuss implementation issues, and illustrate these methods with examples such as simulating from a storage model and the auto-gamma distribution. Also see [8].

We illustrate the use of perfect sampling algorithms for Bayesian variable selection in a linear regression model in **Chapter 5**. Starting with a basic case solved by Huang and Djurić (2002), where the model coefficients and noise variance are assumed to be known, we generalize the model step by step to allow for other sources of randomness. We specify perfect simulation algorithms that solve these cases by incorporating various techniques including Gibbs sampling, a variant of the perfect IMH algorithm from Chapter 3, and slice coupling algorithms from Chapter 4. Applications to simulated data sets suggest that our algorithms perform well in identifying relevant predictor variables. We also show results for a real data set. Also see [52].

Another application of the IMH algorithm from Chapter 3 is considered in **Chapter 6** where we estimate self energy of the interacting fermion model via Monte Carlo summation. Simulations suggest that the algorithm in this context converges extremely rapidly and results compare favorably to true values obtained by brute force computations for low dimensional toy problems. A variant of the perfect sampling scheme which improves the accuracy of the Monte Carlo sum for small samples is also given. Also see [9].

We conclude this thesis in **Chapter 7** where we give a summary over previous chapters and tie up loose ends by stating ideas for future research.

Appendix A contains a description of the Gibbs sampler which is used several times throughout this thesis. **Appendix B** concerns *hierarchical sampling*, used in Chapter 5.

Chapter 2

Perfect Sampling

This chapter contains a description of perfect sampling and a literature review of advances in this area. We also refer the reader to D.B. Wilson's website [59] which contains many links to research papers and even an FAQ page for perfect sampling! Tutorials can be found, for example, by Robert and Casella [4] and Thönnes [55].

2.1 Illustration of the Basic Idea

As introduced in Chapter 1, perfect sampling, if applicable, enables exact simulation from the stationary measure of a Markov chain with a given transition law.

Markov Chains

A *Markov chain* $\{X_t\}$ with state space X is a stochastic process on X, indexed over time (for illustration purposes, we assume $t \in \mathbb{N}$), which satisfies

$$P(X_{n+1} = x_{n+1} | X_n = x_n, \ldots, X_0 = x_0), \qquad x_i \in \mathsf{X},\ i = 1, \ldots, n. \tag{2.1}$$

(2.1) is known as the *Markov property* and essentially states that the distribution of future states is completely determined by the present state of the chain. The right hand side of (2.1), specified for all states, constitutes the *transition rule* of $\{X_t\}$. The *stationary distribution* of the process is one that is invariant with respect to the transition rule, meaning that if the chain is in its stationary distribution, the updates given by (2.1) will preserve that distribution. We will consider only Markov chains that will converge to their unique stationary distribution over time. The specific class of chains that are of interest to us is discussed in more detail below.

MCMC and Perfect sampling

While in regular MCMC methods, one chain is started at some value at time $t = 0$ (and run forward for a certain number of time steps according to the transition rule of the chain), in perfect sampling, we can think of starting *parallel chains* in each state of the state space.

Essentially, these chains are run as if they were started at time $t = -\infty$ and stopped at time $t = 0$. Fortunately, this can also be done in finite time for certain Markov chains as discussed below.

An important observation for understanding perfect simulation is the idea of coupling sample paths of Markov chains. Once the paths of two (or more) parallel chains meet, they will, due to the Markov property, coincide from then on. We say that the chains have *coupled* or *coalesced*, and the time when this occurs is called a *forward coupling time*. Figure 2.1 illustrates this.

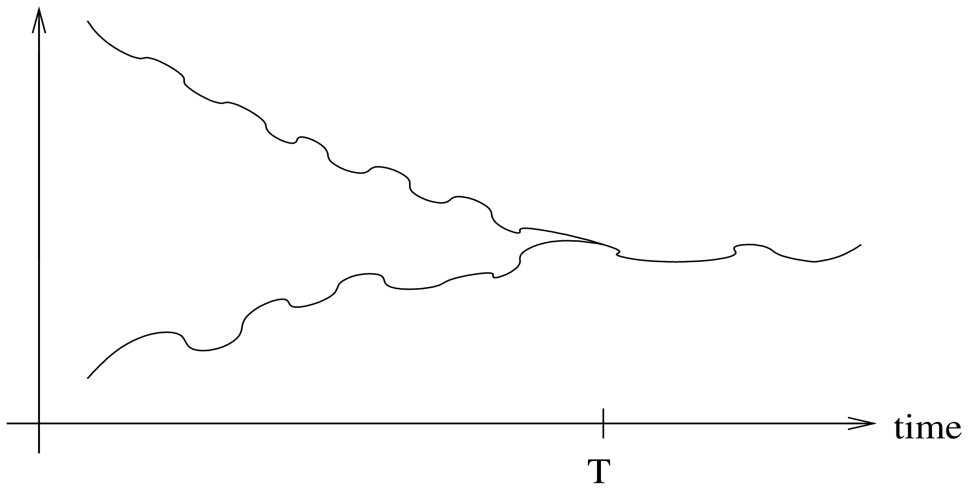


Figure 2.1: Two paths of a Markov chain coalesce at the forward coupling time T.

The basic structure of the perfect simulation procedure that underlies all of these algorithms is given in the following description.

We start parallel chains in **all** states of the state space at time $t = -1$ and check whether all paths have met at time $t = 0$. If this is not the case, we start parallel chains again at $t = -2$ and determine whether all paths have coupled by time $t = 0$, and so on. This is repeated for starting points successively further in the past until coalescence occurs for **all** chains by time $t = 0$. We call the time $t = T$ in the past that guarantees that all chains have coupled by time $t = 0$ a *backward coupling time.*

The pictures in Figure 2.2 illustrate this procedure. They show an example of using perfect sampling to obtain one draw from the stationary distribution of a Markov chain with state space $\mathsf{X} = \{s_1, s_2, s_3, s_4\}$ containing four elements. The vertical axis represents X and the horizontal axis displays the backwards time t. Coalescence occurs after going back $T = -3$ time steps.

Look at the last picture in Figure 2.2 and think of the chain having started stationary (in some state) at time $t = -\infty$. Then, by time $t = -3$ it will be **one** of the four possible states and therefore necessarily end in state s_2 at time $t = 0$. Hence s_2 represents a draw from the stationary distribution.

Note that it is essential to append on to previous sample paths or to reuse random numbers. Assume, for example, that we run sample paths forward from time $t = -1$ to time $t = 0$ using a random number (or random vector) u_{-1}, and suppose that these paths have not coalesced at time 0. We then need to go back an additional time step to start sample paths at time $t = -2$ and run the paths forward to time $t = 0$. In order to carry out these transitions, we use a random number (or random vector) u_{-2} for the transition from $t = -2$ to $t = -1$ and the **previously used** u_{-1} for the update from $t = -1$ to $t = 0$. This is also demonstrated in Figure 2.2.

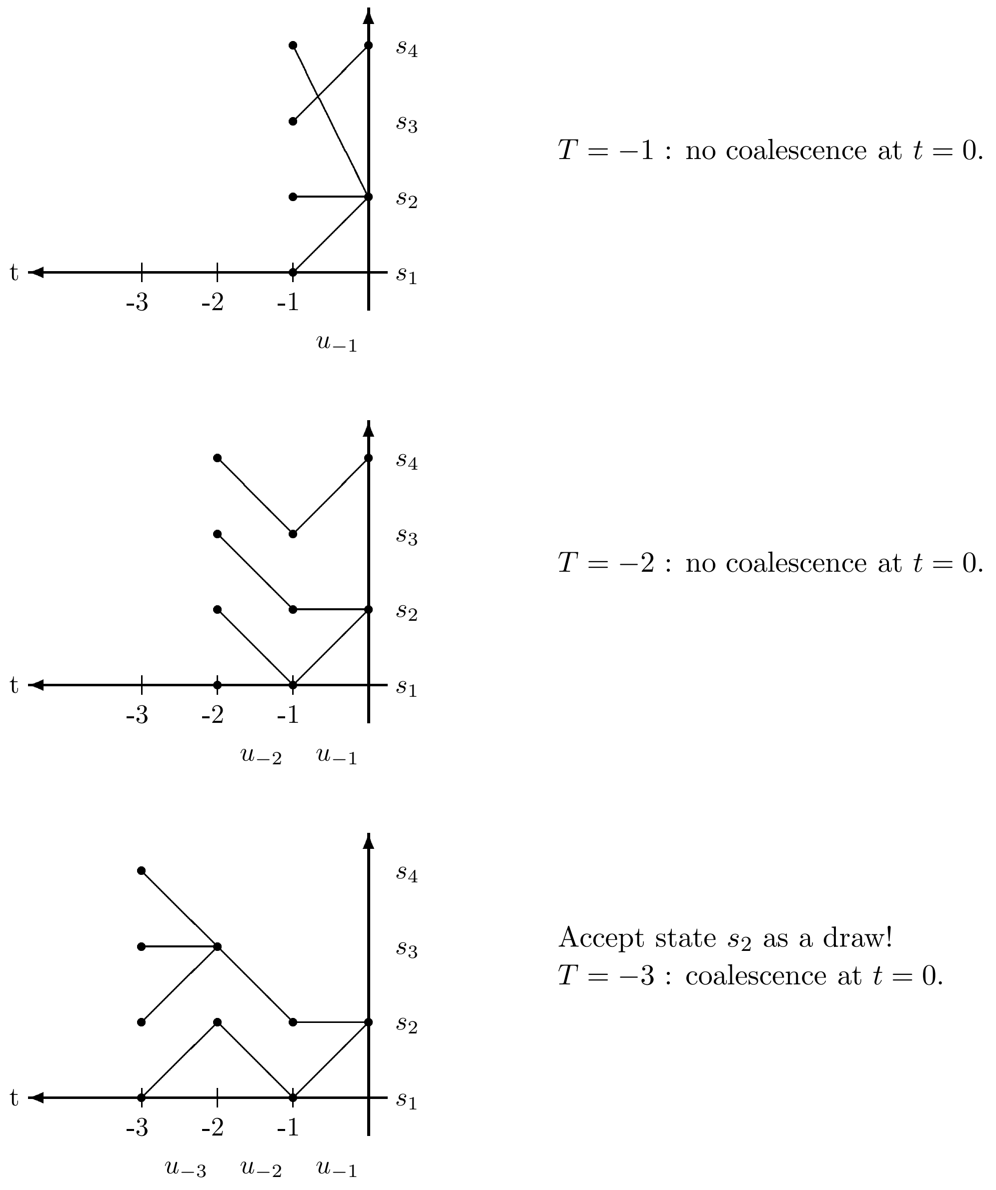
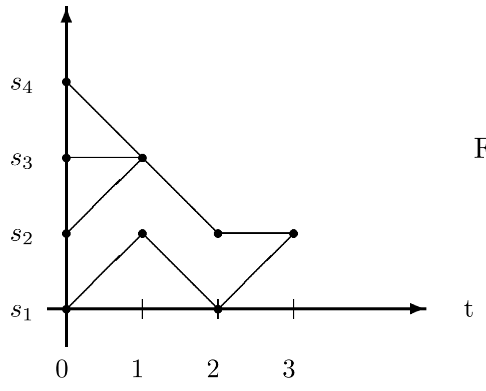

s_4
s_3
s_2
s_1
t
-3
-2
-1
u_{-1}
$T = -1$: no coalescence at $t = 0$.
s_4
s_3
s_2
s_1
t
-3
-2
-1
u_{-2} u_{-1}
$T = -2$: no coalescence at $t = 0$.
s_4
s_3
s_2
s_1
t
-3
-2
-1
u_{-3} u_{-2} u_{-1}
Accept state s_2 as a draw!
$T = -3$: coalescence at $t = 0$.

Figure 2.2: Illustration of how perfect sampling works for a 4-state Markov chain. Coalescence occurs at $T = -3$ (backward coupling time).

Why backwards?

Why not just start chains at $t = 0$ and go forward in time until they have coalesced as shown in Figure 2.3?



Forward coupling time $T = 3$.

Figure 2.3: Running parallel chains forward.

Forward and backward coupling times are different in the sense that the forward coupling time denotes the **actual time** of coalescence, whereas the backward coupling time refers to a time in the past such that **by time** $t = 0$, all chains have met. For a backward coupling time T, coalescence can occur at any time between time $t = T$ and time $t = 0$.

If we start the chain according to the stationary measure at time $t = 0$, it will stay stationary at all **fixed** time steps, but since coupling times are **random**, reporting states at these times would no longer necessarily represent draws from the desired distribution.

In perfect sampling, we always report draws at the **fixed** time $t = 0$!

When does it work?

Let $P^n(x, .)$ denote the n-step transition law of the chain and let π be its stationary measure. Perfect sampling theoretically works for all *uniformly ergodic* Markov chains, which are defined to be the ones that satisfy

$$\| P^n(x, \cdot) - \pi \| \rightarrow 0 \qquad \text{as } n \rightarrow \infty \tag{2.2}$$

uniformly over all x of the state space. The norm $\| \cdot \|$ in (2.2) refers to the *total variation norm* of a measure which is defined as

$$\| \mu \| := 2 \times \sup_{A \text{ Borel set}} | \mu(A) |.$$

Foss and Tweedie [16] show that these chains are exactly the ones that have a finite backward coupling time with probability 1.

For a Markov chain with a finite state space, uniformly ergodic is equivalent to the chain being *irreducible* (essentially meaning that any state can be reached from any other state in a finite number of time steps) and *aperiodic* (meaning that all states have period 1). In that case, a straightforward implementation of perfect simulation is possible using the procedure described above and we refer the reader to Propp and Wilson [46] for further details.

For uniformly ergodic chains with infinite or continuous state spaces the challenge remains to **detect** a backward coupling time, even if we are assured that it is finite! The following section lists some ideas for this purpose.

2.2 Extensions to Infinite and Continuous State Spaces

We now describe some ideas to extend the basic idea discussed in Section 2.1 to infinite and continuous state spaces.

Monotonicity and Bounding Processes

Perfect sampling algorithms can be particularly efficient if the chain is *stochastically monotone*, meaning that paths from lower starting points stay below paths from higher starting points with respect to some partial ordering placed on the state space. In this case, one only needs to couple sample paths from the "top" and "bottom" of the state space, as all other paths will be sandwiched in between. It is possible to generalize this concept one step further to monotone chains on an unbounded state space by considering *stochastically dominating* processes to bound the journeys of sample paths. For example, Møller [39] uses this idea to specify an “almost perfect” simulation algorithm to simulate from the so-called auto-gamma distribution up to a computable order of accuracy. (We present a perfect method for that purpose in Section 4.5.2.)

Other Ideas

Other and related ideas include Fill’s algorithm [15, 51], minorization criteria in Murdoch and Green [43], the Harris coupler in Corcoran and Tweedie [10], the horizontal backward coupling scheme in Foss et al. [17], the IMH algorithm in Corcoran and Tweedie in [11] (discussed in Chapter 3 and applied in Chapter 5 and Chapter 6), slice sampling methods in Mira et al. [37], and slice coupling techniques in Corcoran and Schneider [8] where we present procedures that force continuous sample paths to couple (this is described in detail in Chapter 4).

Chapter 3

Using the Metropolis-Hastings Chain

3.1 Introduction

The Metropolis-Hastings algorithm [56] allows simulation of a probability density $\pi(x)$ which is known only up to a factor, that is, when we have $\pi(x) = ch(x)$ with c unknown. Instead of simulating from π directly, the simulator proceeds by drawing from a *candidate* or *proposal distribution* Q with density $q(y)$ and generating transitions for a discrete-time Markov chain evolving over the state space of π.

In order to sample a value drawn from π, one must generally select a distribution Q and run a Metropolis-Hastings sample path for "a long time" until it is suspected that convergence to π has been achieved. We describe this algorithm in more detail in Section 3.2.

In Corcoran and Tweedie [11], a Metropolis-Hastings based "perfect sampling" algorithm was introduced based on the backward coupling approach of Propp and Wilson [46], eliminating the need to address issues of convergence. A drawback of this algorithm, described in Section 3.3, is that it is necessary to maximize a certain ratio of densities. We address this issue in Section 3.4.1, where we implement an "imperfect" perfect sampling algorithm by using only an approximate maximum in the algorithm of [11]. Empirical results suggest that it is worthwhile to use the imperfect backward coupling algorithm even with this introduced error, as it appears to outperform the traditional forward approach.

With a traditional Metropolis-Hastings algorithm, a perfect version, or even an imperfect perfect version, there remains the issue of choosing a suitable candidate distribution Q. Algorithms with self-targeting candidates exist [54], and in a similar spirit we introduce an "adaptive" Metropolis-Hastings algorithm in Section 3.4.2. Preliminary results suggest that this approach considerably speeds up convergence and, in fact, will converge when the traditional approach fails altogether.

3.2 The Metropolis-Hastings Algorithm

We outline the Metropolis-Hastings algorithm to draw from a distribution with density π on the state space X by describing the underlying Markov chain of the algorithm. The Metropolis-Hastings X_t chain is defined by the following transition law. Assume that the chain is currently in state x, that is $X_t = x$. Then propose a candidate value y according to the distribution Q with density $q'_{Y|X_t=x}(y|x) = q(y,x)$ and state space X and *accept* the transition to y with probability

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

Note that the candidate distribution depends on the current state of the chain. If the candidate draw is accepted, we set $X_{t+1} = y$. If we do not accept, we *reject* to move to the suggested value y and obtain $X_{t+1} = x$. This procedure creates a Markov chain on X with transition density

$$p(x,y) = q(x,y)\,\alpha(x,y), \qquad y \neq x,$$

which will remain at the same point with probability

$$P(x,\{x\}) = \int q(x,y)\,[1 - \alpha(x,y)]\,dy.$$

It is easy to verify that π is the invariant (or stationary) measure for the chain, see, for example, [26, 48, 49, 5, 50]. Choices for Q and rates of convergence have been studied extensively in [35, 48], for example.

Note that the form of the acceptance probability $\alpha(x,y)$ requires that π and q need to be known only up to a constant (as long as it is possible to generate samples from Q).

The original Metropolis algorithm, which was first proposed by Nicolas Metropolis [36] in 1953, uses a symmetric candidate transition Q for which $q(x,y) = q(y,x)$. In that case, the acceptance probability $\alpha(x,y)$ can be simplified to

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

In 1970, Hastings [30] extended the Metropolis algorithm to a more general proposal distribution Q as described above.

3.3 The IMH Chain

In this section, we focus on the so-called *independent Metropolis-Hastings* (IMH) chain, where we have a given candidate distribution Q which we will assume to have a density q, and from which we can generate i.i.d. samples. We use the term "independent" to describe the Metropolis-Hastings algorithm where candidate states are generated by a distribution that is independent of the current state of the chain. A candidate value generated according to q is then accepted with probability $\alpha(x,y)$ given by

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

where x is the current state and y is the candidate state.

It is known [35] that the IMH chain has desirable convergence properties (and, for example, is uniformly ergodic) if there exists a $\beta > 0$ such that

$$\frac{q(x)}{\pi(x)} \geq \beta. \tag{3.1}$$

We call q *overdispersed* with respect to π if (3.1) holds. This is the case if q is "heavier in the tails" than π which is illustrated in Figure 3.1.

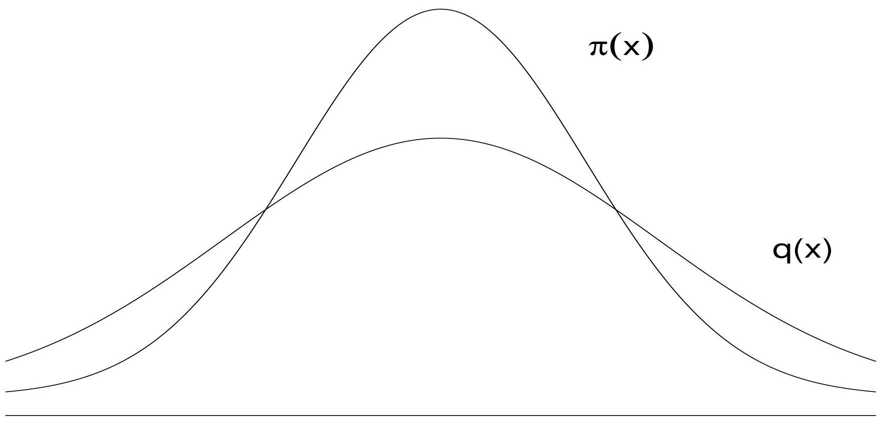


Figure 3.1: A candidate density q that is overdispersed with respect to π.

3.3.1 The Perfect IMH Algorithm

In the paragraph on "Monotonicity and Bounding Processes" in Section 2.2 we briefly explained how certain monotonicity properties can be used to implement perfect sampling on a continuous state space. We now illustrate how specific monotonicity features of the IMH scheme can be exploited to find a perfect implementation of the IMH chain – the resulting *IMH algorithm* is due to Corcoran and Tweedie [11].

The IMH algorithm uses the ratios in the acceptance probabilities $\alpha(x, y)$ to reorder the states in such a way that we always accept moves "downwards" to states that are ranked lower with respect to the ordering. More specifically, if we write $\pi(x) = c\, h(x)$ where c is unknown, we define the *IMH ordering*

$$x \succeq y \qquad \Leftrightarrow \qquad \frac{\pi(y)q(x)}{\pi(x)q(y)} \geq 1 \qquad \Leftrightarrow \qquad \frac{\pi(y)}{q(y)} \geq \frac{\pi(x)}{q(x)}. \tag{3.2}$$

With this ordering, we can think of the "lowest state" as essentially the one that is the hardest to move away from when running the IMH algorithm.

In fact, the lowest state with respect to the ordering defined in (3.2) is any $l \in \mathsf{X}$ which satisfies

$$\frac{\pi(l)}{q(l)} = \sup_{x \in \mathsf{X}} \frac{\pi(x)}{q(x)}. \tag{3.3}$$

This l also guarantees that

$$\alpha(l, y) \leq \alpha(x, y) \ \ \forall x \in \mathsf{X}. \tag{3.4}$$

Thus, if we are able to accept a move from l to a candidate state y drawn from the distribution Q with density q, then sample paths from every point in the state space will also accept a move to y, so all possible sample paths will coalesce.

Note that if such an element l satisfying (3.3) exists, (3.1) also holds and the underlying chain is uniformly ergodic. Also note that in the above requirements, the density π can be replaced by $h(x) \propto \pi(x)$ with no additional changes for the algorithm.

The perfect sampling algorithm is formally described as follows:

(1) Draw a sequence of random variables $Q_n \sim Q$ for $n = 0, -1, -2, \ldots$, and a sequence $\alpha_n \sim \text{Uniform}(0, 1)$ for $n = -1, -2, \ldots$.

(2) For each time $-n = -1, -2, \ldots$, start a lower path L at l, and an upper path, U at Q_{-n}.

(3) (a) For the lower path: Accept a move from l to Q_{-n+1} at time $-n+1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l. That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \alpha(l, Q_{-n+1})$.

(b) For the upper path: Similarly, accept a move from Q_{-n} to Q_{-n+1} at time $-n+1$ if $\alpha_{-n} \leq \alpha(Q_{-n}, Q_{-n+1})$; otherwise remain at state Q_{-n}.

(4) Continue until T defined as the first n such that at time $-n+1$ each of these two paths accepts Q_{-n+1}. (Continue the Metropolis-Hastings algorithm forward to

time zero using the α's and Q's from steps (1) - (3) to get the draw from π at time zero.)

By monotonicity, the upper path will accept a candidate point whenever the lower path will, and the two paths will be the same from that time forward. Consequently, our description of the upper process is a formality only, and, indeed, the upper process need not be run at all. Figure 3.2 illustrates a realization of the perfect IMH algorithm.

We refer the reader to [11] for details on how one's choice of Q will affect the expected backward coupling time for the perfect IMH algorithm.

3.3.2 Bounded IMH – step (3) $(a)'$

For sampling from complicated densities, we may wish to take advantage of the observation that neither the lowest state, l, nor the maximum value $\pi(l)/q(l)$ need be attained explicitly. If we are able to find a constant C such that

$$\sup_{x \in \mathsf{X}} \frac{\pi(x)}{q(x)} \leq C \qquad \forall x \in \mathsf{X}$$

then we know that

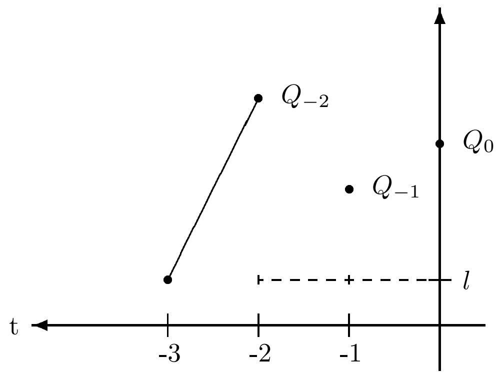
$$\frac{\pi(y)}{q(y)} C^{-1} \leq \alpha(x, y) \qquad \forall x \in \mathsf{X}$$

so we could modify step (3) (a) of the IMH algorithm to read

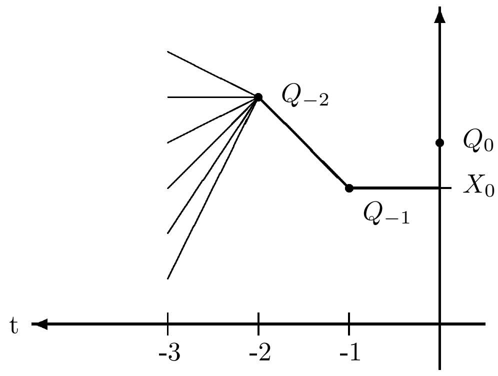
(3) $(a)'$ for the lower path: Accept a move from l to Q_{-n+1} at time $-n+1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l. That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \frac{\pi(Q_{-n+1})}{q(Q_{-n+1})} C^{-1}$.

3.4 Variants on the IMH algorithm

In this section, we present two variants on the IMH algorithm. The first one addresses the problem of not being able to maximize the π/q, the second one the question of how to choose a candidate distribution Q.



Going backwards: l accepts the candidate Q_{-2} at time $t = -2$.
$T = -3$ is a backward coupling time.



Going forward to time $t = 0$.
All possible paths will be in state $X_{-2} = Q_{-2}$ at time $t = -2$!

The first graph represents the process of finding a backward coupling time. Dashed lines show when the lowest state l rejects the given candidate and the chain stays "flat". Moving from time $t = -3$ to time $t = -2$, l accepts to move to the proposed candidate Q_{-2}, so **any** sample path at that time will accept the to move to Q_{-2} (which is illustrated in the second graph). Starting at time $t = -2$, the thick lines in the second graph represent the path moving forward to time $t = 0$ according to the Metropolis-Hastings updates.

Figure 3.2: A realization of the perfect IMH algorithm with backward coupling time at $T = -3$ and exact draw X_0.

3.4.1 An “Approximate Perfect” Algorithm

Identifying the “lowest point” l for the perfect IMH algorithm can sometimes be problematic. We now consider an imperfect variant of the perfect IMH algorithm where we use an approximate value for l. Even though this introduces error into the perfect IMH algorithm, preliminary empirical results suggest that the method is still superior to the traditional forward-time Metropolis-Hastings algorithm.

Recall that l is a point that maximizes the ratio

$$\frac{\pi(x)}{q(x)} \qquad \text{or, equivalently} \qquad \frac{h(x)}{q(x)}.$$

We could maximize this with either traditional deterministic methods or stochastic optimization methods – we describe both approaches in this section. We start out with a random search that may be built into the perfect IMH algorithm at no additional computational cost. As some may view the built-in approach as too complicated to implement, we also present the obvious approximate perfect IMH algorithm where l is approximated independently of the backward coupling algorithm as a second approach.

Built-In Maximization

At each time step, we draw a candidate and compute the corresponding π/q-ratio for the Metropolis-Hastings algorithm. But we can also use this information for the stochastic search for the maximum of π/q! If we find that the current candidate improves the current maximum, we update. To ensure progress in this search process, we go back at least T time steps for each Metropolis-Hastings iteration. After these T time steps, we stop if either of the following occurred.

(1) We improved the maximum (that implies that we actually drew the current low point that every state will accept).

(2) Coalescence occurred with respect to the current low point.

Otherwise, we go back further (past T) in time until either of the above takes place.

Algorithm

While the basic idea of the algorithm is very simple, unfortunately, its implementation is less straightforward.

We present pseudo-code showing that the different blocks of the algorithm can be collapsed into only two loops. We also wish to alert the reader that while we hope that the pseudo-code might be useful for someone wishing to implement the algorithm, we believe that it is too complicated to help in understanding the underlying idea.

We can basically split the algorithm into two parts – a backward and a forward block.

(1) **Backward block**

```
• Initialize max $= 0, t = 0$, coupled=FALSE.
• While $(t > -T)$ or (not coupled)
    * t=t-1
    * Draw a candidate $y_t \sim q(x)$ and $u_t \sim$ Uniform$[0, 1]$.
    * Set cur $= \frac{\pi(y_t)}{q(y_t)}$.
    * If cur > max
        – max = cur
        – bct $= t$
        – coupled=TRUE
        – $x = y_t$
    * If $(t > -T)$ and $(u_t <$ cur/ max$)$
        – bct $= t$
        – coupled=TRUE
        – $x = y_t$
```

(2) **Forward block**

```
• for $t = \text{bct} - 1$ to 1
    * $\text{acc} = \frac{\pi(x)\, q(y_t)}{q(x)\pi(y_t)}$
    * if $u_t < \text{acc}$
        – set $x = y_t$
```

Remark Although the algorithm is used when we don't know the maximum, we still need to know that it exists – otherwise we are not guaranteed convergence.

Simulation Results

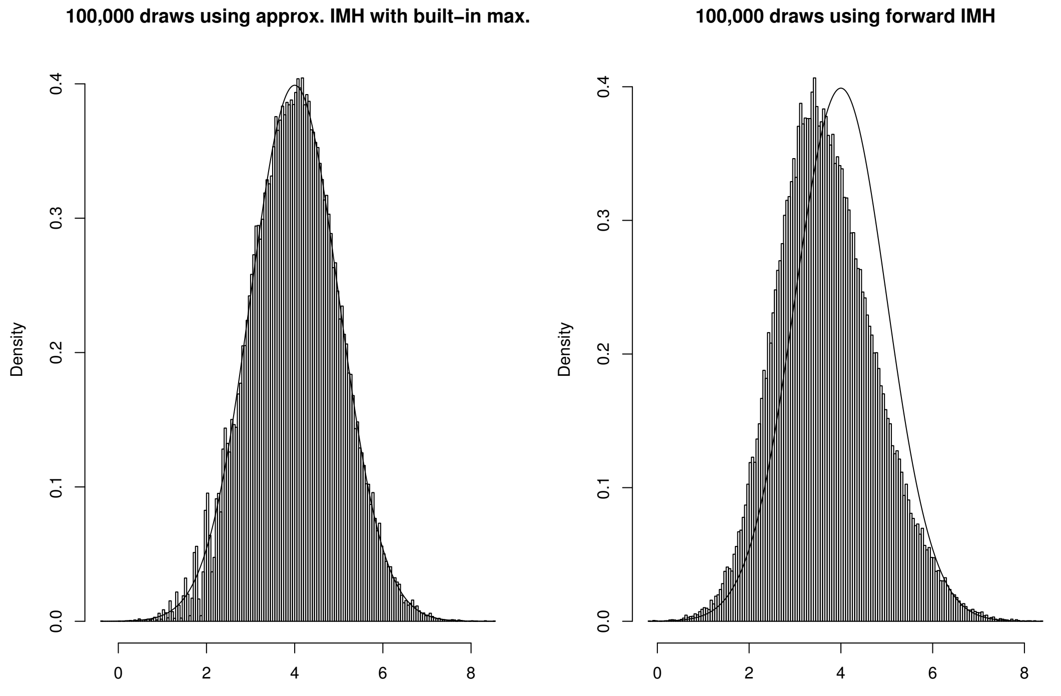
To assess the performance of the approximate IMH algorithm with a built-in maximization, we compare the output of this procedure to a regular forward IMH algorithm with approximately the same computational cost. To do so, we show the histograms of the draws from both algorithms with the target density superimposed.

In Figure 3.3, we wish to draw from a $N(4, 1)$ distribution using a double exponential candidate with rate 1 as previously specified. After 100,000 draws, the average backward coupling time was again 70 (the parameter T was set to 5), so the forward chain was run for 70 time steps for each draw to have comparable computational cost.

We also considered an example where the target (and candidate) density have bounded support.

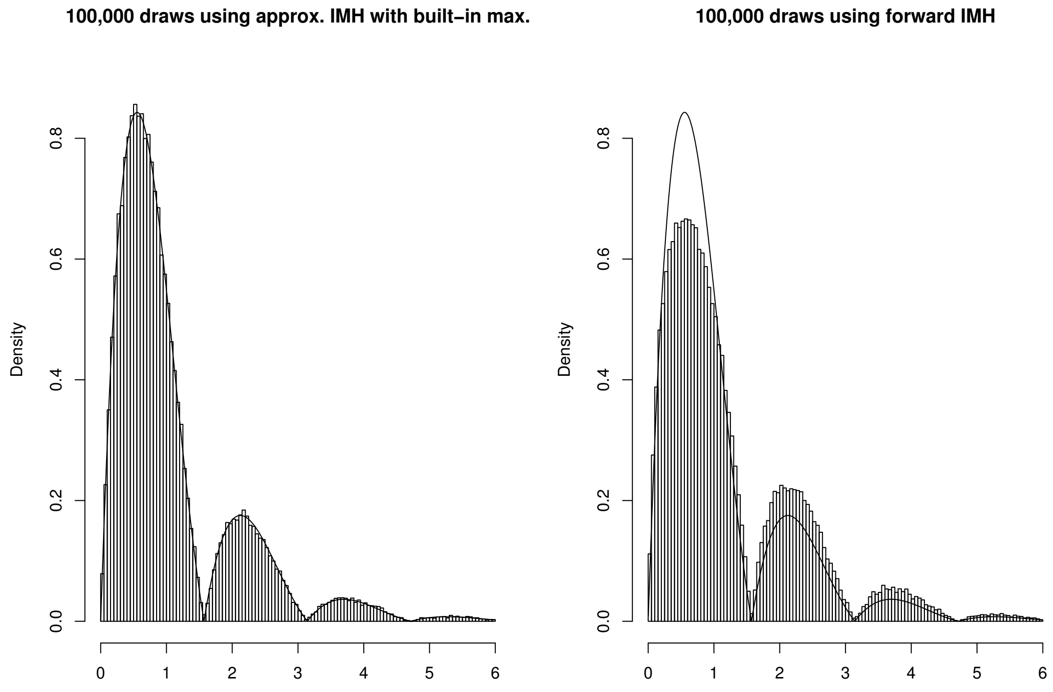
$$\begin{aligned} \pi(x) &\propto \mathbb{1}_{(0,6)} \exp\{-x\} |\sin(x)\cos(x)| \\ q(x) &\propto \exp\{-|x|\}. \end{aligned}$$

With these choices (and the parameter T set to 2), after 100,000 draws the average backward coupling was 5, so we ran the regular IMH algorithm forward for 5 time steps for each sample. The resulting histograms for the approximate IMH and the forward IMH algorithm are shown in Figure 3.4.



The target density is $N(4, 1)$ with a double exponential candidate with rate 1. Histograms show 100,000 draws with an approximate IMH algorithm with built-in maximization and a regular Metropolis-Hastings procedure. The target density is superimposed.

Figure 3.3: Comparison of an approximate IMH algorithm with built-in maximization to an independent Metropolis-Hastings scheme.



The target density is $f(x) \propto \mathbb{1}_{(0,6)} \exp\{-x\}|\sin(x)\cos(x)|$ with a uniform candidate on $(0, 6)$. Histograms show 100,000 draws with an approximate IMH algorithm with built-in maximization and a regular Metropolis-Hastings procedure. The target density is superimposed.

Figure 3.4: Comparison of an approximate IMH algorithm with built-in maximization to an independent Metropolis-Hastings scheme.

Independent Maximization

For some fixed N, we draw a sequence of values $Q_1, Q_2, \ldots, Q_n$ form the distribution Q and approximate the low point by

$$\hat{l}_N = \arg \max_{x \in \{Q_1, Q_2, \ldots, Q_n\}} \frac{h(x)}{q(x)}$$

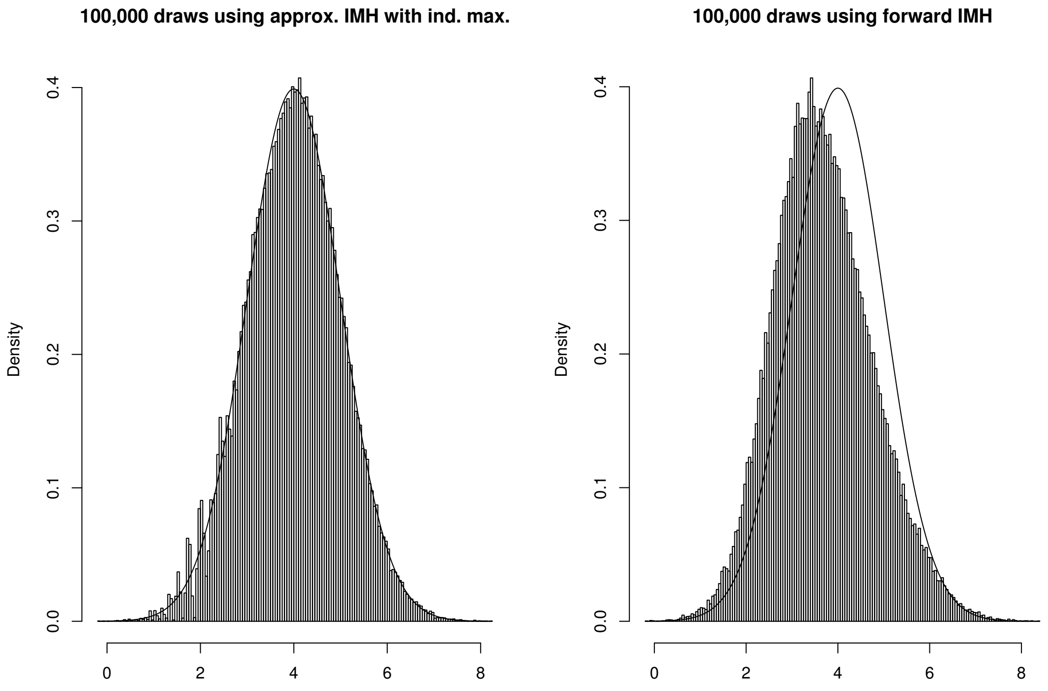
We then proceed with the IMH algorithm as described in Section 3.3.1 using $\hat{l}_N$ in place of l. Ideally, one would watch for a better estimate of l during the evaluation of h/q ratios in the IMH algorithm. Indeed, it is this advice that forms the basis for the built-in maximization described in the paragraph below.

We have assumed no knowledge of the structure of π or of h/q. Obviously, if the region containing the maximizing point can be narrowed down, one should take advantage of this. One should also take advantage of more sophisticated deterministic and/or stochastic optimizers. We have presented an approach in the spirit of parsimony, using only random deviates that must already be computable in order to run the perfect IMH algorithm.

In Figure 3.5 we compare the output of this approximately perfect algorithm to that of a traditional forward Metropolis-Hastings algorithm for a $N(4, 1)$ target density with a double exponential candidate density with rate 1. More precisely, we used

$$\begin{aligned} \pi(x) &\propto \exp\{-\frac{1}{2}(x-4)^2\} \\ q(x) &\propto \exp\{-|x|\}. \end{aligned}$$

To approximate the maximum, we simulated $N = 1000$ values from the candidate density to compute $\hat{l}_N$. As the average backward coupling time for the approximate IMH scheme turned out to be 70 after 100,000 draws, we ran a regular IMH chain forward for 70 time steps for each draw. In this “comparison” of computational cost we neglect the computational effort that has to be made to maximize π/q as we leave the choice of this procedure up to the user. Aside from this issue, Figure 3.5 suggests that



The target density is $N(4, 1)$ with a double exponential candidate with rate 1. Histograms show 100,000 draws with an approximate IMH algorithm with an independent maximization procedure and a regular Metropolis-Hastings procedure. The target density is superimposed.

Figure 3.5: Comparison of an approximate IMH algorithm with independent maximization to an independent Metropolis-Hastings scheme.

the approximate IMH algorithm does outperform the regular forward implementation.

For both examples the approximate IMH algorithm seems to outperform the regular forward IMH scheme at comparable computational effort. Moreover, Figure 3.5 seems to be very similar to Figure 3.3 despite the lower computational effort. One could use standard convergence assessment techniques, e.g. [20, 48, 2, 12] to diagnose and compare convergence – we leave this for future work.

3.4.2 An Adaptive IMH Algorithm

It is likely that any person who has ever given a presentation on the Metropolis-Hastings algorithm, perfect or otherwise, has been asked the question:

How do you choose the candidate distribution?

We usually find ourselves giving the unsatisfactory two-part response:

Choose a candidate distribution Q with density q so that:

(1) q is overdispersed with respect to the target density π, and

(2) you are able to simulate (draw) values from Q.

We realize that this is not the most helpful answer. There do, in fact, exist "self-targeting" approaches such as in Stramer and Tweedie [53, 54], and we now propose, in a similar spirit, an "adaptive" algorithm that will create a candidate for the user. We begin with a motivating example.

Suppose we wish to draw values from the distribution with density

$$\pi(x) \propto \mathbb{1}(0,\infty)(x) \exp\{-x\} \, |\sin(x) \, \cos(x)|. \tag{3.5}$$

Let us further suppose that we choose a $\Gamma(5, \frac{1}{2})$ [1] candidate. That is, we will draw candidate values from the distribution with density

$$q(x) \propto \mathbb{1}_{(0,\infty)}(x) x^4 \exp\{-\frac{1}{2}x\}.$$

In many cases, one can still get decent results from the Metropolis-Hastings algorithm even with a "bad" candidate density. We have chosen the target density (3.5) since it is a multi-modal example and have chosen a candidate density which not only fails condition (3.1) but is "very bad" in the sense that the bulk of the mass is far from that for the target density. Both densities are shown together in Figure 3.6.

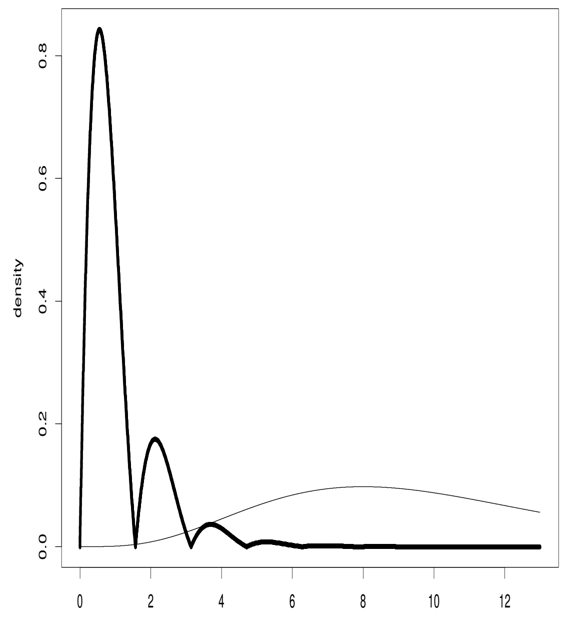
In Figure 3.7 we give histograms showing the resulting distribution of values given by 100,000 draws using a standard Metropolis-Hastings algorithm run forward for 300, 500, 1000, and 2000 time steps.

As expected, we are not achieving convergence to the target distribution. Indeed, there is not much difference at all between 1000 and 2000 time steps. However, we would not like to waste the information gained after 300 (or fewer) time steps – as we have at this point at least made some progress in moving towards the target distribution and we have certainly begun to capture it's multi-modal behavior. We propose to start over with the Metropolis-Hastings algorithm using the output of our previous failed attempt as a candidate distribution. This involves representing the output of our previous attempt in the form of a histogram to a desired accuracy (bin width), and drawing values from this distribution of values. Note that we are now taking the candidate density q to be a step function that is constant over each bin.

Consider the output after 300 time steps depicted in the upper left plot of Figure 3.7. Any distribution estimated with a finite sample will have a finite support. In

[1] Our notation for the parameters of a gamma distribution follows

$$X \sim \Gamma(\alpha, \beta) \Rightarrow X \text{ has density } f(x; \alpha, \beta) = \mathbb{1}_{(0,\infty)}(x) \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp\{-\beta x\}.$$



The multi-modal density is the normalized target density π. The unimodal density is the candidate density q.

Figure 3.6: The target and candidate densities.

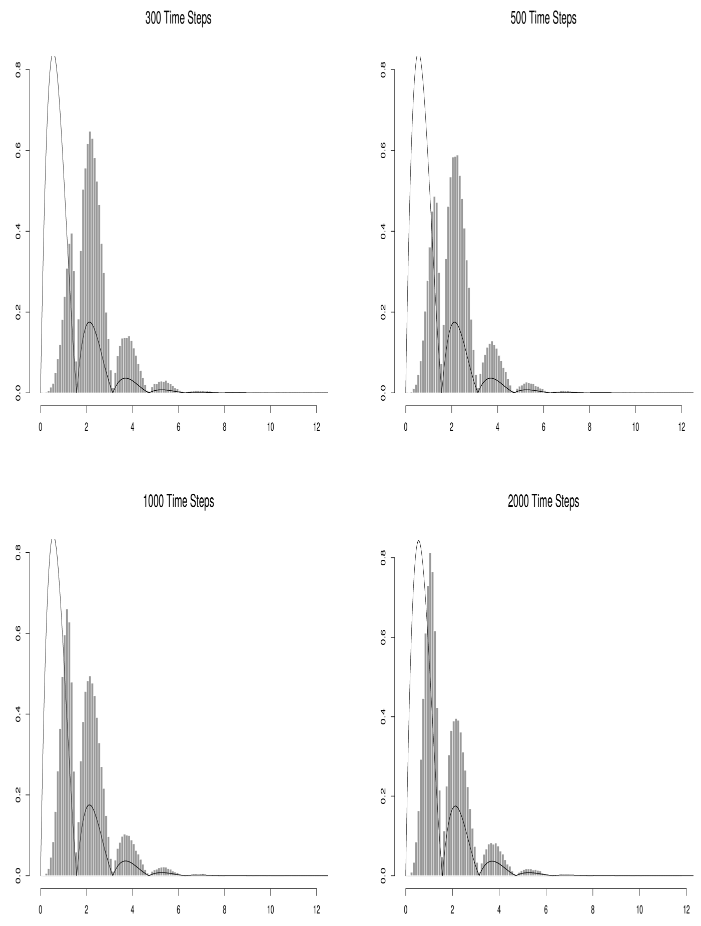


Figure 3.7: 100,000 draws from $\pi(x) \propto \mathbb{1}(0,\infty)(x)\ \exp\{-x\}\,|\sin(x)\cos(x)|$ using the Metropolis-Hastings algorithm with the $\Gamma(5, \frac{1}{2})$ candidate.

this particular case, if we were to make a "refinement" to the Metropolis-Hastings algorithm by drawing candidate values from this histogram distribution, we would never get proposals that are less than 0.4 or greater than approximately 6.0. For this reason, we must "spread out" the candidate histogram distribution. We illustrate the procedure with a simplified histogram.

Spreading the Candidate Distribution

Suppose that we are trying to ultimately draw values from a target distribution with density π with support $(0, \infty)$ and that we begin with an independent candidate distribution with density q. Let us further suppose that we have already run the Metropolis-Hastings for some $T > 0$ time steps and that this has resulted in a sample with the histogram depicted in Figure 3.8.

We have chosen, in this case, to store the output in a histogram with bin width 0.1. We store this width and the vector of histogram heights:

$$\begin{aligned} \text{binwidth} &= 0.1 \\ \text{histogram} &= [0.0,\ 0.0,\ 2.5,\ 3.5,\ 0.0,\ 3.0,\ 1.0]. \end{aligned}$$

We "spread" this histogram over the entire support set as follows.

(1) For gaps on the left end, we put a mass that is equal to that for the first non-empty bin encountered when moving from left to right. In this example, the histogram vector becomes

$$\text{histogram} = [2.5,\ 2.5,\ 2.5,\ 3.5,\ 0.0,\ 3.0,\ 1.0].$$

(2) For gaps in the middle, we put a mass that is equal to the average mass of the two surrounding bins. In this example, the histogram vector becomes

$$\text{histogram} = [2.5,\ 2.5,\ 2.5,\ 3.5,\ 3.25,\ 3.0,\ 1.0].$$

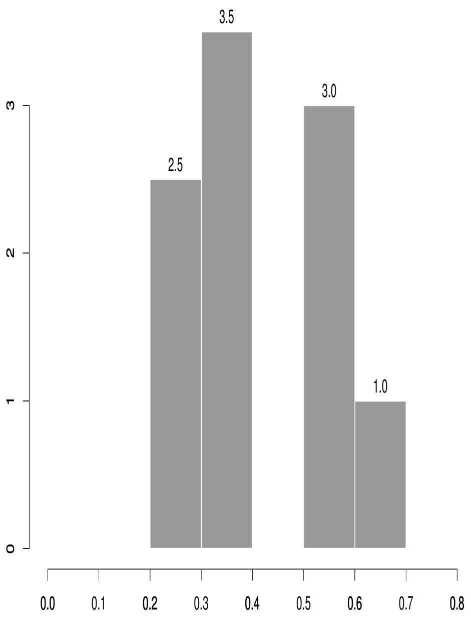


Figure 3.8: Output of the Metropolis-Hastings algorithm run for $T > 0$ time steps with candidate q.

(Note: If there had been more empty bins between the 3.5 and the 3.0, then they would have **all** been assigned the height of 3.25.)

(3) We create a "dummy bin" on the right end that, in this case, will represent the probability of drawing a value greater than 0.7. We do this by putting a mass equal to that for the last non-empty bin. Our histogram vector becomes

$$\text{histogram} = [2.5,\ 2.5,\ 2.5,\ 3.5,\ 3.25,\ 3.0,\ 1.0,\ 1.0].$$

(4) Finally, we renormalize the histogram. In this case, with a bin width of 0.1, the heights should add up to 10. Since they add up to 19.25, we multiply all values by 10/19.25 to get

$$\begin{aligned}\text{histogram} &= [\,1.2987,\ 1.2987,\ 1.2987,\ 1.8182,\\ &\quad 1.6883,\ 1.5584,\ 0.5195,\ 0.5195\,].\end{aligned}$$

Drawing from the Candidate Histogram Distribution

(1) We begin by turning the histogram heights into probabilities. We create the vector

$$\begin{aligned}\text{probabilities} &= \text{binwidth} \times \text{histogram}\\ &= [\,1.2987,\ 1.2987,\ 1.2987,\ 1.8182,\\ &\quad 1.6883,\ 1.5584,\ 0.5195,\ 0.5195\,].\end{aligned}$$

(2) We select a bin by drawing a random value U_1 from the Uniform$(0, 1)$ distribution. In this case if $U_1 < 0.12987$, we select the first bin. If $U_1 < 0.12987 + 0.12987$, we select the second bin, etc.

(3) (a) If we have selected any bin but the last, we then draw our candidate value uniformly from within this bin. For example, if we have selected the fourth

bin, we draw a value U_2 from the Uniform$(0,1)$ distribution and set

$$\text{candidate} \;=\; \text{binwidth} \times U_2 + (4-1) \times \text{binwidth}.$$

(b) If we have selected the last bin, we draw a value from a distribution with support (c, ∞) where c is the upper bound for the original histogram. In this example, $c = 0.7$.

We have chosen to use a simple shifted exponential distribution with a rate 1. (This rate was chosen so that this exponential piece of the candidate extends in a continuous way off of the last bar of the histogram.) In this case, we draw a value from an exponential distribution with rate 1 that is shifted 0.7 units to the right.

To summarize, in this example our candidate density is

$$q(x) = \begin{cases} 1.2987 & \text{for } 0 < x < 0.1 \\ 1.2987 & \text{for } 0.1 \le x < 0.2 \\ 1.2987 & \text{for } 0.2 \le x < 0.3 \\ 1.8182 & \text{for } 0.3 \le x < 0.4 \\ 1.6883 & \text{for } 0.4 \le x < 0.5 \\ 1.5584 & \text{for } 0.5 \le x < 0.6 \\ 0.5195 & \text{for } 0.6 \le x < 0.7 \\ 0.5195\, e^{-(x-0.7)} & \text{for } x \ge 0.7. \end{cases}$$

We now return to our example.

Example

We will use the adaptive IMH algorithm to draw values from

$$\pi(x) \propto \mathbb{1}_{(0,\infty)} \exp\{-x\} \cdot |\sin(x)\cos(x)|,$$

using the $\Gamma(5, \frac{1}{2})$ candidate distribution with density

$$q(x) \propto \mathbb{1}_{(0,\infty)} x^4 e^{-\frac{1}{2}x} \qquad x > 0$$

as our starting candidate.

Running a standard Metropolis-Hastings algorithm forward for 100 time steps produces the distribution of values shown in Figure 3.9. The target density is also shown. Using the histogram shown in Figure 3.9 as the new candidate density, we again run the standard Metropolis-Hastings algorithm forward for 100 time steps. The output is after this refinement shown in Figure 3.10. Finally, using the histogram shown in Figure 3.10 as the new candidate density, we run the standard Metropolis-Hastings algorithm forward for 100 time steps. The output is after this second refinement is shown in Figure 3.11. At this point, it appears, at least visually, that we have achieved convergence. Assessing convergence for the adaptive IMH algorithm is subject of future work as noted in Section 7.2.1.

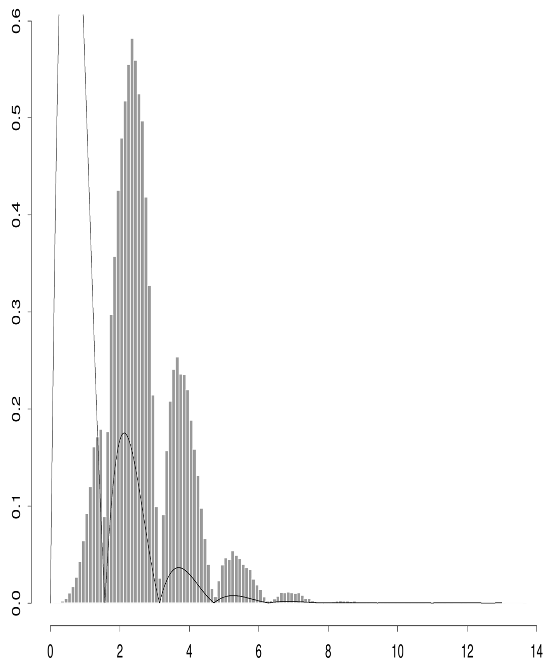


Figure 3.9: 100,000 draws from π after 100 time steps using $Q \sim \Gamma(5, \frac{1}{2})$.

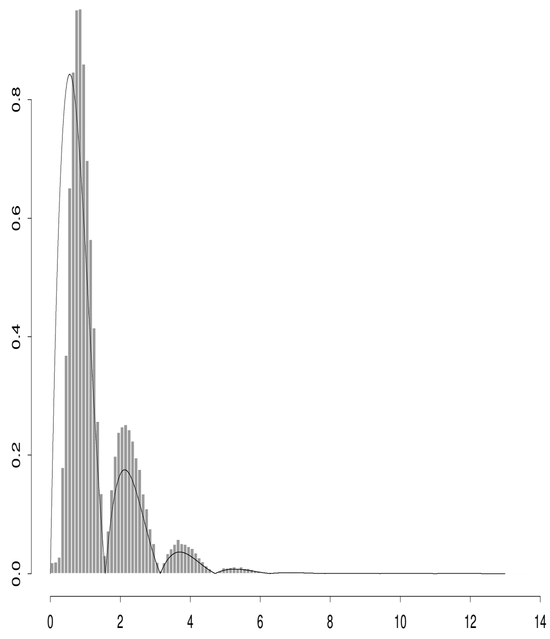


Figure 3.10: Refinement 1: 100,000 draws from π after 100 time steps using distribution depicted in Figure 3.9 as a candidate.

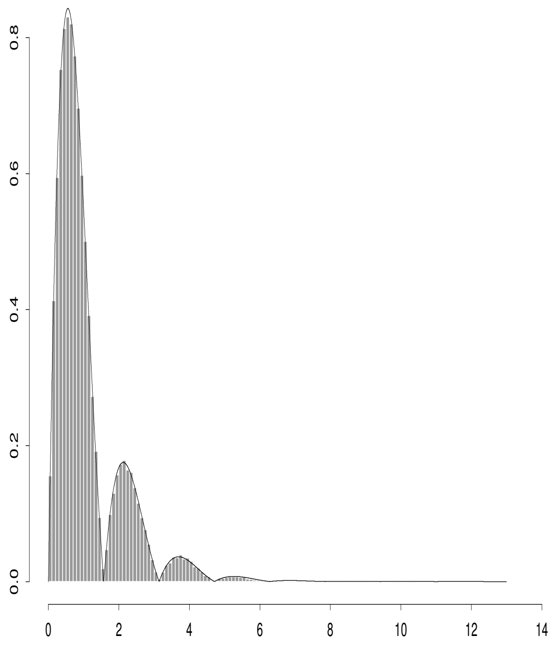


Figure 3.11: Refinement 2: 100,000 draws from π after 100 time steps using distribution depicted in Figure 3.10 as a candidate.

Chapter 4

Slice Coupling Methods

4.1 Introduction

As illustrated in Section 2.1, perfect sampling methods usually rely on the investigators' ability to **couple sample paths of a Markov chain**. This is often a non-trivial task, and, in the case of a continuous state space, it may depend on the development of tedious minorization conditions as in Corcoran and Tweedie [10] and Murdoch and Green [43]. In this chapter, as an alternative, we describe and develop variations on the *layered multishift coupler* due to Wilson [60] that allows one **to obtain potentially common draws from two different continuous distributions**.

These common draws are relatively straightforward to accomplish for uniform distributions through an *accept/reject step*, for example. The essential idea to achieve the coupling for non-uniform distribution is to make use of standard slice sampling ideas which are presented in Section 4.2.2. The final step in slice sampling, even for non-uniform distributions, requires one to make uniform draws and the basic idea is to use one of the coupling methods for uniform distributions to carry out this last step.

Wilson [60] has used this idea for his layered multishift coupler (the wording "layered multishift coupler" is often used synonymously for the uniform coupler as well as for its extensions to non-uniform distributions). We present additional uniform coupling ideas that allow one to obtain potentially common draws for more general classes of distributions, like the *folding coupler*, *shift-and-patch algorithm*, and the so-called *shift and fold coupler*.

This chapter is organized as follows. In Section 4.2, we present the the folding coupler which allows one to draw from two different uniform distributions where the support of one is contained within the support of the other using a **single** random number for both draws. The advantage of this method over an accept/reject step is the following. The final step in slice sampling where one has to make uniform draws on subsequently narrower and narrower slices is typically implemented with an accept/reject algorithm that uses an a priori unknown number of values from a random number generator. In a perfect simulation setting, it is essential that one organizes and reuses random numbers as mentioned in Section 2.1. It has been our experience that newcomers to perfect simulation have benefited from the streamlined storage and retrieval offered by the folding coupler. In Section 4.2.3 we illustrate how this can be useful within slice sampling to achieve common draws from distributions with different scale parameters with the example of two normal distributions with common means, but different variances.

Section 4.3 discusses Wilson's layered multishift coupler, which enables one to sample potentially common values from two distributions with the same shape but with different locations. This is demonstrated in Section 4.3.2 for two normal distributions with a common variance but different means. We then extend the procedure to distributions with non-invertible densities in Section 4.3.3 using the *shift-and-patch algorithm*.

In Section 4.4, we combine both folding and shifting methods to allow potentially common draws from distributions that differ both in shape and location such as two normal distributions with different means **and** different variances in Section 4.4.1 or two gamma distributions with different shape and scale parameters in Section 4.4.2.

Finally, we demonstrate in Section 4.5 how the above techniques can be used to couple sample paths within perfect sampling. We specify algorithms to simulate from an auto-exponential distribution in Section 4.5.1 (using a folding coupler), from an auto-gamma distribution in Section 4.5.2 (using an accept/reject step), and a storage model in Section 4.5.3 (using a shift and fold coupler).

4.2 The Folding Coupler

Our goal is to draw a potentially common value from two differently shaped distributions for the purpose of coupling sample paths of a Markov chain. The folding coupler requires that only a single random number be generated and is hence a simple alternative to a more standard accept/reject algorithm that requires an a priori unknown number of random numbers. This is especially useful in a perfect simulation setting where one stores and reuses random numbers.

4.2.1 The Uniform Distribution

We begin by defining and illustrating the folding coupler for simple uniform random variables. Suppose that we want to generate a random variable X that is uniformly distributed on the interval $[a, b]$ and another random variable Y that is uniformly distributed on the interval (c, d) where $a \leq c < d \leq b$. In Section 4.3 we remove the requirement that (c, d) be contained in (a, b) by combining the folding coupler with Wilson's shift coupler [60].

The folding coupler, illustrated in Figure 4.1, proceeds as follows.

(1) Draw a value $X \sim \text{Uniform}(a, b)$.

(2) If $X \in (c, d)$, accept this also as a uniform draw from (c, d) and set $Y = X$.

(3) If $X \notin (c, d)$, map X onto the interval (c, d) by "folding it in" according to the function

$$f(x) = f(x; a, b, c, d) = \begin{cases} \frac{x-a}{(c-a)+(b-d)}(d-c)+c, & \text{if } x \in (a, c) \\ d - \frac{b-x}{(c-a)+(b-d)}(d-c), & \text{if } x \in (d, b) \end{cases} \tag{4.1}$$

and set $Y = f(X)$.

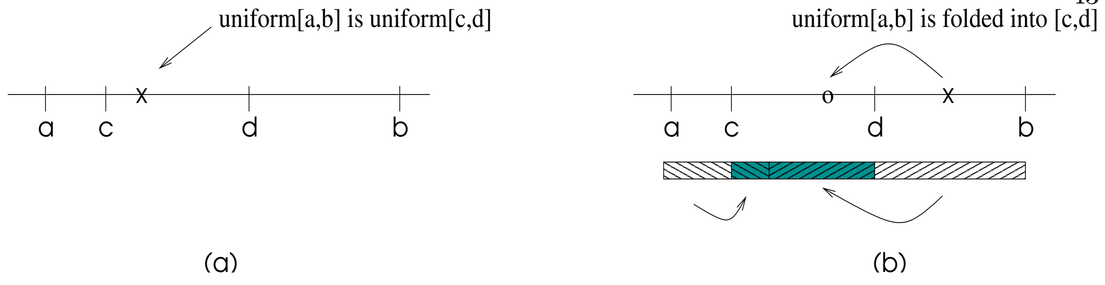

uniform[a,b] is uniform[c,d]
x
a
c
d
b
(a)
uniform[a,b] is folded into [c,d]
o
x
a
c
d
b
(b)

Figure 4.1: (a) Step 2 of the folding coupler, (b) Step 3 of the folding coupler

It is routine to show that the resulting Y value will be uniformly distributed on the interval (c, d) and easy to see that $Y = X$ whenever $X \in (c, d)$.

We also have two particular cases that will be of interest when we use this technique to generate random transitions for Markov chains.

(1) If $c = a$ and $d \leq b$ (the interval for Y is contained in and left aligned with the interval for X), then this algorithm will result in draws x, y, where $y \leq x$.

(2) If $m := (c + d)/2 = (a + b)/2$ (the interval for Y is contained in and centered in the interval for X), then this algorithm will result in a draws x and y, where $|y - m| \leq |x - m|$.

4.2.2 Other Distributions - Using Slice Coupling

We now extend the procedure of Section 4.2.1 to enable us to draw a common value from other non-uniform distributions.

One way to sample from a distribution is to sample uniformly from the region under the plot of its density function. To formalize a method for achieving this, we describe the basis for the *slice sampler* approach.

Suppose we can draw a value x for a random variable X with density function $\pi(x) = c \cdot h(x)$ where the constant of proportionality c is possibly unknown. We then draw a value $Y = y$ given $X = x$ uniformly over the interval $(0, h(x))$ and finally draw a value of X' uniformly from $H(y)$ where $H(y)$ is defined to be the "horizontal slice"

$$H(y) = \{x | h(x) > y\}.$$

Y has density

$$\begin{aligned} f_Y(y) &= \int_{-\infty}^{\infty} f_{Y|X}(y|x)\pi(x)dx \\ &= \int_{-\infty}^{\infty} \frac{1}{h(x)} \mathbb{1}_{(0,h(x))}(y)\pi(x)dx \\ &= c\int_{H(y)} dx \\ &= c\,|H(y)| \end{aligned}$$

and so X' has density

$$\begin{aligned} f_{X'}(x) &= \int_{-\infty}^{\infty} f_{X'|Y}(x|y)f_Y(y)dy \\ &= \int_{-\infty}^{\infty} \frac{1}{|H(y)|} \mathbb{1}_{H(y)}(x)c\ |H(y)|dy \\ &= c\int_0^{h(x)} dy \\ &= ch(x) \\ &= \pi(x). \end{aligned}$$

The final step of the slice sampler requires that one draws uniformly from a horizontal slice. Using the folding coupler (or another uniform coupler) at this step allows us to potentially draw a common value for two differently shaped non-uniform distributions.

We illustrate this approach with the normal distribution.

4.2.3 Common draws from $N(\mu, \sigma_1^2)$ and $N(\mu, \sigma_2^2)$

Assume that our goal is to draw from two normal distributions $N(\mu, \sigma_1^2)$ and $N(\mu, \sigma_2^2)$ where $\sigma_1 > \sigma_2$.

Let $h_1(x)$ and $h_2(x)$ be the likelihood (or density) functions for the $N(\mu, \sigma_1^2)$ and $N(\mu, \sigma_2^2)$ distributions, respectively.

(1) Draw $X_1 \sim N(\mu, \sigma_1^2)$ and, in tandem (using the same random numbers), generate $X_2 \sim N(\mu, \sigma_2^2)$ distribution. In this example, we can simply re-scale and let $X_2 = \frac{\sigma_2}{\sigma_1} X_1$. See Figure 4.2 (A).

(2) Draw $U \sim \text{Uniform}(0, 1)$ and let $Y_1 = U \cdot h_1(X_1)$ and $Y_2 = U \cdot h_2(X_2)$.

(3) Compute the left and right endpoints for each horizontal slice.

$$\begin{aligned} L_1 &= -\sigma_1\sqrt{-2\log(\sigma_1 Y_1)} + \mu & L_2 &= -\sigma_2\sqrt{-2\log(\sigma_2 Y_2)} + \mu \\ R_1 &= \sigma_1\sqrt{-2\log(\sigma_1 Y_1)} + \mu & R_2 &= \sigma_2\sqrt{-2\log(\sigma_2 Y_2)} + \mu. \end{aligned}$$

Note that $L_1 < L_2 < R_2 < R_1$ and that the intervals (L_1, R_1) and (L_2, R_2) are both centered at μ. See Figure 4.2 (B).

(4) Draw a value $X \sim \text{Uniform}(L_1, R_1)$. This uniform distribution is conditional on the previous steps of this algorithm. Unconditionally, X has the $N(0, \sigma_1^2)$ distribution. Furthermore,

if $X \in (L_2, R_2)$, then X is uniformly distributed over (L_2, R_2) and hence the unconditional distribution of X is $N(0, \sigma_2^2)$ and so we have achieved a common draw from both normal distributions. See Figure 4.2 (C).

Otherwise, if $X \notin (L_2, R_2)$, we will not get a common draw for both normal distributions, and we finish by drawing from the $N(\mu, \sigma_2^2)$ distribution by folding $X \sim \text{Uniform}(L_1, R_1)$ onto the interval (L_2, R_2) according to (4.1).

We note that when $X \notin (L_2, R_2)$ at the end of step 4 above, we could instead finish drawing from $N(\mu, \sigma_2^2)$ by independently drawing a uniform on (L_2, R_2) but our intention is to use this construction in a specific way to drive transitions of a Markov chain while preserving Markov and monotonicity properties.

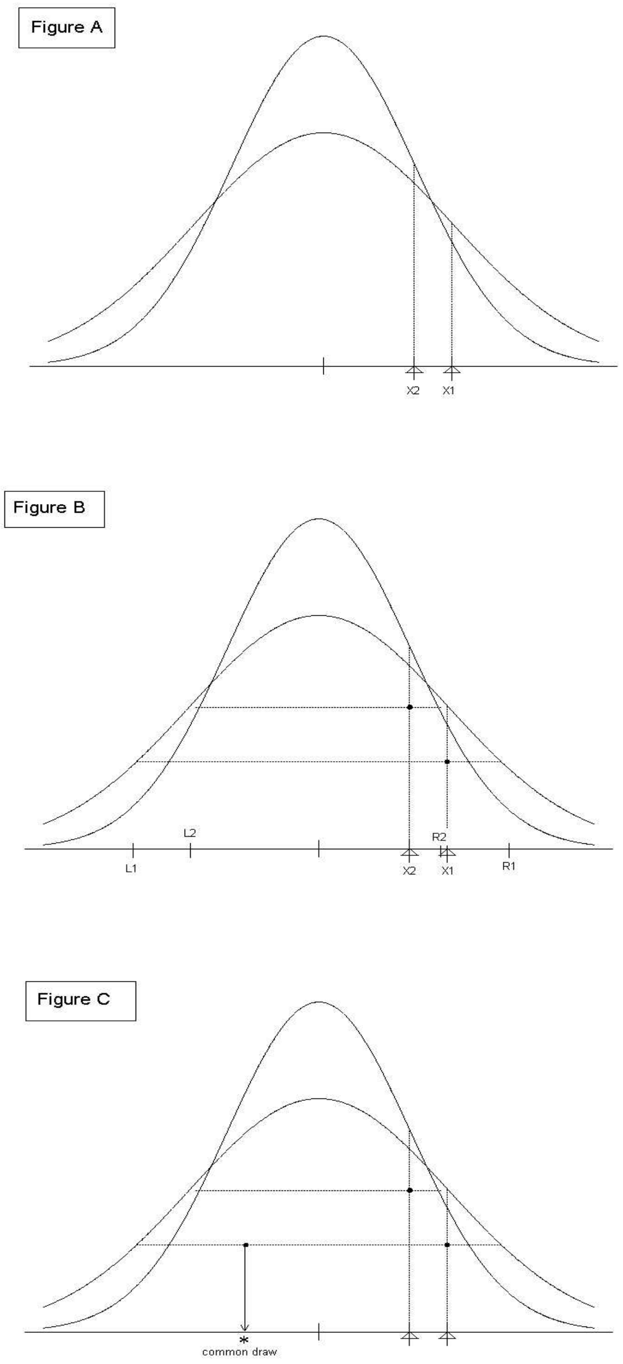


Figure 4.2: A common draw from $N(\mu, \sigma_1^2)$ and $N(\mu, \sigma_2^2)$ using the folding coupler

4.3 The Layered Multishift Coupler

The folding coupler of Section 4.2 allows us to draw common values from two differently shaped distributions. In this section, we describe the layered multishift coupler of Wilson [60] which allows one to draw common values from distributions with different location parameters. We extend these methods to non-invertible distributions in Section 4.3.3. In Section 4.4 we will combine folding and shifting methods to enable common draws from differently shaped distributions and different (but overlapping) supports.

4.3.1 The Uniform Distribution

We begin by describing the layered multishift coupler of Wilson [60] for the uniform distribution.

Let X be uniformly distributed over the interval (L, R) and consider the mapping

$$g(s) = g_X(s; L, R) = \left\lfloor \frac{s + R - X}{R - L} \right\rfloor (R - L) + X \tag{4.2}$$

where $\lfloor \cdot \rfloor$ is the greatest integer or floor function.

It is easy to see that for any fixed s, $g(s)$ is uniformly distributed over the interval $(s + L, s + R)$ since

$$\begin{aligned} g(s) &= \left\lfloor \frac{s + R - X}{R - L} \right\rfloor (R - L) + X \\ &= \left[\left(\frac{s + R - X}{R - L} \right) - \text{frac} \left(\frac{s + R - X}{R - L} \right) \right] (R - L) + X \\ &= s + R - \text{frac} \left(\frac{s + R - X}{R - L} \right) (R - L). \end{aligned}$$

Now since $\left(\frac{s+R-X}{R-L}\right)$ is uniformly distributed over the interval $(\frac{s}{R-L}, \frac{s}{R-L} + 1)$, the fractional part of this expression is uniformly distributed over the interval $(0, 1)$. Therefore, from $s + R$, we subtract a quantity that is uniform over $(0, R - L)$, and we have that $g(s)$ is uniform over $(s + L, s + R)$.

As a result of the truncation provided by the greatest integer function, it is possible for different values of s to map to the same $g(s)$. Consequently, we can use this function to draw a common value from both a particular uniform distribution and a shifted version. Additionally, this transformation is monotone in the sense that for $s_1 \leq s_2$, it will result in random variables $g(s_1) \leq g(s_2)$.

4.3.2 Common Draws from $N(\mu_1, \sigma^2)$ and $N(\mu_2, \sigma^2)$

In this section we describe how Wilson [60] extends the procedure of Section 4.3.1 to enable one to draw a common value from other non-uniform distributions with different location parameters using the same slice sampler approach as in Section 4.2.2. We again illustrate with the normal distribution.

Assume that the goal is to draw from two normal distributions $N(\mu_1, \sigma^2)$ and $N(\mu_2, \sigma^2)$ with $\mu_1 < \mu_2$.

Let $h(x)$ denote the $N(0, \sigma^2)$ likelihood function,

(1) Draw a value X from the $N(0, \sigma^2)$ distribution.

(2) Draw a value U from the Uniform$(0, 1)$ distribution. Let $Y = U \cdot h(X)$.

(3) Compute the endpoints of the horizontal slice at Y.

$$\begin{aligned} L &= -\sigma\sqrt{-2\log Y} \\ R &= \sigma\sqrt{-2\log Y} \end{aligned}$$

(4) Draw a value $X = x$ from the Uniform(L, R) distribution.

(5) Shift this draw by computing $x_1 = g(\mu_1)$ and $x_2 = g(\mu_2)$.

x_1 and x_2 are values drawn from the $N(\mu_1, \sigma^2)$ and $N(\mu_2, \sigma^2)$ distributions respectively and are potentially the same. Again, since $g(\cdot)$ is monotone, this algorithm will result in draws x_1 and x_2, with $x_1 \leq x_2$, of the stochastically ordered random variables $X_1 \sim N(\mu_1, \sigma^2)$ and $X_2 \sim N(\mu_2, \sigma^2)$.

4.3.3 Non-invertible distributions

The final steps $(3)-(5)$ in the multishift coupler require a draw from a horizontal slice which can be problematic if the distribution is not invertible. Of course one can use an accept/reject scheme to draw uniformly from an interval with unknown endpoints, but the true endpoints are still needed for the shift coupler in (4.2). That is, if X is distributed uniformly on the interval (L, R) but we only have approximate endpoints L' and R' where $L' \leq L < R \leq R'$ at our disposal, then

$$g'(s) = \left\lfloor \frac{s + R' - X}{R' - L'} \right\rfloor (R' - L') + X \not\sim \text{Uniform}(s + L, s + R). \tag{4.3}$$

In the following, we assume that L and R are unknown but that one can tell whether a given point is or is not in the interval (L, R). This is the case, for example, when slice sampling from a unimodal density as in Section 4.4.2. An accept/reject scheme can then be used to sample uniformly from (L, R).

The Shift-and-Patch Algorithm

We begin by considering using (4.2) to send draws from a uniform distribution on (L, R) to $(L+s, R+s)$. For a specific example, we take $L = 1$, $R = 3$, and $s = 1$. As shown by the shading in Figure 4.3, the first half of the interval maps to the second half and the second half of the interval maps to the first. Points within each half maintain their ordering.

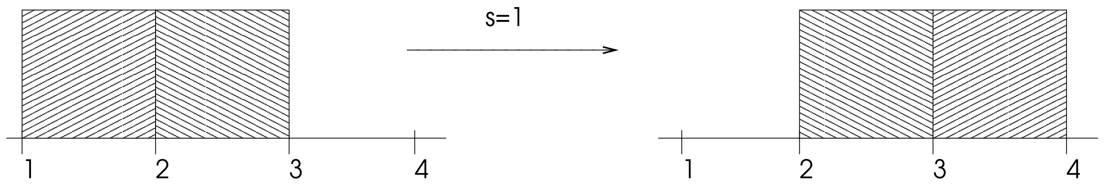


Figure 4.3: $g_X(1; 1, 3)$ for $X \sim \text{Uniform}(1, 3)$

Figure 4.4 shows the result using (4.3) with the approximate endpoints $L' = 0.9$ and $R' = 3.2$. Here, $e_L = L - L'$ and $e_R = R' - R$ denote, respectively, the (unknown) left and right approximation errors and $e_T = e_L + e_R$ denotes the total endpoint approximation error. Points falling outside the interval $(2, 4)$ can be detected (for example, see step 4 in the algorithm described in Section 2.3.2) and hence rejected, but in order to result in a draw uniformly distributed on $(2, 4)$, we must "fill in" or "patch" the gap with unknown endpoints in the center.

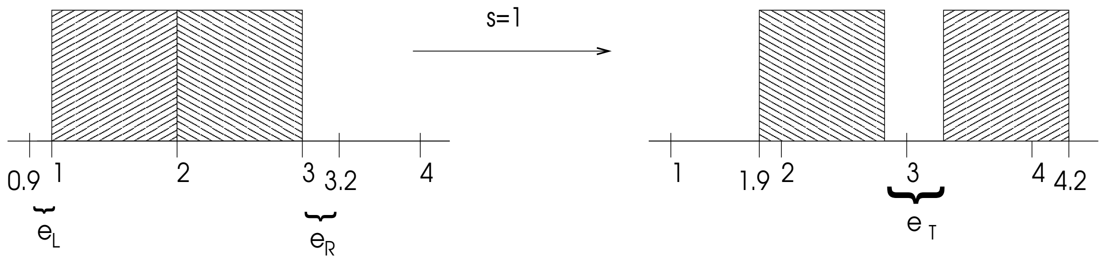


Figure 4.4: $g_X(1; 0.9, 3.2)$ for $X \sim$ Uniform$(1, 3)$

We achieve this patch by reusing the rejected draws falling in $(1.9, 2)$ and $(4, 4.2)$. These draws are manually shifted back by simple subtraction (Figure 4.5(a)) and the result is then re-shifted with the map in (4.3) (Figure 4.5(b)).

More formally, the algorithm can be specified as follows.

(1) **Shift**: let $X' = g_X(s) = \left\lfloor \frac{s+R'-X}{R'-L'} \right\rfloor (R' - L') + X$

(2) if $X' \notin (L + s, R + s)$, then **patch**:

- shift back manually by subtracting s and
- use the truncation map again.

In general, this shift-and-patch algorithm applies whenever $e_T < |s| < R - L$.

Proposition 1 Let $e_L = L - L'$ and $e_R = R' - R$, $e_T = e_L + e_R$ and $l_{int} = R - L$. Assume that

$$e_T < |s| < l_{int}.$$

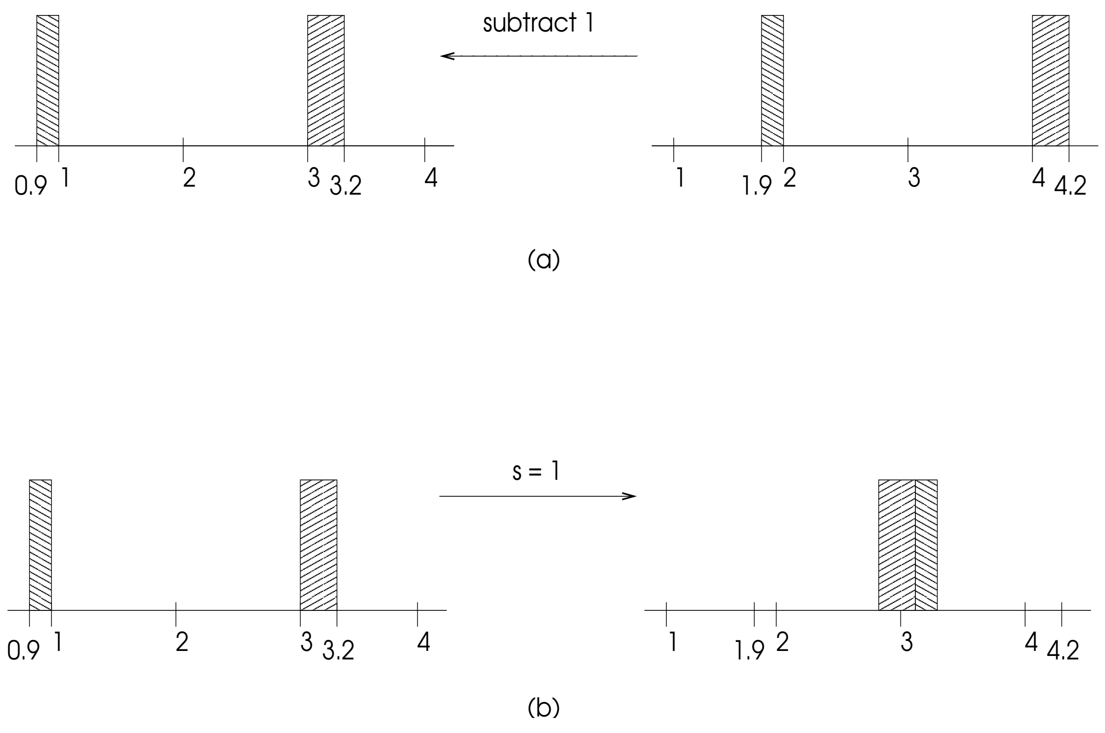


Figure 4.5: Patching the Gap

Then the Shift-And-Patch algorithm defined above shifts $X \sim \text{Uniform}(L, R)$ to $Z \sim \text{Uniform}(L + s, R + s)$.

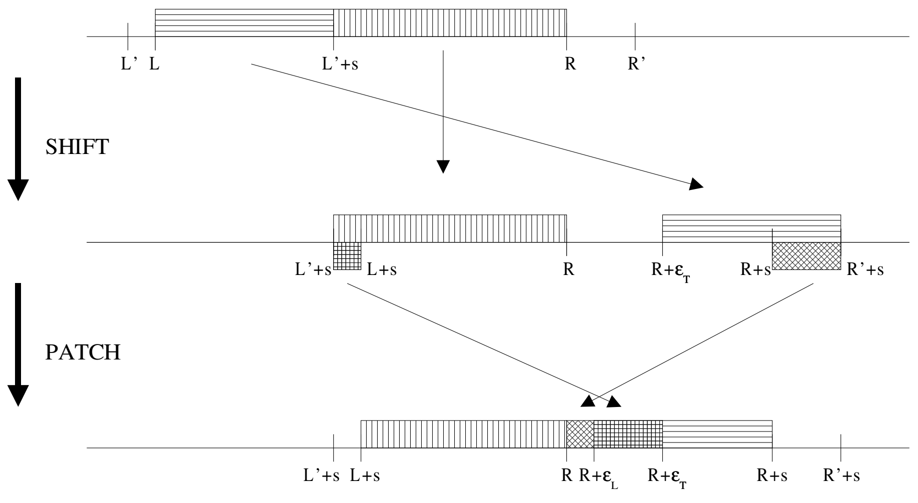


Figure 4.6: Shifting and patching the gap.

Proof

We will prove Proposition 1 by carefully examining where the shift- and patch-step map numbers. Consider first the **shift-step**. We look at the interval $(\frac{R'+s-R}{R'-L'}, \frac{R'+s-L}{R'-L'})$, which is the domain for the floor-function used in the truncation map. Assume that $s > 0$ (the case when $s < 0$ works similarly). We observe that $1 \in (\frac{R'+s-R}{R'-L'}, \frac{R'+s-L}{R'-L'})$ since $\frac{R'+s-X}{R'-L'} = 1$ for $X = L' + s$ and that , by assumption, we have $L = L' + e_L < L' + s < L' + l_{int} = R$ so that $X = L' + s \in (L, R)$.

Since the interval has length $\frac{R-L}{R'-L'} = \frac{l_{int}}{l_{int}+e_T} < 1$, and $1 \in (\frac{R'+s-R}{R'-L'}, \frac{R'+s-L}{R'-L'})$, the floor-function in the truncation map can only take on values in $\{0, 1\}$. In fact, we have

$$\left\lfloor \frac{R'+s-X}{R'-L'} \right\rfloor = \begin{cases} 0 & X \in (L'+s, R) \\ 1 & X \in (L, L'+s). \end{cases}$$

With this information, the truncation map in the shift-step can be written as

$$g_X(s) = \begin{cases} X & X \in (L' + s, R) \\ X + (R' - L') & X \in (L, L' + s). \end{cases}$$

It is obvious that $g_X(s)$ maps $(L'+s, R)$ uniformly onto $(L'+s, R)$. The second interval will simply be shifted by the constant amount of $R' - L'$. To determine where the second interval will be mapped, we evaluate $g_X(s)$ at the corresponding endpoints L and $L' + s$. We find that $g_L(s) = L + (R' - L') = R + (R' - R) + (L - L') = R + e_T$ and $\lim_{X \to (L'+s)^-} g_X(s) = L' + s + (R' - L') = R' + s$, so that the shift-step can be summarized as uniformly mapping the following intervals onto

$$\begin{array}{lcl} (L' + s, R) & \longrightarrow & (L' + s, R) \\ (L, L' + s) & \longrightarrow & (R + e_T, R' + s)\,. \end{array}$$

Note that this leaves us with the unwanted intervals $(L' + s, L + s)$ and $(R + s, R' + s)$, as well as a gap of size e_T in the middle of the interval (see Figure 4.6 for illustration).

We will now show that the **patch-step** places these unwanted intervals outside $(L + s, R + s)$ right into this gap.

In the patch step, we first shift back "manually", so that the domain for the truncation map becomes $(L', L) \cup (R, R')$. Since

$$\left\lfloor \frac{R' + s - X}{R' - L'} \right\rfloor = \begin{cases} 0 & X \in (R, R') \\ 1 & X \in (L', L), \end{cases}$$

the truncation map for these intervals can be written as

$$g_X(s) = \begin{cases} X & X \in (R, R') \\ X + (R' - L') & X \in (L', L), \end{cases}$$

which shows that the patch-step uniformly maps the following intervals onto

$$\begin{array}{lcl} (R + s, R' + s) & \longrightarrow & (R, R + e_L) \\ (L' + s, L + s) & \longrightarrow & (R + e_L, R + e_T)\,, \end{array}$$

which "fills the gap" $\square$.

Remark

Note the assumption $e_T < |s|$ can be satisfied by choosing small enough increments when searching for the estimated endpoints L' and R'. The assumption that $|s| < l_{int}$ is natural in the sense that with $|s| > l_{int}$ it is impossible to get common draws anyway.

4.4 The Shift and Fold Coupler

In this section, we combine the folding coupler with the layered multishift coupler of Wilson [60] to increase the rate of common draws for two distributions with different shapes. Again, we use normal distributions (this time with both different means and different variances) for illustration purposes. (We apply the shift and fold coupler within a perfect simulation setting in Section 4.5.3.)

4.4.1 Common Draws from $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$

Suppose that we want a (potentially) common value from draw from $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$. Assume that $\sigma_1 > \sigma_2$. We proceed as follows.

(1) Start with a value X_1 drawn from the $N(\mu_1, \sigma_1^2)$ distribution.

(2) Also, draw a value X_2 in tandem (using the same random numbers) from the $N(\mu_1, \sigma_1^2)$ distribution. Or, more simply, let $x_2 = \sigma_2 \left(\frac{x_2 - \mu_1}{\sigma_1}\right) + \mu_2$.

(3) Use a common draw from the Uniform$(0, 1)$ to draw heights for the horizontal slices and find the left and right endpoints L_1, R_1 and L_2, R_2. That is, draw a value U from the Uniform$(0, 1)$ distribution, let $Y_1 = Uh_1(X_1)$ and $Y_2 =$

$Uh_2(X_2)$, and let

$$\begin{array}{rclrcl} L_1 &=& -\sigma_1\sqrt{-2\log(\sigma_1 Y_1)}+\mu_1 & L_2 &=& -\sigma_2\sqrt{-2\log(\sigma_2 Y_2)}+\mu_2 \\ R_1 &=& \sigma_1\sqrt{-2\log(\sigma_1 Y_1)}+\mu_1 & R_2 &=& \sigma_2\sqrt{-2\log(\sigma_2 Y_2)}+\mu_2 \end{array}$$

(4) Follow the procedure of Section 4.3.1 (Wilson's layered multishift coupler) to draw values uniformly from (L_1, R_1) and $(L_1 + (\mu_2 - \mu_1), R_1 + (\mu_2 - \mu_1))$.

(5) Follow the procedure of Section 4.2.1 (folding coupler) the uniform draw from $(L_1 + (\mu_2 - \mu_1), R_1 + (\mu_2 - \mu_1))$ as/into a uniform draw on (L_2, R_2).

4.4.2 Coupled Gamma Random Variates

We give an example that combines shifting, patching, and folding in order to produce (potentially common) draws from two gamma distributions.

We used a "peak-to-peak" or "mode-to-mode" shift to produce 100,000 draws from the $\Gamma(3, 2)$ and the $\Gamma(2, 3)$ distributions. Resulting histograms are shown in Figure 4.7.

4.5 Coupling Sample Paths in Perfect Sampling

As mentioned before, the purpose of the coupling methods presented in this section is to couple continuous sample paths of Markov chains in perfect sampling algorithms. We now describe three examples using techniques from previous sections within (perfect) MCMC methods.

4.5.1 The Auto-Exponential Distribution

We apply the folding coupler to updates within the Gibbs sampler in order to draw from the so-called *auto-exponential* density

$$\pi(x_1, x_2) \propto \exp\{-\beta_1 x_1 - \beta_2 x_2 - \beta_{12} x_1 x_2\} \tag{4.4}$$

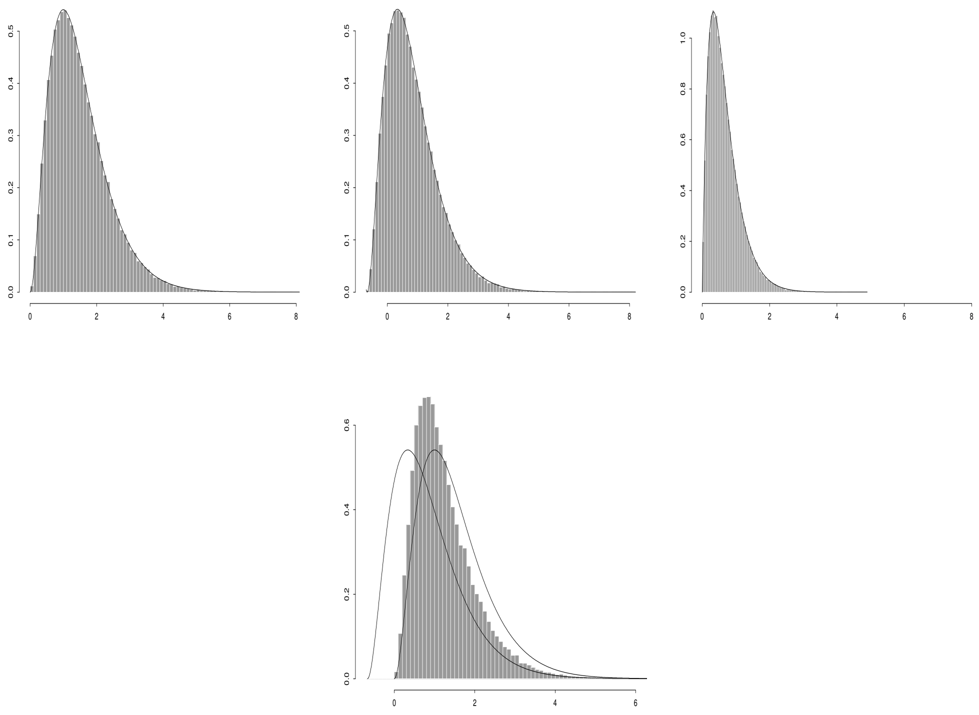


Figure 4.7: Top Row: draws from $\Gamma(3, 2)$, draws from shifted $\Gamma(3, 2)$, draws from (shifted and scaled) $\Gamma(2, 3)$. Bottom Row: common draws for $\Gamma(3, 2)$ and shifted version (60%).

where $\beta_1 > 0$, $\beta_2 > 0$, $\beta_{12} < 0$, $0 < x_1 < -\frac{\beta_2}{\beta_{12}}$, and $0 < x_2 < -\frac{\beta_1}{\beta_{12}}$.

We restrict β_{12} to be negative (which then forces restrictions on the ranges of x_1 and x_2) only for ease of algorithm exposition. This restriction is unnecessary and will be removed in Section 4.5.2.

To run the standard Gibbs sampler (see Appendix A), we wish to alternate draws from the conditional truncated exponential [1] distributions

$$\begin{aligned} X_1|X_2 = x_2 &\sim \text{truncated } \exp(\mathit{rate} = \beta_1 + \beta_{12}x_2) \\ X_2|X_1 = x_1 &\sim \text{truncated } \exp(\mathit{rate} = \beta_2 + \beta_{12}x_1), \end{aligned}$$

where the truncated exponential densities are simply exponential densities truncated to and renormalized over the appropriate support sets. We denote these densities (likelihoods) by $\pi_{x_2}(x)$ ($h_{x_2}(x)$) and $\pi_{x_1}(x)$ ($h_{x_1}(x)$), respectively. To be precise,

$$\pi_{x_2}(x) = \frac{(\beta_1 + \beta_{12}x_2)\exp[-(\beta_1 + \beta_{12}x_2)x]}{[1 - \exp(-(\beta_1 + \beta_{12}x_2)(-\beta_2/\beta_{12}))]} \qquad \text{for } 0 < x < -\frac{\beta_2}{\beta_{12}}$$

and

$$\pi_{x_1}(x) = \frac{(\beta_2 + \beta_{12}x_1)\exp[-(\beta_2 + \beta_{12}x_2)x]}{[1 - \exp(-(\beta_2 + \beta_{12}x_1)(-\beta_1/\beta_{12}))]} \qquad \text{for } 0 < x < -\frac{\beta_1}{\beta_{12}}.$$

Since β_{12} is negative, this model is *attractive* in the sense that a small (large) x_1 will produce a small (large) x_2 and vice versa. The Gibbs sampler is then stochastically monotone if we consider the partial ordering

$$(x_1, y_1) \preceq (x_2, y_2) \qquad \Longleftrightarrow \qquad x_1 \leq x_2 \text{ and } y_1 \leq y_2 \tag{4.5}$$

with a "lowest point" of $(0, 0)$ and a "highest point" at $\left(-\frac{\beta_2}{\beta_{12}}, -\frac{\beta_1}{\beta_{12}}\right)$.

That is, a sample path started at any point (x_1, x_2) in the state space will always stay in between sample paths started from $(0, 0)$ and $\left(-\frac{\beta_2}{\beta_{12}}, -\frac{\beta_1}{\beta_{12}}\right)$.

[1] Our notation for the parameter of an exponential distribution follows

$$X \sim \exp(\lambda) \Rightarrow X \text{ has density } f(x;\lambda) = 1\!\!1_{(0,\infty)}(x)\exp\{-\lambda x\}.$$

Updating sample paths

Suppose that at time n, the "lower process", started in state $(l_1^{(0)}, l_2^{(0)}) = (0,0)$ is at some point $(l_1^{(n)}, l_2^{(n)})$, and that the "upper process", started in state $(u_1^{(0)}, u_2^{(0)}) = \left(-\frac{\beta_2}{\beta_{12}}, -\frac{\beta_1}{\beta_{12}}\right)$, is at some point $(u_1^{(n)}, u_2^{(n)})$.

We want to draw values for

$$\begin{array}{ll} l_1^{(n+1)} & \text{from truncated } \exp(\beta_1 + \beta_{12} l_2^{(n)}) \text{ and} \\ u_1^{(n+1)} & \text{from truncated } \exp(\beta_1 + \beta_{12} u_2^{(n)}), \end{array}$$

and we want these to potentially be the same value in order for the sample paths to coalesce. Suppressing time notation, this means that we want to draw potentially common values for

$$\begin{array}{ll} l_1 & \text{from the truncated exponential density } \pi_{l_2}(x) \propto h_{l_2}(x) \text{ and} \\ u_1 & \text{from the truncated exponential density } \pi_{u_2}(x) \propto h_{u_2}(x) \end{array}$$

where both densities have common support $\left(0, -\frac{\beta_2}{\beta_{12}}\right)$.

To achieve these common draws, we

(1) Draw, in tandem (using the same random numbers), coupled values l_1' and u_1' directly from $\pi_{l_2}(x)$ and $\pi_{u_2}(x)$ so that $l_1' \leq u_1'$. One could, for example, evaluate the inverses of the distribution functions at a single value U_1 drawn from the Uniform$(0,1)$ distribution. These values give the locations of the initial vertical slices for each density.

(2) Draw $V_1 \sim$ Uniform$(0,1)$ and set $Y_l = V_1 \cdot h_{l_2}(l_1')$ and $Y_u = V_1 \cdot h_{u_2}(u_1')$. These values give the heights of the horizontal slices for each density.

(3) Find the right endpoints of the horizontal slices, $R_l = R_l(Y_l)$ and $R_u = R_u(Y_u)$, where $R_l = h_{l_2}^{-1}(Y_l)$ and $R_u = h_{u_2}^{-1}(Y_u)$. Note that the left endpoints are always at 0 in this example and that $R_l \leq R_u$.

(4) Draw uniformly from the horizontal slices as follows.

(i) Draw $W_1 \sim \text{Uniform}(0,1)$ and set $W_1' = -\frac{\beta_2}{\beta_{12}} W_1$. This value is uniformly distributed over $\left(0, -\frac{\beta_2}{\beta_{12}}\right)$, the *widest* possible horizontal slice.

(ii) If $W_1' \leq R_u$, then W_1' is uniform over $(0, R_u)$, so we set $u_1 = W_1'$. On the other hand, if $W_1' > R_u$, we obtain the outcome u_1 by folding W_1' into the interval $(0, R_u)$ according to

$$u_1 = \frac{W_1' - R_u}{R - R_u} R_u$$

where $R = -\frac{\beta_2}{\beta_{12}}$.

(iii) Similarly, if $W_1' \leq R_l$, then W_1' is uniform over $(0, R_l)$, so we set $l_1 = W_1'$. On the other hand, if $W_1' > R_l$, we obtain the outcome l_1 by folding W_1' into the interval $(0, R_l)$ according to

$$l_1 = \frac{W_1' - R_l}{R - R_l} R_l$$

where $R = -\frac{\beta_2}{\beta_{12}}$.

Note that coupling of the lower and upper sample paths is achieved when $W_1' \leq R_l$. That is, coupling is achieved when the lower process accepts a horizontal uniform draw without folding. In this case, $l_1 = u_1 = W_1'$, so it is not necessary to run the upper process at all.

We have described only the updates for one of the first components of the lower and upper processes. The second components

$$\begin{aligned} l_2 = l_2^{(n+1)} &\quad \text{from truncated } \exp(rate = \beta_2 + \beta_{12} l_1^{(n+1)}), \quad \text{and} \\ u_2 = u_2^{(n+1)} &\quad \text{from truncated } \exp(rate = \beta_2 + \beta_{12} u_1^{(n+1)}), \end{aligned}$$

are updated in a similar manner with independent uniform variates U_2, V_2, and W_2.

We can now describe the algorithm for generating a perfect sample from π.

Folding Backward Coupling Algorithm

(1) Draw three independent sequences of Uniform$(0,1)$ variables $U_1^{(n)}$, $V_1^{(n)}$, and $W_1^{(n)}$ for the first component of the Gibbs chain, and three independent sequences of Uniform$(0,1)$ variables $U_2^{(n)}$, $V_2^{(n)}$, and $W_2^{(n)}$ for the second component of the Gibbs chain, for $n = 0, -1, -2, \ldots$.

(2) For each time $-n = -1, -2, \ldots$, start a lower path $(l_1^{(-n)}, l_2^{(-n)})$ at $(0,0)$.

(3) (a) For the first component:

For $k = n, n-1, \ldots, 1$, update from $l_1^{(-k)}$ to $l_1^{(-k+1)}$ by

(i) using $U_1^{(-k+1)}$ to draw a value l_1' directly from $\pi_{l_2^{(-k)}}(x)$,

(ii) setting $Y_1 \equiv Y_1^{(-k+1)} = V_1^{(-k+1)} h_{l_2^{(-k)}}(l_1')$

(the height of the vertical slice),

(iii) setting $R_1 \equiv R_1^{(-k+1)} = h^{-1}_{l_2^{(-k)}}(Y_l)$

(the right endpoint of the horizontal slice).

(iv) setting $W_1' \equiv W_1'^{(-k+1)} = -\frac{\beta_2}{\beta_{12}} W_1^{(-k+1)}$ and

$$l_1^{(-k+1)} = \begin{cases} W_1' & W_1' \leq R_1 \\ \frac{W_1' - R_1}{R - R_1} R_1 & W_1' > R_1, \end{cases}$$

where $R = -\frac{\beta_2}{\beta_{12}}$.

(b) For the second component:

For $k = n, n-1, \ldots, 1$, update from $l_2^{(-k)}$ to $l_2^{(-k+1)}$ by

(i) using $U_2^{(-k+1)}$ to draw a value l_2' directly from $\pi_{l_1^{(-k)}}(x)$

(ii) setting $Y_2 \equiv Y_2^{(-k+1)} = V_2^{(-k+1)} h_{l_1^{(-k)}}(l_2')$

(the height of the vertical slice),

(iii) setting $R_2 \equiv R_2^{(-k+1)} = h^{-1}_{l_1^{(-k)}}(Y_l)$

(the right endpoint of the horizontal slice),

(iv) setting $W_2' \equiv W_2'^{-k+1)} = -\frac{\beta_1}{\beta_{12}} W_2^{(-k+1)}$ and setting

$$l_1^{(-k+1)} = \begin{cases} W_2' & W_2' \leq R_2 \\ \frac{W_2' - R_2}{R - R_2} R_2 & W_2' > R_2, \end{cases}$$

where $R = -\frac{\beta_1}{\beta_{12}}$.

(4) Continue until time $T = \max(T_1, T_2)$ where T_i is the minimum n such that $W_i'^{(-n+1)} \leq R_i^{(-n+1)}$.

Simulation Results

We simulated 100,000 draws from the auto-exponential distribution given by (4.4) using $\beta_1 = 2$, $\beta_2 = 3$, and $\beta_{12} = -1$. Table 4.1 gives resulting estimates for the probabilities that draws come from (arbitrarily) selected regions in the plane and Figure 4.8 and Figure 4.9 show the estimated marginal densities for X_1 and X_2 along with curves showing the true marginal densities. The mean backward coupling time in 100,000 draws was 3.44 with a minimum of 1 and a maximum of 26. A histogram of backward coupling times is given in Figure 4.10.

Table 4.1: Estimating probabilities $\int\int_R \pi(x_1, x_2)$ for the auto-exponential distribution using proportions of simulated draws.

R	true value	95% conf. int.	est. value	abs. err.	rel. err.
$[0,1] \times [0,1]$	0.7340	(0.7301, 0.7357)	0.7329	0.0011	0.0016
$[0,0.5] \times [0,1]$	0.5136	(0.5104, 0.5167)	0.5128	0.0008	0.0015
$[0.2,3] \times [0,0.5]$	0.4812	(0.4780, 0.4843)	0.4823	0.0011	0.0023
$[0,1] \times [1,2]$	0.0547	(0.0533, 0.0561)	0.0548	0.0000	0.0009
$[1,3] \times [0,1.5]$	0.1955	(0.1930, 0.1980)	0.1971	0.0016	0.0082

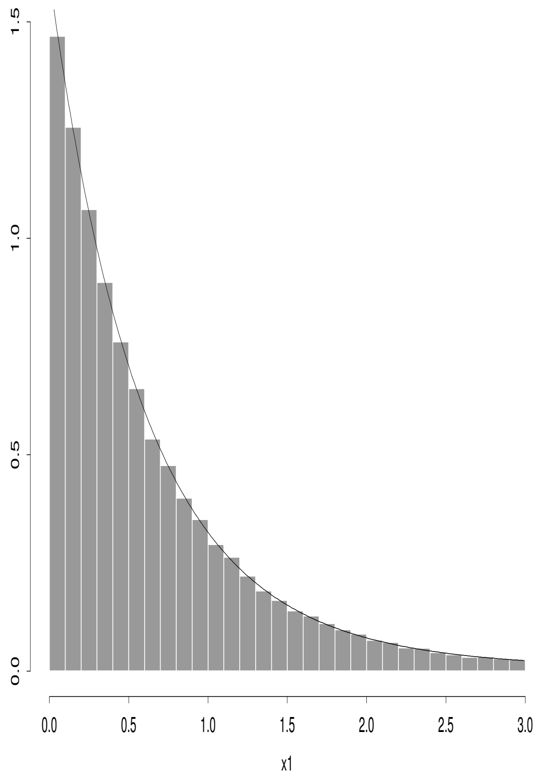


Figure 4.8: 100,000 draws from the auto-exponential distribution – the marginal distribution of X_1.

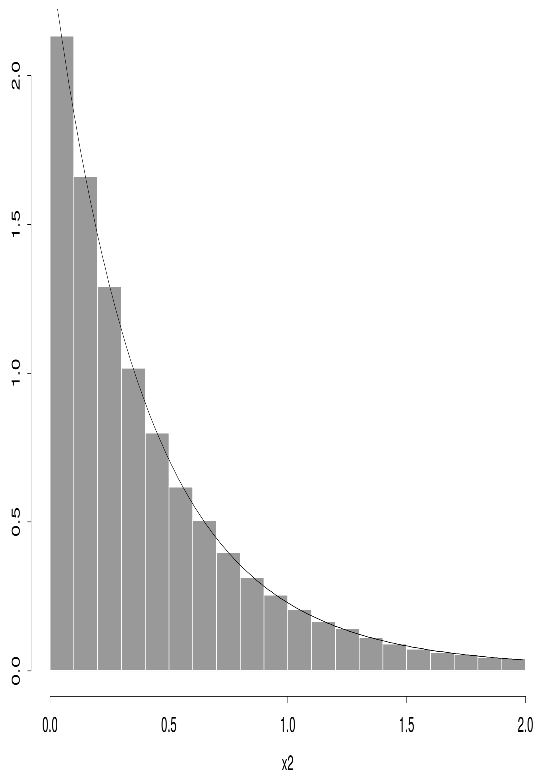


Figure 4.9: 100,000 draws from the auto-exponential distribution – the marginal distribution of X_2.

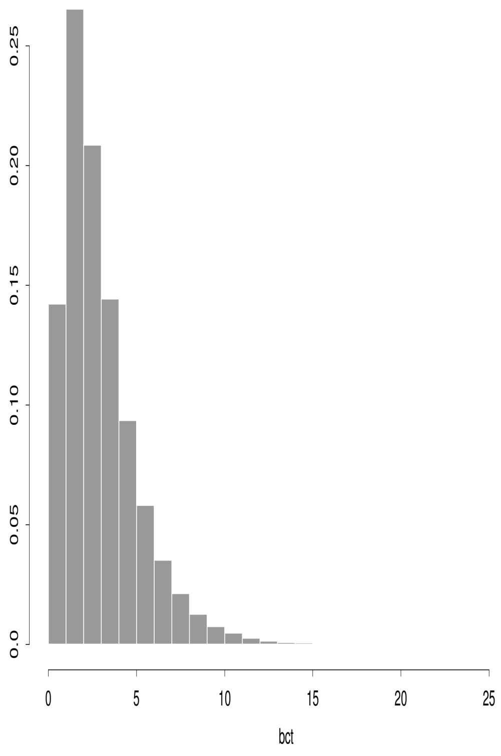


Figure 4.10: 100,000 draws from the auto-exponential distribution – the backward coupling times.

4.5.2 The Auto-Gamma Distribution

The following example is the natural counterpart to the auto-exponential distribution described in Section 4.5.1. We implement a slice coupling technique that uses an accept/reject step, in contrast to the folding coupler used in Section 4.5.1.

We consider the bivariate distribution

$$\pi(x_1, x_2) \propto x_1^{\alpha_1 - 1} x_2^{\alpha_2 - 1} \exp\{-\beta_1 x_1 - \beta_2 x_2 - \beta_{12} x_1 x_2\} \tag{4.6}$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2, \beta_{12}$ are positive. Note that the conditional densities are given by

$$X_1 | X_2 = x_2 \sim \Gamma(\alpha_1, \beta_1 + \beta_{12} x_2)$$
$$X_2 | X_1 = x_1 \sim \Gamma(\alpha_2, \beta_2 + \beta_{12} x_1).$$

The model (4.6) can be generalized to a k-variate density and is known as the *auto-gamma model* (see, for instance, Møller [39]). The algorithm described below, based on an idea of Kendall [32], can be easily modified to simulate from a k-variate density, however, for simplicity, we will restrict ourselves to the case $k = 2$.

Repulsive Gibbs chain

Since $\beta_{12} > 0$, this model is *repulsive* in the sense that a large (small) x_1 will produce a large (small) scaling parameter for the conditional gamma-density for X_2 and therefore result in a small (large) value for x_2 (and vice versa). If we wish to run a Gibbs sampler in order to draw values from $\pi(x_1, x_2)$, the Gibbs chain will be *stochastically anti-monotone* or *repulsive* with respect to the natural partial ordering on $\mathbb{R}^2$ as defined (4.5).

Running the chain perfectly

To run the chain in a perfect sampling setting, we wish to start in a *highest* and *lowest* point. In order to maintain an upper and lower process $u^{(n)} = (u_1^{(n)}, u_2^{(n)})$ and

$l^{(n)} = (l_1^{(n)}, l_2^{(n)})$, we use the upper process to update the lower chain and the lower process to update the upper chain.

$$\begin{aligned} \text{draw } u_1^{(n+1)} &\sim \Gamma(\alpha_1, \beta_1 + \beta_{12} l_2^{(n)}) \quad \text{and } l_1^{(n+1)} \sim \Gamma(\alpha_1, \beta_1 + \beta_{12} u_2^{(n)}) \\ \text{draw } u_2^{(n+1)} &\sim \Gamma(\alpha_2, \beta_2 + \beta_{12} l_1^{(n+1)}) \quad \text{and } l_2^{(n+1)} \sim \Gamma(\alpha_2, \beta_2 + \beta_{12} u_1^{(n+1)}). \end{aligned}$$

This idea due to Kendall [32] and implemented by Møller [39], will preserve monotonicity, so that all sample paths will be sandwiched between the lower and upper processes.

Møller [39] showed that once the lower and upper process differ only by an amount of ε, they will, with the updates described above, stay within a distance ε of each other (*ε-coupling*).

The "ε-perfect" sampling algorithm (which results in draws from a target distribution up to any desired accuracy ε) requires that one initializes lower and upper chains at successively more distant times in the past and run the two processes forward to time zero until the chains at time zero differ by at most ε (*ε-coupling*). Again we wish to emphasize that it is necessary to reuse random number streams as described in Section 2.1.

We now describe how to couple sample paths using the slice-sampling approach from Section 4.2.2 to achieve actual coalescence and therefore perfect draws from (4.6).

A bounding process

Since the state space is $(0, \infty) \times (0, \infty)$, $(0, 0)$ is clearly a "smallest point" to start the lower process, however there is no "highest point". As mentioned in the paragraph on "Monotonicity and Bounding Processes" in Section 2.2, we can instead use a bounding process which we find by making the following observation. The conditional distributions for X_i (which have a fixed shape parameter α_i) have a smallest scale parameter β_i, which will yield a "largest" X_i, so that $D_i \sim \Gamma(\alpha_i, \beta_i)$ is an upper bounding or "dominating" process for the i^{th} component of the Gibbs chain. We therefore initialize the lower and upper process with

$$l^{(0)} = (0,0) \quad \text{and} \quad u^{(0)} = (d_1, d_2),$$

where d_i are realizations of $D_i,\ i = 1,2$.

Updating and coupling sample paths

We will now describe in detail how to update the sample paths using slice sampling in order to get potentially common draws.

Assume we wish to update the first component of each (upper and lower) process. As with the auto-exponential model in Section 4.5.1, we will draw for both processes simultaneously.

We choose to describe the case for $\alpha_1 \leq 1$. The case $\alpha_1 > 1$ works analogously, although one has to keep in mind that the shape of the density function is quite different when doing the slice sampling step.

Suppose we need to draw $l_1^{(n+1)} \sim \Gamma(\alpha_1, \beta_l)$ and $u_1^{(n+1)} \sim \Gamma(\alpha_1, \beta_u)$ where $\beta_l = \beta_1 + \beta_{12} u_2^{(n)}$ and $\beta_u = \beta_1 + \beta_{12} l_2^{(n)}$ are the scale parameters defined by the second component of the upper and lower process, respectively. Note that we have $\beta_1 < \beta_u < \beta_l$. Since we do not need to know the constant of proportionally in order to do slice sampling, we use the likelihood $h(x) = (\beta x)^{\alpha-1} e^{-\beta x}$ to define the shape of the curve. For the sake of simplicity, let $h_1(x)$ denote the gamma likelihood with parameters α_1, β_1 for the dominating process and let $h_{l_1}(x)$ and $h_{u_1}(x)$ denote the likelihoods with parameters α_1, β_l and α_1, β_u, respectively.

Let s_1 and s_2 denote seeds to initialize the random number generator used. We describe the algorithm for a generic time step and omit the time index for ease of exposition, for example, $s_i = s_i^{(n)}$.

To update l_1 and l_2

(1) Seed the random number generator (RNG) using s_1.

(2) Draw (e.g. by an accept-reject method) $X \sim \Gamma(\alpha_1, \beta_1)$ from the dominating process.

(3) Draw $U \sim$ Uniform$(0,1)$ and set $Y = U \cdot h_1(X)$.

(4) Approximate the exact endpoint $R = h_1^{-1}(Y)$ defined by the horizontal slice $H(Y) = (0, h_1^{-1}(Y))$ by R' (where $R < R'$) and draw $X' \sim$ Uniform$(0, R')$ until $X' \in (0, R)$, updating R' by X' anytime $X' \notin (0, R)$. (Note that $X' \in (0, R)$ can be verified be checking if $h_1(X') < Y$. Also, note that such an approximation is only used as an intermediate step and that the ultimate outcome will not be an approximation.)

(5)
- If $h_{u_1}(X') < Y$ set $u_1 = X'$,
- otherwise seed the RNG using s_2, set $R'' = X'$, then
    (a) draw $X'' \sim$ Uniform$(0, R'')$.
    (b) If $h_{u_1}(X'') < Y$, set $u_1 = X''$, otherwise, set $R'' = X''$ and return to step 5(a).

(6)
- If $h_{l_1}(X') < Y$ set $l_1 = X'$,
- otherwise seed the RNG using s_2, set $R'' = X'$, then
    (a) draw $X'' \sim$ Uniform$(0, R'')$.
    (b) If $h_{l_1}(X'') < Y$, set $l_1 = X''$, otherwise, set $R'' = X''$ and return to step 6(a).

Remarks

(1) The update described above eventually yields potentially common values for u_1 and l_1 and therefore allows the upper and lower processes to eventually couple.

(2) Since it is not known a priori how many random numbers will be used to update the sample paths, it is convenient to store seeds instead. For this algorithm, two seeds are needed for each component and time step.

(3) To initialize the upper process, we need to draw the i^{th} component from a $\Gamma(\alpha_i, \beta_i)$ distribution. In order to run the sample paths preserving the Markov-property, the draw for the dominating process must be the X' obtained in steps $(1) - (4)$ in the algorithm described above.

(4) In the slice sampling described in Section 4.5.1, we needed to scale (and shift) the starting point X to make it a starting point for the other distribution(s) (say X_l and X_u) we are sampling from. In this case, this step is unnecessary since we would have $X_l = \frac{\beta_1}{\beta_l} X$ and $X_u = \frac{\beta_1}{\beta_u} X$. However, by omitting certain constants in the choice of h, we get that $h_l(X_l) = h_u(X_u) = h(X)$, so that the y-value which defines the horizontal slice is the same for all three distributions.

(5) In the 6 steps above, we describe only the update in one component for one time step $-n \to -n+1$. To actually run the Perfect-Sampling algorithm, this has to be extended for the second component as well as for starting points at successively further distant times in the past until the lower and upper process have coupled at time zero. The random seeds must be reused for the corresponding time steps.

Simulation results

Again, we simulated 100,000 draws from the distribution given by (4.6) using $\alpha_1 = \alpha_2 = 0.5$ and $\beta_1 = 2, \beta_2 = 3, \beta_{12} = 1$. Table 4.2 gives resulting estimates for the probabilities that draws come from (arbitrarily) selected regions in the plane and Figure 4.11 and Figure 4.12 show the estimated marginal densities for X_1 and X_2 along with curves showing the true marginal densities. The mean backward coupling time in 100,000 draws was 1.07 with a minimum of 1 and a maximum of 3.

Table 4.2: Estimating probabilities $\int\int_R \pi(x_1, x_2)$ for the auto-gamma distribution using proportions of simulated draws.

R	true value	95% conf. int.	est. value	abs. err.	rel. err.
$[0, 0.5] \times [0, 0.2]$	0.6306	(0.6276,0.6336)	0.6280	0.0025	0.0040
$[0.2, 1] \times [0.5, 2]$	0.0201	(0.0192,0.0210)	0.0204	0.0003	0.0134
$[0.1, \infty] \times [0.2, 3]$	0.1245	(0.1225,0.1266)	0.1241	0.0004	0.0031
$[0.2, 2] \times [0, 1]$	0.3476	(0.3447,0.3506)	0.3477	0.0001	0.0004

4.5.3 Storage Model

We now illustrate the application of the folding and shifting coupler for a finite storage system on $[0, K]$ with independent and identically distributed exponential replenishments with mean $1/\mu$ at the arrival times of a Poisson process with rate λ. Excessive input above $K < \infty$ is considered overflow and cannot be saved for future use. Between arrivals, content is released deterministically at rate $r(u)$. The Markov chain embedded just prior to arrival times satisfies the PASTA property (see Asmussen [1]) which ensures that its stationary distribution is identical to the stationary distribution of the continuous time chain.

The assumption of finiteness of the system is not critical as we may remove this restriction using an upper bounding random process as in Tweedie and Corcoran [57]. This idea is due to Kendall [32], David Wilson (private communication) has also developed a system for doing this.

We shall consider specifically the case $r(u) = \beta u$ for $\beta > 0$. For $x \in (0, K]$, it can be shown (Lund, private communication) that the density $\pi(x)$ of the stationary

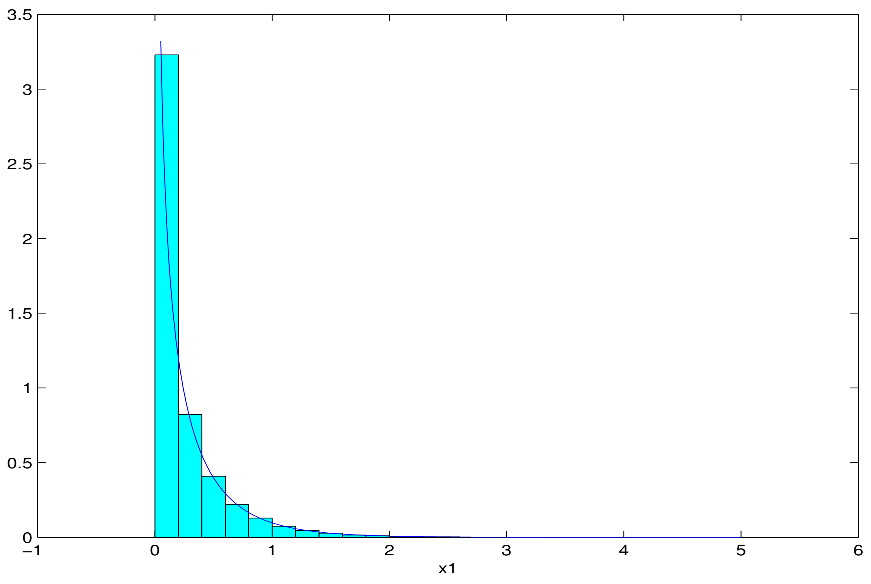


Figure 4.11: 100,000 draws from the auto-gamma distribution – the marginal distribution of X_1.

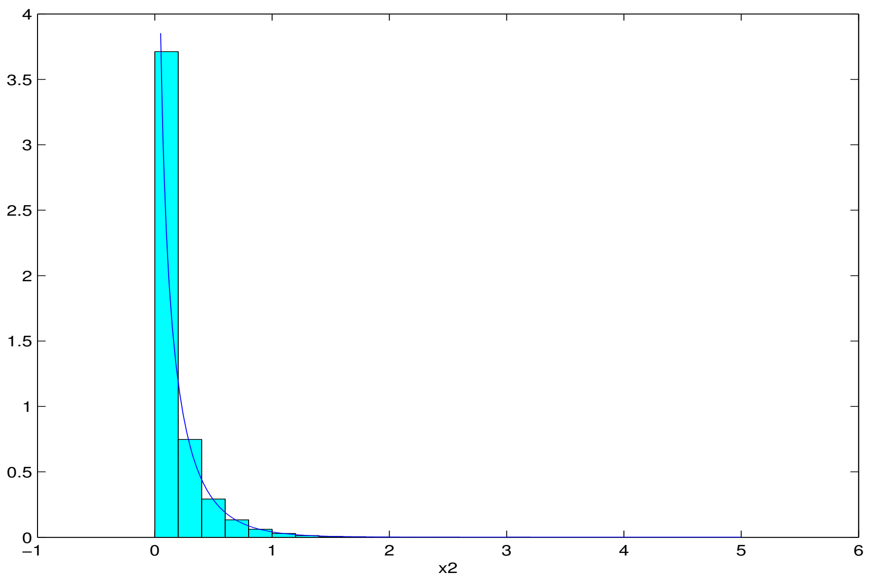


Figure 4.12: 100,000 draws from the auto-gamma distribution – the marginal distribution of X_2.

distribution π is given by

$$\pi(x) = \frac{x^{(\lambda\beta^{-1}-1)}e^{-\mu x}}{\int_0^K x^{(\lambda\beta^{-1}-1)}e^{-\mu x}dx} \quad \text{for } x \in (0, K], \tag{4.7}$$

but in general the denominator cannot be integrated.

Corcoran and Tweedie [10] and Tweedie and Corcoran [57] describe perfect sampling algorithms applicable to this system involving minorizations and chain splitting, but the following described folding and shifting technique is much easier to apply.

The "natural" way to simulate an embedded time step of this storage process is to make a transition from state x to state y as follows.

(1) Draw replenishment jump and inter-arrival values j and t from the exponential distributions with rates μ and λ, respectively.

(2) Jump to $z = \min(x + j, K)$.

(3) End at $y = ze^{-t}$. (This is the result of the release rate $r(u) = u$.)

Alternatively, we may make this transition as follows.

(1) Draw a jump value j from the exponential distribution with rate μ.

(2) Jump to $z = \min(x + j, K)$.

(3) End at value y drawn from the density $f_Y^z(y) = \frac{\lambda}{z^\lambda} y^{\lambda-1} I_{(0,z)}(y)$.

The density in step 3 is simply that of $Y = ze^{-T}$ where T is exponentially distributed with rate λ. Now, to attempt to couple two sample paths with current values x_1 and x_2, we make a transition as follows.

(1) Draw a jump value j from the exponential distribution with rate μ.

(2) Jump the paths to $z_1 = x_1 + j$ and $z_2 = x_2 + j$.

(3) Use shifting and folding to draw values (potentially common) from $f_Y^{z_1}$ and $f_Y^{z_2}(y)$.

We consider here only the case $\lambda > 1$. The relative shapes of the two densities in step (3) are shown in Figure 4.13. We obtain (potentially common) draws from these distributions by first using the shift method described in Section 4.3.2 to draw values from $f_Y^{z_2}(y)$ and

$$f(y) = \frac{\lambda}{z_2^{\lambda}}(y + z_2)^{\lambda-1} I_{(z_1 - z_2, z_1)}(y)$$

(one can imagine sliding the wider density to the left until the vertical lines match up) and then folding this shifted wide draw (potentially negative at this point) using (4.1).

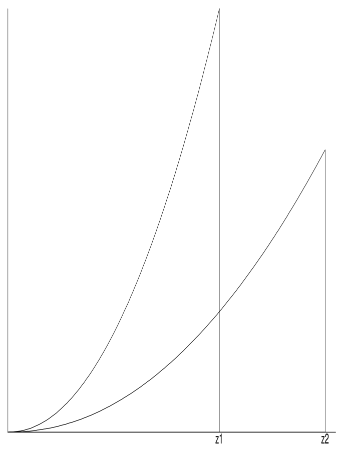


Figure 4.13: Relative shapes of two densities for $f_Y^{z_1}$ and $f_Y^{z_2}(y)$

Chapter 5

Bayesian Variable Selection in a Linear Regression Model

5.1 Introduction

In this chapter, we describe the use of perfect sampling algorithms for Bayesian variable selection. Variable selection is an important and well-studied problem in many areas. There are various approaches to solve this problem – among them Bayesian methods that use MCMC-algorithms. **The novelty of our approach is the use of exact methods that completely eliminate convergence issues which would arise when applying regular MCMC techniques.** This chapter was sparked by the paper of Huang and Djurić [31] who solve a basic case of a variable selection problem using perfect sampling and state the need for algorithms to solve more general cases.

The chapter is organized as follows. We formulate the problem of variable selection in a linear regression model in Section 5.2. We describe perfect simulation methods from the posterior of the Bayesian model in Section 5.3. We begin by describing Huang and Djurić's [31] approach using perfect simulation in Section 5.3.1. The remainder of the section is devoted to generalizing the model step by step, allowing more and more components to be random. We specify perfect sampling algorithms that can be used to solve these different cases by incorporating a Gibbs sampler along with slice coupling techniques from Chapter 4 as well as the IMH algorithm [1] from Chapter 3. At last, we present promising results in Section 5.4 that were found testing the different algorithms

[1] In this chapter, by "IMH algorithm", we refer to the perfect version from Section 3.3.1.

on artificial data sets. We show tests on a real data set in Section 5.5.

5.2 The Problem of Variable Selection

This section deals with a model for variable selection and describes the approach we chose to solve it. For a general overview of the variable selection problem and relevant references, see for example George [22] or the website [38] with links to many papers on model/variable selection.

Consider data records given by a linear regression model with i.i.d. Gaussian noise

$$\boldsymbol{y} = \gamma_1\theta_1\boldsymbol{x}_1 + \cdots + \gamma_r\theta_r\boldsymbol{x}_r + \boldsymbol{\varepsilon}, \tag{5.1}$$

where $\boldsymbol{y}$ is an $n \times 1$ response vector, $\boldsymbol{x}_i \in \mathrm{I\!R}^{\mathrm{n}}$ are $n \times 1$ vectors of predictors, $\theta_i \in \mathrm{I\!R}$ are the corresponding coefficients, and $\gamma_i \in \{0, 1\}$ are indicators taking values 0 and 1 for $i = 1, \ldots, r$. The noise vector $\boldsymbol{\varepsilon} \in \mathrm{I\!R}^{\mathrm{n}}$ is assumed to have independent components with $\varepsilon_j \sim N(0, \sigma^2)$ for $j = 1, \ldots, n$.

5.2.1 The Goal

Our task is to recover from the data the subset of the r predictors that are a part of the model, that is, we want to determine the values of the indicators $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_r)'$.

We approach this problem from a Bayesian perspective, selecting as an optimal $\boldsymbol{\gamma}$ the one that appears most frequently when sampling from the posterior density

$$\pi_{\boldsymbol{\Gamma},\sigma^2,\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\gamma}, \sigma^2, \boldsymbol{\theta}|\boldsymbol{y}) \propto L(\boldsymbol{\gamma}, \sigma^2, \boldsymbol{\theta})g(\boldsymbol{\gamma}, \sigma^2, \boldsymbol{\theta}) \tag{5.2}$$

of the parameters given the data. Here, $L(\cdot)$ is the likelihood, $g(\cdot)$ is a prior for the parameters, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_r)'$ and $\boldsymbol{\gamma}$ and σ are as above.

This is a common way to address variable selection (see for example George and McCulloch, and Chipman et al. [23, 24, 25, 6]). As mentioned in Section 5.1, generating samples from the posterior is usually achieved with regular (approximate) MCMC methods – the significance of this chapter is to develop and extend exact methods for this purpose.

5.2.2 The Priors

We use the standard normal-gamma conjugate class of priors suggested by Raferty et al. [47],

$$\frac{\lambda\nu}{\sigma^2} \sim \chi^2(\nu) \longrightarrow Z := \frac{1}{\sigma^2} \sim \Gamma(\frac{\nu}{2}, \frac{\lambda\nu}{2}), \tag{5.3}$$

$$\boldsymbol{\theta} \sim N(\boldsymbol{\xi}, \sigma^2\mathbf{V}) \tag{5.4}$$

in addition to the non-informative prior for $\boldsymbol{\gamma}$,

$$\gamma_i \overset{iid}{\sim} \text{Bernoulli}(\frac{1}{2})\ i = 1 \ldots, q \tag{5.5}$$

Here, λ and ν and $\mathbf{V}$ are hyperparameters to be chosen.

5.3 Perfect Simulation From the Posterior

In Section 5.3.1 we describe an existing algorithm of Huang and Djurić [31] for perfect sampling from (5.2) when σ^2 and $\boldsymbol{\theta}$ are fixed and known. In the remaining sections we gradually allow for other layers of randomness.

5.3.1 Fixed Variance, Fixed Coefficients

For fixed σ^2 and $\boldsymbol{\theta}$, the posterior distribution of the indicators in (5.2) with the uniform noninformative prior on the components of $\boldsymbol{\gamma}$ is

$$\pi_{\mathbf{\Gamma}|\boldsymbol{Y}}(\boldsymbol{\gamma}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{q}\sum_{k=1}^{r}\gamma_j\gamma_k\theta_j\theta_k\boldsymbol{x}_j'\boldsymbol{x}_k + \frac{1}{\sigma^2}\sum_{j=1}^{r}\gamma_j\theta_j\boldsymbol{x}_j'\boldsymbol{y}\right). \tag{5.6}$$

The full conditional distributions are then given by

$$\begin{aligned}\pi(\gamma_i = 1|\boldsymbol{\gamma}_{-i}, \boldsymbol{y}) &= \frac{Pr(\gamma_i = 1, \boldsymbol{\gamma}_{-i}|\boldsymbol{y})}{Pr(\gamma_i = 0, \boldsymbol{\gamma}_{-i}|\boldsymbol{y}) + Pr(\gamma_i = 1, \boldsymbol{\gamma}_{-i}|\boldsymbol{y})} \\ &= \left[1 + \exp\left(\frac{1}{\sigma^2}\sum_{\substack{j=1\\ j\neq i}}^{r} \gamma_j\theta_i\theta_j \boldsymbol{x}_i'\boldsymbol{x}_j + \frac{1}{2\sigma^2}\theta_i^2\boldsymbol{x}_i'\boldsymbol{x}_i - \frac{1}{\sigma^2}\theta_i\boldsymbol{x}_i'\boldsymbol{y}\right)\right]^{-1} \quad (5.7)\end{aligned}$$

where $\boldsymbol{\gamma}_{-i} = (\gamma_1, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_r)$.

A first-pass perfect sampling approach would be to look for a monotonicity structure (mentioned in the paragraph on "Monotonicity and Bounding Processes" in Section 2.2) and to consider the partial ordering

$$\Gamma_1 \preceq \Gamma_2$$

when any component that is a 1 in Γ_1 is also a 1 in Γ_2. One could then consider the "top" and "bottom" of the space to be $(1, \ldots, 1)$ and $(0, \ldots, 0)$, respectively. Unfortunately, Huang and Djurić [31] show that the standard Gibbs update (see Appendix A) does not, in general, preserve the monotonicity property required for the feasibility of a perfect sampling scheme for large r. Instead they make clever use of the Gibbs coupler which is a *support set coupling* technique that we now describe.

To begin, we require lower and upper bounding processes for the Gibbs update given by (5.7). That is, we seek, for $n = 1, 2, \ldots$, processes $\{L_{-n}\}$ and $\{U_{-n}\}$ so that

$$L_{-n} \preceq X_{-n} \preceq U_{-n},$$

where $\{X_{-n}\}$ is the r-dimensional Gibbs chain with stationary distribution given by (5.6) and updates given by (5.7).

Huang and Djurić [31] construct $\{L_{-n}\}$ and $\{U_{-n}\}$ by assigning, for any time step, the i^{th} components to be 1 with probabilities

$P_L(\gamma_i = 1)$ and $P_U(\gamma_i = 1)$, respectively, where these probabilities are such that

$$P_L(\gamma_i = 1) \leq \pi(\gamma_i = 1|\boldsymbol{\gamma}_{-i}, \boldsymbol{y}) \leq P_U(\gamma_i = 1).$$

To obtain such *sandwich distributions*, we begin by breaking the sum in (5.7) into two parts

$$\pi(\gamma_i = 1|\boldsymbol{\gamma}_{-i}, \boldsymbol{y}) =$$

$$\left[1 + \exp\left\{\frac{1}{\sigma^2}\left(\sum_{j\in C}\gamma_j\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \sum_{j\notin C}\gamma_j\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \frac{1}{2}\theta_i^2\boldsymbol{x}_i'\boldsymbol{x}_j - \theta_i\boldsymbol{x}_i'\boldsymbol{y}\right)\right\}\right]^{-1} \quad (5.8)$$

where C is the set of indices of all of the components that have coupled successfully by this particular time step. (We have suppressed the time notation.)

For the γ-components referenced by C, all three processes are equal, and for the γ-components not in C, the current state of these components are still unknown. Using the second sum in (5.8) we can make the overall expression smallest if we set γ_j with $j \notin C$ to be 1 when the coefficient $\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j$ is positive and, likewise, we can make it largest if we set γ_j with $j \notin C$ to be 1 when the coefficient $\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j$ is negative. The other γ-components are set to zero.

So, we have

$$P_L(\gamma_i = 1) = [1 + \exp\{\frac{1}{\sigma^2}(\sum_{j\in C}\gamma_j\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \sum_{j\in P}\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \frac{1}{2}\theta_i^2\boldsymbol{x}_i'\boldsymbol{x}_i - \theta_i\boldsymbol{x}_i'\boldsymbol{y})\}]^{-1} \quad (5.9)$$

and

$$P_U(\gamma_i = 1) = [1 + \exp\{\frac{1}{\sigma^2}(\sum_{j\in C}\gamma_j\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \sum_{j\in N}\theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j + \frac{1}{2}\theta_i^2\boldsymbol{x}_i'\boldsymbol{x}_i - \theta_i\boldsymbol{x}_i'\boldsymbol{y})\}]^{-1} \quad (5.10)$$

where

$$P = \{j \mid \theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j > 0\} \qquad \text{and} \qquad N = \{j \mid \theta_i\theta_j\boldsymbol{x}_i'\boldsymbol{x}_j < 0\}.$$

Rather than following the values of the γ-components through time, support set coupling keeps track of the **potential** values generated by the Gibbs sampler. For each $i = 1, 2, \ldots, r$, we will determine, at each time step, if γ_i is 0, 1, or undecided. That is, we will assign its support set, S_i, to be $\{0\}$, $\{1\}$, or $\{0, 1\}$. It is most difficult to obtain, for the value of γ_i, a 1 in the lower process and a 0 in the upper process. Hence, if the former is achieved, all possible sample paths will have γ_i set equal to 1 ($S_i = \{1\}$). In the latter case, all possible sample paths will have γ_i set equal to zero ($S_i = \{0\}$). If neither case arises, we will leave the value of γ_i in our sample path of interest undecided ($S_i = \{0,1\}$). More specifically, at each backward time step $n = 1, 2, \ldots$ and for each component $i = 1, 2, \ldots r$, we draw a Uniform$(0,1)$ random variable V, ($V = V_i^{-n}$), and set

$$S_i = \begin{cases} \{0\}, & V > P_U(\gamma_i = 1) \\ \{1\}, & V < P_L(\gamma_i = 1) \\ \{0,1\}, & \text{otherwise.} \end{cases}$$

Coupling is achieved when all r components have a singleton support set.

We wish to strongly emphasize here that we have suppressed important notation in an attempt to not (we hope) overwhelm the reader in order to get some general ideas across. In particular, we have suppressed

(1) General Time Notation

As mentioned in Section 2.1, the organization of random number streams, that is, the re-use of random numbers, is critical to the success of any perfect simulation algorithm. Newcomers to perfect simulation algorithms might best be served by first gaining an understanding of the basic ideas involved. We again refer interested readers to Casella and Robert [4].

(2) Gibbs Component Time Notation

In the update probabilities given by (5.9) and (5.10), we are revising the r γ-components in order at each time step. If we use $\gamma_i^{(n)}$ to represent the value of the i^{th} component at time n, the update for $\gamma_i^{(n)}$ involves $\gamma_j = \gamma_j^{(n)}$ for $j = 1, \ldots, i-1$ and $\gamma_j = \gamma_j^{(n-1)}$ for $j = i+1, \ldots, r$.

Expanded notation may be found in Huang and Djurić [31].

5.3.2 Random Variance, Fixed Coefficients

We first extend the model by incorporating a random variance for the Gaussian noise according to (5.3). Assuming that the coefficient-vector $\boldsymbol{\theta}$ is fixed, the posterior distribution becomes

$$\pi_{\mathbf{\Gamma},Z|\boldsymbol{Y}}(\boldsymbol{\gamma},z|\boldsymbol{y}) \propto z^{\frac{n}{2}+\frac{\nu}{2}-1}\exp\{-\frac{1}{2}z[\lambda\nu+(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)'(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)]\}. \quad (5.11)$$

Since we intend to apply a Gibbs sampler (see Appendix A) again, we need to look at the conditional probabilities. It is easy to see that

$$Z|\mathbf{\Gamma},\boldsymbol{Y} \sim \Gamma\left(\frac{n+\nu}{2},\frac{1}{2}[(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)'(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)+\lambda\nu]\right) \quad (5.12)$$

and that the posterior conditional distribution for the vector of indicators $\boldsymbol{\gamma}$ given Z is

$$\pi_{\mathbf{\Gamma}|\boldsymbol{Y},Z}(\boldsymbol{\gamma}|\boldsymbol{y},z) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{q}\sum_{k=1}^{r}\gamma_j\gamma_k\theta_j\theta_k\boldsymbol{x}_j'\boldsymbol{x}_k+\frac{1}{\sigma^2}\sum_{j=1}^{r}\gamma_j\theta_j\boldsymbol{x}_j'\boldsymbol{y}\right). \quad (5.13)$$

Note that the distribution for $\mathbf{\Gamma}|\boldsymbol{Y},Z$ is the same as in (5.6) since $Z=z$ is fixed. So once we know the value for Z, we can use the support set coupling from Section 5.3.1 to simulate perfectly from $\mathbf{\Gamma}|\boldsymbol{Y},Z$. On the other hand, if we know the value for $\mathbf{\Gamma}$, we can draw from a gamma distribution for $Z|\mathbf{\Gamma},\boldsymbol{Y}$. Hence, we can run sample paths of a "bivariate" Markov chain that has stationary distribution given by (5.11). Note that we are thinking of $\boldsymbol{\gamma}$ and Z as two random variables as opposed to $r+1$ random variables $(z,\gamma_1,\ldots,\gamma_r)'$.

Our problem now is to consider coupling together these Gibbs-generated sample paths. Note that we will be describing a perfect simulation within a perfect simulation.

The Coupling

The key observation for simulating from (5.11) is that the conditional gamma distribution for $Z|\mathbf{\Gamma},\boldsymbol{Y}$ has a **fixed** shape parameter $\alpha=\frac{n+\nu}{2}$ and scale parameter $\beta(\boldsymbol{\gamma})=\frac{1}{2}[(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)'(\boldsymbol{y}-\sum_{i=1}^{r}\gamma_i\theta_i\boldsymbol{x}_i)+\lambda\nu]$ that can be bounded above and below **independently of $\boldsymbol{\gamma}$** (see the next paragraph for details),

$$\beta_{min} \leq \beta(\boldsymbol{\gamma}) \leq \beta_{max} \qquad \forall \boldsymbol{\gamma} \in \{0,1\}^r. \tag{5.14}$$

If we can get a common draw for $\Gamma(\alpha, \beta_{min})$ and $\Gamma(\alpha, \beta_{max})$, it will be a draw from $\Gamma(\alpha, \beta)$ for all β such that $\beta_{min} \leq \beta \leq \beta_{max}$, and therefore a draw for $Z|\boldsymbol{\Gamma}, \boldsymbol{Y}$ no matter what the particular values for $(\gamma_1, \ldots, \gamma_r)$ were! This can be achieved by the slice coupling approach described in Chapter 4 and we address how it applies to this problem in more detail in the Paragraph "Slicing for the Z-Component" below.

So, at each time point $-n$, $n = 1, 2, \ldots$, we begin by drawing two values for Z, one from each of the distributions $\Gamma(\alpha, \beta_{min})$ and $\Gamma(\alpha, \beta_{max})$ using the slice coupling approached described in Chapter 4 until the first time point $-n^*$ where we get a common value for Z. Once this value is locked in, all ambiguity is removed and we move forward in time from $-n^*$ to zero with a Gibbs sampler where we alternate at each time point between

(1) using Huang and Djurić's algorithm (Section 5.3.1) in its entirety to sample a vector of γ components given Z, and

(2) sampling a new Z from the resulting gamma distribution described by (5.12) for the fixed vector $\boldsymbol{\gamma}$ from step 1.

Bounding β

Expanding the expression for $\beta(\boldsymbol{\gamma})$, we get

$$\beta(\boldsymbol{\gamma}) = \frac{1}{2}(\boldsymbol{y}'\boldsymbol{y} + \lambda\nu) - \sum_{i=1}^{r} \gamma_i \theta_i \boldsymbol{x}_i' \boldsymbol{y} + \frac{1}{2} \sum_{i=1}^{r} \gamma_i^2 \theta_i^2 \boldsymbol{x}_i' \boldsymbol{x}_i + \sum_{i<j} \gamma_i \gamma_j \theta_i \theta_j \boldsymbol{x}_i' \boldsymbol{x}_j.$$

By defining

$$\begin{aligned} I_1 &= \{\, i \mid \theta_i \boldsymbol{x}_i' \boldsymbol{y} \geq 0\} \\ I_2 &= \{\, i \mid \theta_i \boldsymbol{x}_i' \boldsymbol{y} < 0\} \\ M_1 &= \{\, (i,j) \mid \theta_i \theta_j \boldsymbol{x}_i' \boldsymbol{x}_j \geq 0\} \\ M_2 &= \{\, (i,j) \mid \theta_i \theta_j \boldsymbol{x}_i' \boldsymbol{x}_j < 0\} \end{aligned}$$

and

$$\begin{aligned} \beta_{min} &:= \max\{\frac{1}{2}\lambda\nu, \frac{1}{2}(\boldsymbol{y}'\boldsymbol{y} + \lambda\nu) - \sum_{i \in I_1} \theta_i \boldsymbol{x}_i' \boldsymbol{y} + \sum_{(i,j) \in M_2} \theta_i \theta_j \boldsymbol{x}_i' \boldsymbol{x}_j\} \\ \beta_{max} &:= \frac{1}{2}(\boldsymbol{y}'\boldsymbol{y} + \lambda\nu) - \sum_{i \in I_2} \theta_i \boldsymbol{x}_i' \boldsymbol{y} + \frac{1}{2}\sum_{i=1}^{r} \gamma_i^2 \theta_i^2 \boldsymbol{x}_i' \boldsymbol{x}_i + \sum_{(i,j) \in M_1} \theta_i \theta_j \boldsymbol{x}_i' \boldsymbol{x}_j \end{aligned}$$

(5.14) is easily verified.

Slice Coupling For the Z-Component

We now demonstrate how to couple the z-component using the slice sampling approach from Chapter 4 in more detail.

Our goal is to get a potentially common draw from two different gamma distributions, $\Gamma(\alpha, \beta_{min})$ and $\Gamma(\alpha, \beta_{max})$. In Section 4.2.2 we demonstrate that we only need to know the density function up to a constant, so we choose $h_1(x) = (\beta_{min} x)^{\alpha-1} e^{-\beta_{min} x}$, $x > 0$, and $h_2(x) = (\beta_{max} x)^{\alpha-1} e^{-\beta_{max} x}$, $x > 0$, respectively. We assume that $\alpha \leq 1$ (the case where $\alpha > 1$ works in a similar manner) and we clearly have $\beta_{min} < \beta_{max}$. Note that the horizontal slice $H(y)$ in the last step of the slice sampling procedure is of the form $(0, r)$. Also note that a **smaller** scale parameter β will yield a **larger** gamma variable and a **wider** horizontal slice $H(y)$.

(1) Draw $x \sim \Gamma(\alpha, 1)$, let $x_1 = \frac{x}{\beta_{min}}$, and $x_2 = \frac{x}{\beta_{max}}$.

(2) Draw $u \sim$ Uniform$(0, 1)$ and let $y_1 = u h_1(x_1)$, and $y_2 = y_1$.

(3) Approximate the endpoint with r_1 of the wider slice and

  (a) draw $u' \sim \text{Uniform}(0, r_1)$,

  (b) if $h_1(u') > y_1$, accept $x'_1 = u'$ otherwise update $r_1 = u'$ and go to step (1).

(4) If $h_2(x'_1) > y_2$, accept $x'_2 = x'_1$ – a **common draw**!

Otherwise (accept-reject), approximate the endpoint $r_2 = x'_1$ and

  (a) draw $u' \sim \text{Uniform}(0, r_2)$,

  (b) if $h_2(u') > y_2$, accept $x'_2 = u'$

  otherwise update $r_2 = u'$ and go to step (1).

Remarks

- Note that our choice of h_1 and h_2 yields $h_1(x_1) = h_2(x_2)$ (with the notation from step (1)), which justifies setting $y_2 = y_1$ in step (2).
- Even though the pdf of a gamma distribution is non-invertible, this did not create a problem in steps (3) and (4) of the slice sampling procedure. We may be approximating the right endpoint of the horizontal slice, but we are drawing uniformly from the true horizontal slice in an accept/reject procedure by checking whether the draw from the approximated slice falls under the density or not.
- The approximation r_1 of the right endpoint of the wider slice can be computed by, for example, approaching the true endpoint "from the left" (e.g. starting at 0) and adding small increments until the true endpoint is exceeded. This approximation (as well as the approximation r_2 for the endpoint of the narrower slice) will be updated during the algorithm to increase efficiency.

Sampling For the Variance and Indicators Together

An algorithm to draw from (5.11) is given by the following.

(1) Find a *backward coupling time* T: for each time $t = 0, -1, -2, \ldots$

   (a) Draw z_t^{min} and z_t^{max} according to $\Gamma(\alpha, \beta_{min})$ and $\Gamma(\alpha, \beta_{max})$ using the slice sampling procedure described in the Paragraph "Slicing for the Z-Component" above.

   (b) If $z_t^{min} = z_t^{max}$, set $T \leftarrow t$ and go to step (2) (**coalescence**).

   (c) Otherwise, set $t \leftarrow t - 1$ and go to step (1) (a)

(2) Draw $\boldsymbol{\gamma}_T \sim \boldsymbol{\Gamma} | (\boldsymbol{Y}, Z = z_T)$ using the support set coupling from Section 5.3.1.

(3) For each time $t = T + 1, \ldots, -1, 0$ (**outer Gibbs sampler**)

   (a) Draw $z_t \sim Z \; | (\boldsymbol{Y}, \boldsymbol{\Gamma} = \boldsymbol{\gamma}_{t-1})$ using the slice sampling procedure from the above Paragraph "Slicing for the Z-Component".

   (b) Draw $\boldsymbol{\gamma}_t \sim \boldsymbol{\Gamma} \; | (\boldsymbol{Y}, Z = z_t)$ using the support set coupling from Section 5.3.1 (**inner Gibbs sampler**).

(4) $(\boldsymbol{\gamma}_0, z_0)$ is a draw from the posterior (5.11)!

5.3.3 Random Variance, Random Coefficients

In this section, we incorporate both a random variance and random coefficients with prior distributions according to (5.3) and (5.4). While we originally treated the case of a fixed variance and random coefficients separately, we have realized that that case can be collapsed into the more general case of this section.

Combining γ and θ

To reduce the size of the state space, we combine the random components γ_i and θ_i by defining $\beta_i := \gamma_i\theta_i$ $(i = 1, \ldots, r)$ to have the appropriate mixture distribution. Ultimately, since we are maximizing a marginal distribution, we are only interested in the values for $\boldsymbol{\gamma}$ which can be "recovered" from $\boldsymbol{\beta}$ by setting

$$\gamma_i = \begin{cases} 0 & \text{if } \beta_i = 0 \\ 1 & \text{if } \beta_i \neq 0 \end{cases} \qquad i = 1, \ldots, q.$$

Let $g_{\boldsymbol{B}}, Z(\boldsymbol{\beta}, z)$ denote the joint pdf for the prior distribution on $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_r)'$ and z which satisfies

$$g_{\boldsymbol{B},Z}(\boldsymbol{\beta}, z) = g_{\boldsymbol{B}|Z}(\boldsymbol{\beta}|z)g_Z(z) \propto g_{\boldsymbol{B}|Z}(\boldsymbol{\beta}|z) \times z^{\frac{\nu}{2}-1}\exp\{-\frac{\lambda\nu}{2}z\},$$

where $g_{\boldsymbol{B}|Z}(\boldsymbol{\beta}|z)$ denotes the density for the distribution where $\boldsymbol{\beta} \sim N(\boldsymbol{\xi}, \frac{1}{z}\boldsymbol{V})$, but each component $\beta_i = 0$ with probability $\frac{1}{2}$ $(i = 1, \ldots, r)$.

After combining γ and θ, the likelihood can be written as

$$L(\boldsymbol{\beta}, z) = z^{\frac{n}{2}}\exp\{-\frac{z}{2}(\boldsymbol{y} - \sum_{i=1}^{r}\beta_i\boldsymbol{x}_i)'(\boldsymbol{y} - \sum_{i=1}^{r}\beta_i\boldsymbol{x}_i)\}$$

and the posterior distribution we want to draw from is given by

$$\pi_{\boldsymbol{B},Z|\boldsymbol{Y}}(\boldsymbol{\beta}, z|\boldsymbol{y}) \propto L(\boldsymbol{\beta}, z) \times g_{\boldsymbol{B},Z}(\boldsymbol{\beta}, z). \tag{5.15}$$

Using IMH

To simulate from the posterior (5.15), we apply the "bounded" version of the IMH algorithm from Section 3.3.2. We choose $q(\boldsymbol{\beta}, z) \propto z^{\frac{\nu}{2}-1}g_{\boldsymbol{B},Z}(\boldsymbol{\beta}, z)$ as the candidate distribution since with this choice, the $\frac{\pi}{q}$ ratio satisfies $\max\frac{\pi}{q} = \max\limits_{(\boldsymbol{\beta},z)}\frac{L(\boldsymbol{\beta},z)}{z^{\frac{n}{2}}} = \max\limits_{(\boldsymbol{\beta},z)}\exp\{-\frac{z}{2}(\boldsymbol{y} - \sum\limits_{i=1}^{r}\beta_i\boldsymbol{x}_i)'(\boldsymbol{y} - \sum\limits_{i=1}^{r}\beta_i\boldsymbol{x}_i)\} \leq 1.$

Sampling from the Candidate

Using what we call *hierarchical sampling* (which is described in some detail in Appendix B), we can implement the following steps to get samples $(z, \boldsymbol{\beta}) \sim q(\boldsymbol{\beta}, z)$

(1) draw $Z \sim \Gamma(\frac{n+\nu}{2}, \frac{\lambda\nu}{2})$,

(2) draw $B|Z \sim N(\boldsymbol{\xi}, \frac{1}{z}\boldsymbol{V})$,

(3) set $\beta_i = 0$ with probability $\frac{1}{2}$ $(i = 1 \ldots, q)$.

Since we can simulate from the candidate density $q(\boldsymbol{\beta}, z)$, we can specify the IMH algorithm to simulate from (5.15).

```
(1) Find a backward coupling time $T$: for each time $t = 0, -1, -2, \ldots$
    (a) Draw $(\boldsymbol{\beta}_t, z_t) \sim q(\cdot)$ and $u_t \sim$ Uniform$(0, 1)$.
    (b) If $u_t \leq L(\boldsymbol{\beta}_t) \cdot 1$, set $T \leftarrow t$ and go to step (coalescence).
    (c) Otherwise, set $t \leftarrow t - 1$ and go to step (1) $(a)$.
(2) Set $X_T = q_T$ and for each time $t = T + 1, T + 2 \ldots, 0$:
    (a) If $u_t \leq \frac{L(X_{t-1})}{L(\boldsymbol{\beta}_t)}$ (accept the candidate), set $X_t = (\boldsymbol{\beta}_t, z_t)$.
    (b) Otherwise (reject the candidate), set $X_t = X_{t-1}$.
    (c) Set $t \leftarrow t + 1$.
(3) $X_0$ is a draw from the posterior (5.15)!
```

5.4 Simulation Results

To test the performance of the algorithms, we used simulated data to mimic the scenario of the different models. In all cases, the predictors $\boldsymbol{x}_i$ were generated independent and identically distributed according to $\boldsymbol{x}_i \sim N(\boldsymbol{0}, \boldsymbol{I}),\ i = 1, \ldots, r$.

For each different model, we describe in detail how the test data was produced. Results are presented by listing the frequencies for the $\boldsymbol{\gamma}$-vectors as well as specifying the marginal probabilities for each component γ_i. To discuss computational cost we list the average, the minimum, and the maximum backward coupling time.

5.4.1 Fixed Variance, Fixed Coefficients

This case has been solved and tested in [31]. We include our test results for completeness. We used $q = 5$ predictors and generated $n = 20$ data records the following way: After simulating a noise vector $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I})$ of size 20 (σ^2 was set to 1), the data was computed by assigning $\boldsymbol{y} = \sum_{i=1}^{5} \gamma_i \theta_i \boldsymbol{x}_i + \boldsymbol{\varepsilon}$ with $\boldsymbol{\theta} = (0.8, 0.7, 0.7, 0.7, 0.9)'$ and $\boldsymbol{\gamma} = (1, 0, 0, 1, 0)'$.

With 1000 i.i.d. samples from the posterior (5.6), we obtained results shown in Table 5.1. The average backward coupling time for the Gibbs sampler was $T = 1.686$ with a minimum of $T = 1$ and a maximum of $T = 3$.

The results show that the algorithm could easily recover which of the predictors had been part of the model. Both lists clearly reveal the $\boldsymbol{\gamma}$-vector that has been used to generate the data $\boldsymbol{y}$.

Table 5.1: Results for 1000 i.i.d. draws from the posterior (5.6) with fixed variance and fixed coefficients.

γ	percentage
(1,0,0,1,0)	92.9 %
(1,1,0,1,0)	4.7 %
(1,0,0,1,1)	1.2 %
(0,0,0,1,0)	0.6 %
(1,0,1,1,0)	0.3 %
(0,1,0,1,0)	0.1 %
(1,1,1,1,0)	0.1 %
(1,0,0,0,1)	0.1 %

component	percentage
$P(\gamma_1 = 1)$	99.3 %
$P(\gamma_2 = 1)$	4.9 %
$P(\gamma_3 = 1)$	0.4 %
$P(\gamma_4 = 1)$	99.9 %
$P(\gamma_5 = 1)$	1.3 %

5.4.2 Random Variance, Fixed Coefficients

To test this case, we simulated a value z according to $Z \sim \Gamma(\frac{\nu}{2}, \frac{\lambda\nu}{2})$ and used $\sigma^2 = \frac{1}{z}$ as the variance for the noise vector $\boldsymbol{\varepsilon}$. We again used $q = 5$ predictors, but this time generated $n = 50$ data records. The coefficient vector of the model was $\boldsymbol{\theta} = (1.2, 1.3, 1, 1.1, 1.2)'$ and the hyperparameters were chosen to be $\lambda = 1$ and $\nu = 1$. The predictors that went into the model were determined by $\boldsymbol{\gamma} = (1, 0, 0, 1, 0)'$.

After 1000 i.i.d. draws from (5.11), we obtained the results shown in Table 5.2 (we only list realizations of $\boldsymbol{\gamma}$ that had a frequency of 1% or more). The average backward coupling time for the coupling the z-component (outer Gibbs sampler) was $T = 30.091$ with a minimum of $T = 1$ and a maximum of $T = 366$. For the inner Gibbs sampler, i.e. for the support set coupling to draw the $\boldsymbol{\gamma}$-component – the average backward coupling time was $T = 1.141$ with a minimum of $T = 1$ and a maximum of $T = 3$. In this case, the algorithm also detected the right predictors, with the second most frequent model only occurring half as often as the true one. Again, the marginal probabilities for each component γ_i clearly point to the correct $\boldsymbol{\gamma}$-vector.

Table 5.2: Results for 1000 i.i.d. draws from the posterior (5.11) with random variance and fixed coefficients.

$\boldsymbol{\gamma}$	percentage
(1,0,0,1,0)	37.8 %
(1,0,1,1,0)	18.2 %
(0,0,0,1,0)	11 %
(1,0,0,1,1)	8.9 %
(1,0,0,0,0)	5.1 %
(0,0,1,1,0)	3.5 %
(1,1,0,1,0)	3.1 %
(1,0,1,1,1)	2.3 %
(1,0,1,0,0)	1.6 %
(1,0,0,0,1)	1.3 %
(0,0,0,1,1)	1 %

component	percentage
$P(\gamma_1 = 1)$	80.9 %
$P(\gamma_2 = 1)$	6.5 %
$P(\gamma_3 = 1)$	28.7 %
$P(\gamma_4 = 1)$	88.9 %
$P(\gamma_5 = 1)$	15.7 %

5.4.3 Fixed Variance, Random Coefficients

While we ranked this case under the most general case using an IMH algorithm when specifying a perfect algorithm to sample from the posterior, we still decided to include simulation results separately. This case was tested with $q = 5$ predictors and $n = 20$ data records. We generated 20 coefficient vectors $\boldsymbol{\theta}$ according to $\boldsymbol{\Theta} \sim N(\boldsymbol{\xi}, \sigma^2 \boldsymbol{I})$ with $\boldsymbol{\xi} = (1,1,1,1,1)'$ and $\sigma^2 = 0.5$ which was also used as the variance for the noise vector $\boldsymbol{\varepsilon}$. The predictors that went into the model were determined by $\boldsymbol{\gamma} = (1,0,0,1,0)'$.

After 100 i.i.d. draws from we obtained the results shown in Table 5.3. The average backward coupling time for the IMH algorithm was $T = 429842.21$ with a minimum of $T = 8809$ and a maximum of $T = 2601550$. Note that the high backward coupling times in this model do not allow to go to much higher dimensions. It is possible that this drawback may be overcome with the idea of a *multistage coupler* which is a subject of future work. See Section 7.2.2 for more details on the multistage coupler. Despite the small number of samples, the algorithm clearly yields the correct model.

Table 5.3: Results for 100 i.i.d. draws from the posterior with fixed variance and random coefficients.

γ	percentage
(1,0,0,1,0)	59 %
(1,0,0,1,1)	28 %
(1,1,0,1,0)	6 %
(1,0,1,1,0)	4 %
(1,0,1,1,1)	2 %
(1,1,0,1,1)	1 %

component	percentage
$P(\gamma_1 = 1)$	100 %
$P(\gamma_2 = 1)$	7 %
$P(\gamma_3 = 1)$	6 %
$P(\gamma_4 = 1)$	100 %
$P(\gamma_5 = 1)$	31 %

5.4.4 Random Variance, Random Coefficients

We used $q = 3$ predictors and $n = 20$ data records. The mean vector for the coefficients was given by $\boldsymbol{\xi} = (1,1,1)'$ and the hyperparameters were chosen to be

$\lambda = 10$, $\nu = 1$, and $\boldsymbol{V} = \boldsymbol{I}$. The predictors that went into the model were determined by $\boldsymbol{\gamma} = (1, 1, 0)'$.

After 100 i.i.d. draws from (5.15), we obtained the results shown in Table 5.4. The average backward coupling time for the IMH algorithm was $T = 542.8$ with a minimum of $T = 2$ and a maximum of $T = 4257$. Due to the very large backward coupling times of the IMH algorithm in higher dimensions, we chose only 3 predictors and 20 data records in this case. Again, the multistage coupler described in Section 7.2.2 might help to reduce these coupling times. Once again, the algorithm clearly detected the correct $\boldsymbol{\gamma}$-vector.

Table 5.4: Results for 100 i.i.d. draws form the posterior (5.15) with random variance and random coefficients.

γ	percentage
(1,1,0)	34 %
(1,0,0)	18 %
(1,1,1)	18 %
(0,1,0)	13 %
(0,1,1)	9 %
(0,0,1)	3 %
(1,0,1)	3 %
(0,0,0)	2 %

component	percentage
$P(\gamma_1 = 1)$	73 %
$P(\gamma_2 = 1)$	74 %
$P(\gamma_3 = 1)$	33 %

5.5 Testing on the Hald Data Set

We now present results using the algorithm from Section 5.3.3 for the Hald data set. This data set describes the effect of the composition of cement on heat evolved during hardening. The heat evolved is important since it determines how quickly the cement dries. It is originally taken from Woods et al. [61], but has been used in many other references including Hald [27], Draper and Smith [13], and George and McCulloch [23]. A description can also be found at [28].

The Hald data consists of $n = 13$ observations on the dependent variable (heat) and $q = 4$ predictor variables which relate to the composition of the cement (tricalcium aluminate, tricalcium silicate, tetracalcium alumino, ferrite dicalcium silicate).

Mimicking the choice as hyperparameters in [23] (although the variable selection model is set up differently to our framework), we pick $\boldsymbol{V} = c\boldsymbol{I}$ (with three different values for c, $c = 1, 5, \text{and } 10$), $\boldsymbol{\xi} = (0, 0, 0, 0)^T$, $\lambda = 10000$, and $\nu = 1$.

While George and McCulloch [23] get different model results for different prior parameters, our problem formulation and/or algorithm seems to yield more stable outcome. Our results would imply that the tricalcium silicate component is directly linked to the shortest hardening time of the cement.

Table 5.5: Results for 100 draws for the Hald data set and different prior parameters.

$\boldsymbol{c = 1}$

γ	percentage
(0,1,0,0)	69 %
(1,1,0,0)	14 %
(1,0,1,0)	13 %
(0,1,1,0)	3 %
(0,1,0,1)	1 %

component	percentage
$P(\gamma_1 = 1)$	27 %
$P(\gamma_2 = 1)$	87 %
$P(\gamma_3 = 1)$	16 %
$P(\gamma_4 = 1)$	1 %

$\boldsymbol{c = 5}$

γ	percentage
(0,1,0,0)	96 %
(1,1,0,0)	1 %
(0,0,1,0)	1 %
(1,0,1,0)	1 %
(0,1,1,0)	1 %

component	percentage
$P(\gamma_1 = 1)$	2 %
$P(\gamma_2 = 1)$	98 %
$P(\gamma_3 = 1)$	3 %
$P(\gamma_4 = 1)$	0 %

$\boldsymbol{c = 10}$

γ	percentage
(0,1,0,0)	99 %
(1,0,0,0)	1 %

component	percentage
$P(\gamma_1 = 1)$	1 %
$P(\gamma_2 = 1)$	99 %
$P(\gamma_3 = 1)$	0 %
$P(\gamma_4 = 1)$	0 %

Chapter 6

Perfect Stochastic Summation in High Order Feynman Graph Expansions

I think that I can safely say that nobody understands quantum mechanics.

Richard Feynman, *The character of physical law* [14].

6.1 Introduction

The interacting fermion problem [18, 33, 44] is of fundamental importance in a wide range of research areas, including fields as diverse as electronic structure theory of solids, strongly correlated electron physics, quantum chemistry, and the theory of nuclear matter. The ultimate objective of this project, which is a massive collaborative effort between physicists, statisticians, and computer scientists, is to combine Monte Carlo summation techniques with self-consistent high-order *Feynman diagram expansions* into an efficient tool for the controlled approximate solution of interacting fermion models.

The concept of *self energy* is one of the most important examples of the power of Feynman diagram resummation. Suppose, for example, that we have a lattice of atoms (such as in the case of a crystal) where electrons are almost localized in atomic orbitals at each site. Suppose further that we create a particle at one site and destroy a particle at another site thereby adding and removing a quanta of vibrational energy to the system. Electrons of opposite spin occupying a single atom give rise to a Coulomb repulsion energy which causes particles to hop between sites. There is a contribution to the energy of the particle due to the virtual emission and absorption of particles. In other words, the particle interacts with its surrounding medium which, in turn, acts back on the particle. Essentially, a "wake" of energy is created around the movement of the particle. It is this self energy, described in more detail in Section 6.2, that we would like to quantify.

Self energy is represented as a large and complicated sum ("outer sum") of terms that are, in themselves, large and complicated sums ("inner sums"). The objective of this chapter is to describe the evaluation of these inner sums via a Monte Carlo approach. Monte Carlo summation techniques, detailed in Section 6.3, involve evaluating an approximating sum at random points drawn from a particular distribution. While there is much flexibility in the choice of this distribution, some choices are better than others in terms of minimizing the (first level) error in the approximating sum. The downside to choosing a nearly optimal distribution is that we often cannot sample points from it directly and are faced with a second level of error (sampling error) resulting from a Monte Carlo sampling approach. We choose a nearly optimal complicated distribution, yet sample from it using perfect simulation techniques (more precisely, the IMH algorithm [1]) to completely eliminate the second level error.

This chapter is organized as follows. In Section 6.2, we formulate and state the problem, describe to a certain extent the physical quantities that are involved, and discuss computer-friendly representation of the information needed from Feynman diagrams. Section 6.3 gives a general introduction to Monte Carlo techniques.

We show the core of our work in Section 6.4. We present the groundwork to apply Monte Carlo methods for computing self-energy in Section 6.4.1, while Section 6.4.2

[1] As in the previous chapter, by "IMH algorithm" we again refer to the perfect version from Section 3.3.1.

concerns using the IMH algorithm for this problem. We depict how to approximate a required proportionality constant in Section 6.4.3, and weaken a previously made restriction of one of the parameter spaces in Section 6.4.4.

Simulation results are given in Section 6.5 and Section 6.6. In the latter section we discuss the idea of *conditional averaging* (Section 6.6.1) and the bounded IMH algorithm (Section 6.6.2) from Section 3.3.2 to address some computational issues with the Monte Carlo sum.

6.2 Self Energy, Feynman Diagrams, and Computer-Friendly Representation

The Hubbard model was originally developed in the early sixties in order to understand the magnetic properties of electron motion in transition metals. The original model remains a subject of active research today, and it is within this context that we describe our numerical approach. (The approach applies, with minor adaptations, to all interacting fermion models which possess a Feynman diagram expansion in terms of a two-body interaction potential.)

Suppose we have a lattice of atoms, representing atoms in a crystal, where electrons are almost localized in atomic orbitals at each site (see Figure 6.1). We consider the following Hamiltonian model governing the motion and interactions between the particles.

$$H = \sum_{j',j} \frac{1}{2} V_{j',j}\, n_{j'}\, n_j - \sum_{j',j} \sum_{\sigma} t_{j',j}\, c^{\dagger}_{j'\sigma} c_{j\,\sigma} \tag{6.1}$$

Here

$c^{\dagger}_{j'\sigma}$	=	creation operator that creates an electron of spin $\sigma = \uparrow, \downarrow$ at site j'
$c_{j\,\sigma}$	=	destruction operator that destroys an electron of spin $\sigma = \uparrow, \downarrow$ at site j
$c^{\dagger}_{j'\sigma}\, c_{j\,\sigma}$	=	transfers an electron of spin σ from site j to site j'
n_j	=	the number of electrons at site j.

$V_{j'j}$ is called the model *Coulomb potential* and it includes the on-site ($j' = j$) Coulomb repulsion energy (U) generated when two electrons of opposite spin occupy a single atom as well as possibly extended (1st, 2nd, ... neighbor) repulsions.

$t_{j'j}$ is a parameter that controls an electron's ability to tunnel or "hop" between sites. If orbitals are highly localized, then this tunneling amplitude will decay exponentially with distance between sites. $t_{j'j}$ includes the on-site ($j' = j$) energy of the electron, a nearest neighbor tunneling amplitude (t) and possibly extended (2nd, 3rd ... neighbor) tunneling amplitudes.

The total energy, given by (6.1), of this Hubbard model is thus represented as a sum of *potential* and *kinetic* energy given by the first and second terms, respectively, of (6.1). Both $V_{j'j}$ and $t_{j'j}$ are assumed to obey period boundary conditions on a finite lattice defined in Section 6.2.1.

6.2.1 Self Energy and Feynman Diagrams

The concept of self energy enables us to understand the feedback of the interacting environment on a propagating particle. Physically, it describes the cloud of particle-hole excitations that form the wake which accompanies a propagating electron.

The following paragraphs describe how to obtain the self energy $\sigma(k)$. Basically, we will depict $\sigma(k)$ through Feynman graph expansions and represent it as is a sum (of sums) involving Green's functions. (These Green's functions will in turn involve the self energy $\sigma(k)$ we are trying to compute – how to deal with this issue is addressed in Section 6.4.) We also refer the reader to Table 6.1 which contains a summary of the

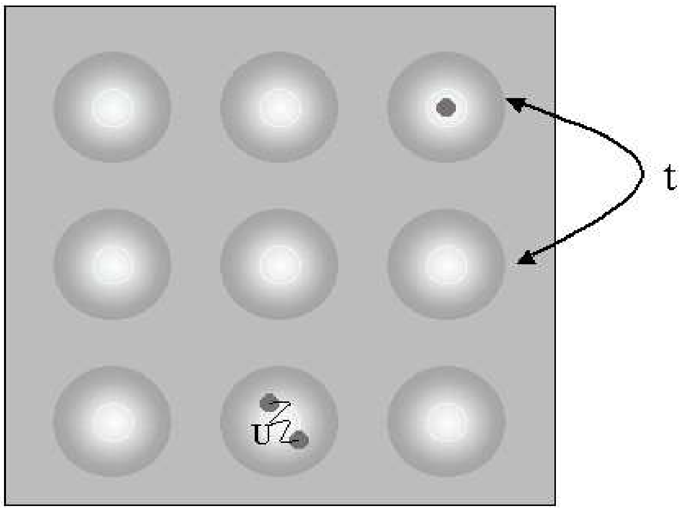


Illustration of the Hubbard model. The amplitude for an electron to hop from site to neighboring site is t and the Coulomb repulsion energy generated by two electrons of opposite spins at the same site is U.

Figure 6.1: The Hubbard model

symbols used in this section.

Green's function $G(k)$

The single-fermion Green's function $G(k)$ is the most basic physical quantity which can be obtained via a Feynman graph expansion. As it is expressed in terms of self energy, which we will denote by $\sigma(k)$, we will describe our approach within this context. Here $k \equiv (\vec{k}, i\nu)$ denotes a $D + 1$ - dimensional "momentum-energy" variable (referred to as "momentum" hereafter), where $\vec{k}$ is a D-dimensional "wavevector" and $i\nu$ represents a frequency. These variables will be described in more detail later.

In general, a Green's function is the response of a linear system to a point input. Diagrammatically, we can represent the single-fermion Green's function, or *propagator*, as a sum of *free propagators* G_0 with various inserted scattering processes. (Mathematically, a propagator is simply an integral kernel.) The self energy represents all scattering processes combined into a single quantity. We depict the Green's function in terms of the self energy in Figure 6.2.

The self energy is in turn represented as a sum of *Feynman diagrams* as shown in Figure 6.3. These diagrams, developed by physicist Richard Feynman, give a convenient shorthand for representing complex mathematical quantities. The "wavy" lines represent photons that are emitted or absorbed by an electron traveling along the straight lines. Assignments of values to the momentum are made on various sections of the diagrams which correspond to arguments of functions in the algebraic self energy expression.

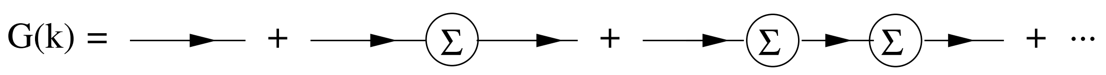


Figure 6.2: Representing a Green's function as a sum of free propagators with inserted scattering processes

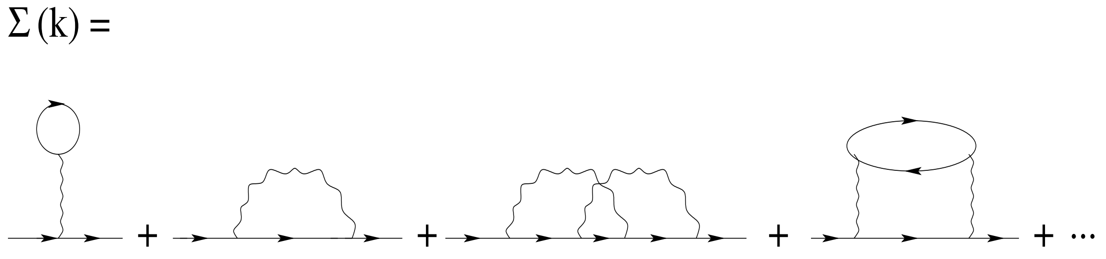


Figure 6.3: Representing self energy as a sum of Feynman diagrams

Algebraically, the free propagator has the form

$$G_0(k) = G_0(\vec{k}, i\,\nu) = \frac{1}{i\,\nu - \varepsilon(\vec{k})}$$

where $\varepsilon(\vec{k})$ is the *electron energy band*

$$\varepsilon(\vec{k}) = -\mu - 2t \sum_{d=1}^{D} \cos(k^{(d)}),$$

and D is the lattice dimension. Here, μ and t are, respectively, the on-site electron energy (chemical potential) and nearest neighbor tunneling amplitude, discussed above in the description of the Hubbard model, and $k^{(j)}$ is the j^{th} element of $\vec{k}$. $\varepsilon(\vec{k})$ is simply the Fourier transform of $t_{j'j}$ in the case that only 1st neighbor tunneling and on-site energy are included in $t_{j'j}$.

Following the Green's function representation from Figure 6.2, we may write

$$\begin{aligned} G(k) &= G_0 + G_0\sigma G_0 + G_0\sigma G_0\sigma G_0 + \cdots \\ &= G_0 + G_0\sigma G_0 + G_0(\sigma G_0)^2 + \cdots \\ &= \frac{1}{G_0^{-1} - \sigma} \end{aligned}$$

by summing the Taylor series expansion in powers of (σG_0).

This is known as the *Dyson equation* [33, 44] and can be written as

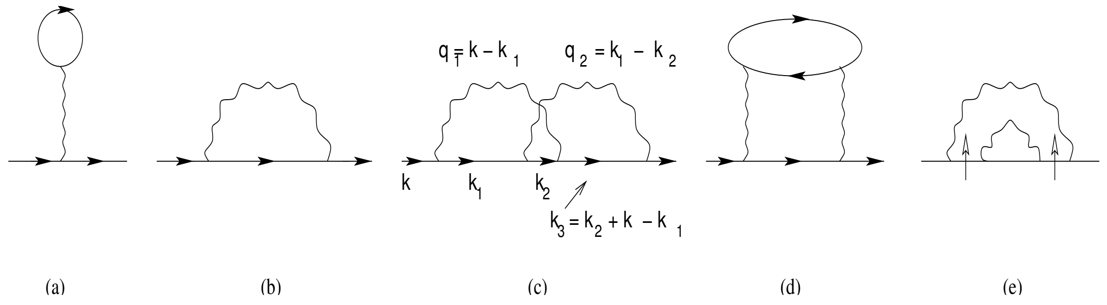
$$G(k) = G(\vec{k}, i\,\nu) = [i\,\nu - \varepsilon(\vec{k}) - \sigma(k)]^{-1}. \tag{6.2}$$

Self energy

The self energy $\sigma(k)$ is obtained self-consistently via a Feynman graph expansion in terms of G and the interaction potential V. As illustrated in Figure 6.4 for orders $n = 1$ and 2, an n^{th} *order σ-graph* consists of n non-directional wavy lines (referred to as "V-lines" hereafter) and of 2 external and $2n-1$ internal directed straight lines, (referred to as "G-lines" hereafter). One incoming and one outgoing G-line is attached to each endpoint ("vertex") of each V-line. The $\sigma(k)$-contribution for each graph is given by the Feynman rules [33, 44] so that for $k \equiv (\vec{k}, i\,\nu)$ and $k_v \equiv (\vec{k}_v, i\,\nu_v)$ $(v = 1, 2, 3, \ldots)$,

$$\sigma(k) = \sum_{n=1}^{n_{max}} \sum_{g \in \mathcal{G}_n} \left(\frac{-T}{N}\right)^n \sum_{k_1,\ldots,k_n \in \mathcal{K}} F_g^{(n)}(k, k_1, \ldots, k_n). \tag{6.3}$$

Here, $\mathcal{G}_n$ denotes the set of all topologically distinct "G-irreducible" σ-graphs g of order n, where g is defined to be G-irreducible (G-reducible) if and only if it cannot (can) be severed into two disjoint pieces by cutting no more than two internal G-lines, as illustrated in Figure 6.4.



Shown are all G-irreducible σ-graphs of order $n = 1$, in (a) and (b), and of order $n = 2$, in (c) and (d), as well as a selected G-reducible graph of order $n = 2$, in (e). In (c), a possible k-assignment is also shown. In (e), vertical arrows indicate the "cuts" which separate the graph into two disjoint pieces.

Figure 6.4: Feynman Diagrams

Summation domains

The $\vec{k}_v$-summation domain is the set $\mathcal{B}$ of $\vec{k}$- grid points in the *first Brillouin zone*:

$$\mathcal{B} := \{\vec{k} = (k^{(1)}, \ldots, k^{(D)}) \,|\, k^{(d)} = \frac{2\pi m_d}{L_d}, m_d \in \mathcal{L}_d \text{ for } d \in 1, \ldots, D\} \tag{6.4}$$

with

$$\mathcal{L}_d := \{-\lfloor (L_d - 1)/2 \rfloor, -\lfloor (L_d - 1)/2 \rfloor + 1, \ldots, \lfloor L_d/2 \rfloor\},$$

where $\lfloor \cdot \rfloor$ is the greatest integer function and L_d denotes the integer sidelength of the D-dimensional finite lattice prism in the d^{th} coordinate direction. N is the total number of sites j in the lattice, hence

$$N = \prod_{d=1}^{D} L_d.$$

The $i\,\nu$-summation domain is the set $\mathcal{M}$ of *odd Matsubara frequencies*:

$$\mathcal{M} := \{i\,\nu \,|\, \nu = (2m_0 + 1)\pi T\,,\ m_0 \text{ an integer}\} \tag{6.5}$$

where T denotes the temperature of the physical system.

The inner sums in (6.3) are therefore taken over $\mathcal{K} := \mathcal{B} \times \mathcal{M}$, and the summand $F_g^{(n)}(k, k_1, \ldots, k_n)$, for $n \geq 1$, contains the internal G- and V-line factors of graph g:

$$F_g^{(n)}(k, k_1, \ldots, k_n) = (-(2s_f + 1))^{l g} \exp(\delta_{n,1}\, i\,\nu\, 0^+) \prod_{u=1}^{2n-1} G(k_u) \times \prod_{x=1}^{n} V(q_x). \tag{6.6}$$

Here, the momenta k_u and q_x associated with the G- and V-lines, respectively, are determined by the graph's topology, via momentum conservation rules at each vertex, as illustrated in Figure 6.4(c). Only the first n of the internal G-line k-variables, $k_1, \ldots, k_n$ can be chosen and summed over independently; the remaining k-variables, $k_{n+1}, \ldots, k_{2n-1}$, and all q-variables, $q_1, \ldots, q_n$, are linear combinations of the external k and of $k_1, \ldots, k_n$.

Finally, $V(q)$ denotes the Fourier transform of the interaction potential V of our lattice model:

$$V(q) = V(\vec{q}) := N^{-1} \sum_{j'\,j} e^{-i\vec{q}\cdot(\vec{r}_{j'} - \vec{r}_j)} V_{j'\,j} \tag{6.7}$$

where $\vec{r}_j$ denotes the position vector of site j. Note that q is a momentum variable with a wavevector component in $\mathcal{B}$ and a frequency component ($q = (\vec{q}, i\,\nu)$). The frequency component is not in the set of odd Matsubara frequencies (i.e. $q \notin \mathcal{K}$), however as $V(q)$ is independent of the frequency component, we will not describe the domain here. Remaining parameters and are described in Table 6.1 where all parameters and quantities addressed thus far are also summarized.

The purpose of this chapter is to give a "perfect" Monte Carlo approach that will perform the **innermost** summation in (6.3). Our collaborators address the other summations in [58].

6.2.2 Computer-Friendly Representation

Computationally, each Feynman graph $g \in \mathcal{G}_n$ in (6.3) can be conveniently represented by labeling each vertex by an integer $n \in \{1, 2, \ldots, 2n\}$, in such a manner that v and $v + 1$ denote endpoints of the same V-line if v is odd. Additionally, the origin of the incoming external G-line and the terminus of the outgoing external G-line are labeled $v = 0$ and $v = 2n + 1$, respectively. The complete topology (connectivity) of the graph can then be specified by a "linkage list", $[w(v)]_{v=0}^{2n}$, where $w(v)$ is the integer label of the terminating vertex of the G-line that originates at vertex v. The integer v then also serves as a convenient label of all G-lines: "v" is that G-line which originates from vertex v for $v \in \{1, 2, \ldots 2n\}$, with $v = 0$ labeling the incoming external G-line. Using the notation $[w^{-1}(v)]_{v=1}^{2n+1}$ for the inverse assignment, an example for a graph of order $n = 2$ is shown in Figure 6.5.

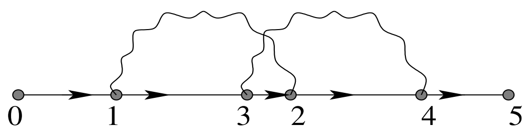
The assignment of k and q variables corresponding to the internal G- and V-lines in (6.6) can be represented by a pair of "k-assignment" lists, $\sigma_G(u, v)$ and $\sigma_V(x, v)$, such that

$$k_u = \sum_{v=0}^{n} \sigma_G(u, v)\, k_v \qquad \text{and} \qquad q_x = \sum_{v=0}^{n} \sigma_V(x, v)\, k_v \tag{6.8}$$

for $u \in \{1, 2, \ldots, 2n - 1\}$ and $x \in \{1, 2, \ldots, n\}$ with $k_0 \equiv k$. Given the graph's linkage list (and its inverse w^{-1}), the σ_G's and σ_V's are constructed to satisfy momentum

Table 6.1: Summary of symbols

notation	meaning	add. information
D	number of dimensions of the lattice	
L_d	integer sidelength of the lattice in dimension d, $(d = 1, 2, \ldots, D)$	
$\mathcal{B}$	set of points in the 1st Brillouin zone	given by (6.4)
N	number of points in $\mathcal{B}$	
$\mathcal{M}$	set of odd Matsubara frequencies	given by (6.5)
$k \equiv (\vec{k}, i\,\nu) \in \mathcal{K} = \mathcal{B} \times \mathcal{M}$	momentum-energy variable (momentum)	consists of a wavevector $\vec{k} \in \mathcal{B}$ and a frequency $i\,\nu \in \mathcal{M}$; associated with "G-lines" (straight lines) in the Feynman diagrams
$k_1, k_2, \ldots$	elements of $\mathcal{K}$	
$k^{(1)}, k^{(2)}, \ldots, k^{(D)}$	components of a vector $\vec{k} \in \mathcal{B}$	
$\sigma(k)$	self energy	given by (6.3)
$G(k)$	Green's function	given by (6.2)
$q \equiv (\vec{q}, i\,\nu)$, $\vec{q} \in \mathcal{B}$	momentum-energy variable (momentum)	consists of a wavevector $\vec{q} \in \mathcal{B}$ and a frequency $i\,\nu$; associated with "V-lines"(wavy lines) in the Feynman diagrams
$V_{j'j}$	model interaction potential	repulsion energy generated by electrons
$V(\vec{q})$	Fourier transform of interaction potential	given by (6.7)
l^g	number of closed G-loops in graph g	Figure 6.4, $l^g = 1, 0, 0, 1, 0$ for (a), (b), (c), (d), and (e), respectively
s_f	single-fermion spin quantum number	$s_f = \frac{1}{2}$ for non-spin-polarized electrons
δ_{ij}	Kronecker delta	1 when $i = j$, zero otherwise
n_{max}	the highest order graphs that will be considered in computing $\sigma(k)$	
$F_g^{(n)}(k, k_1, \ldots, k_n)$	the summand of $\sigma(k)$	given by (6.6)



v	w(v)	$w^{-1}(v)$
0	1	–
1	3	0
2	4	3
3	2	1
4	5	2
5	–	4

Figure 6.5: Representing a Feynman diagram as a linkage list.

conservation at each vertex. For the endpoints of V-line $x \in \{1, 2, \ldots, n\}$, that is, for vertices $v = 2x - 1$ and $v = 2x$, respectively,

$$k_{2x-1} + q_x = k_{w^{-1}(2x-1)} \qquad \text{and} \qquad k_{2x} = q_x + k_{w^{-1}(2x)}. \tag{6.9}$$

We have adopted the convention that the (usually non-directional) V-line x carries momentum q_x **from** vertex $2x - 1$ **towards** vertex $2x$ and $k_0 = k_{2n} \equiv k$. The resulting $\sigma_G(u, v)$ and $\sigma_V(x, v)$ take on values 0, +1, and −1 only, with $\sigma_G(u, v) = \delta_{u,v}$ for $u = 0, 1, \ldots, n$, as illustrated in Figure 6.4(c).

Preliminary steps for this labeling procedure consists of setting up tables with topologies of all irreducible diagrams. This task has been carried out using a depth first search algorithm program developed by Robert Robinson and coworkers (Robinson, private communication).

With this program, we are able to compute the number of irreducible diagrams for any given order. Table 6.2 gives these numbers up to order 30. Ultimately, we include in our calculations all diagrams up to some prespecified and fixed order n_{max}. As the contributions of diagrams to ((6.3) become exponentially small with increasing order, the contributions of very high order diagrams will become negligible when compared to the statistical error of the Monte Carlo sum.

Table 6.2: Number of irreducible Feynman diagrams of order n, $n = 1, 2, \ldots, 30$

n	Number of Irreducible Graphs of Order n
1	2
2	2
3	10
4	82
5	898
6	12018
7	187626
8	3323682
9	65607682
10	1424967394
11	33736908874
12	864372576626
13	23825543471234
14	703074672632018
15	22118247888976170
16	739081808704195650
17	26146116129400483842
18	976382058777174451650
19	38386296866727499728522
20	1584986693941237056394386
21	68581486271828442449029506
22	3103400608176999567390207666
23	146590637383801346734328051562
24	7215429954884025583945405736802
25	369497732954045370736682490044418
26	19656721052188233635666361136506018
27	1084829370056105520244895442117978826
28	62030456805440999734369445536191672754
29	3670462157003477448177381929142409410178
30	224502553026493195585402880052982012256402

6.3 Monte Carlo Summation

We are often faced with integrals (or sums) that cannot be computed analytically. In the case of multidimensional integrals, even the simplest methods of approximation by discretization can become prohibitively expensive. Monte Carlo integration provides an alternative approximation technique.

Suppose we wish to evaluate the integral

$$I(g, A) = \int_A g(x)dx \tag{6.10}$$

for some integrable function $g : \mathbb{R}^n \to \mathbb{R}$ and some set $A \subseteq \mathbb{R}^n$. Let $\pi(x)$ be any probability density function whose support contains A. Then we may write

$$I(g, A) = \int_A g(x)dx = \int_{\mathbb{R}^n} g(x)\mathbb{1}_A(x)dx = \int_{\mathbb{R}^n} \frac{g(x)\mathbb{1}_A(x)}{\pi(x)}\pi(x)dx$$

where $\mathbb{1}_A(x)$ is the indicator function. We shall refer to $\pi(x)$ as the *weight function* and the remaining part of the integrand as the *score function*.

Now if X is a random variable with density $\pi(x)$, we find that we have written the integral as an expected value

$$I(g, A) = \mathsf{E}\left[\frac{g(X)\mathbb{1}_A(X)}{\pi(X)}\right]. \tag{6.11}$$

A basic Monte Carlo integration technique is to simulate independent and identically distributed (i.i.d.) values drawn from the distribution with density $\pi(x)$, say

$$X_1, X_2, \ldots, X_n \overset{iid}{\sim} \pi$$

and to estimate the probability weighted average given in (6.11) by

$$\hat{I}(g, A) = \frac{1}{n}\sum_{i=1}^{n} \frac{g(X_i)\mathbb{1}_A(X_i)}{\pi(X_i)}. \tag{6.12}$$

Clearly, by (6.11), this is an unbiased estimator of $I(g, A)$.

Since the variance of this estimator is

$$\begin{aligned} V[\hat{I}(g,A)] &= \frac{1}{n} V\left[\frac{g(X)\mathbb{1}_A(X)}{\pi(X)}\right] \\ &= \frac{1}{n}\mathsf{E}\left[\left(\frac{g(X)\mathbb{1}_A(X)}{\pi(X)} - I(g,A)\right)^2\right], \end{aligned}$$

the variance is minimized by choosing

$$\pi(x) = \frac{|g(x)|\mathbb{1}_A(x)}{\int_{\mathbb{R}^n} |g(x)|\mathbb{1}_A(x)\,dx}. \tag{6.13}$$

Specifically, we choose

$$\pi(x) = \begin{cases} \frac{|g(x)|}{\int_A |g(x)|\,dx} & x \in A \\ \\ 0 & x \notin A. \end{cases} \tag{6.14}$$

Of course, if we could compute the denominator for this optimal "weight", it is likely that we could have solved our original problem and there is no need to use a Monte Carlo approach to compute the integral in (6.10). Hence, we may consider taking a different, non-optimal candidate weight, keeping in mind that we would like to at least choose something that attempts to mimic the shape of g. We point out that while we may usually sample from (6.14) without knowing the constant of proportionality, (for example using the Metropolis-Hastings algorithm), we ultimately require the constant in order to evaluate (6.12).

6.4 Perfect IMH in the Context of this Self Energy Problem

We now describe the IMH algorithm for computing the innermost sum in (6.3) for the second order diagram depicted in Figure 6.4(c) and Figure 6.5. Ignoring the physical constants, we would like to compute

$$\sigma_2(k) := \sum_{k_1,k_2\in\mathcal{K}} \prod_{u=1}^{3} G(k_u) \times \prod_{x=1}^{2} V(q_x). \tag{6.15}$$

Using momentum conserving assignments, described in Section 6.2.2 and shown in Figure 6.4(c), this becomes

$$\sigma_2(k) = \sum_{k_1,k_2\in\mathcal{K}} G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)\cdot V(k-k_1)\cdot V(k_1-k_2). \qquad (6.16)$$

Recall that each k, k_1, k_2 is a $D+1$-dimensional with the vector of the first D components coming from $\mathcal{B}$ as described in (6.4) and the third component taking values in

$$\mathcal{M} := \{i\,\nu\,|\,\nu = (2m_0+1)\pi T\,,\ m_0 \text{ an integer}\}.$$

As a computational simplification, we will truncate $\mathcal{M}$ to a finite space $\mathcal{M}_T$. (We address alternatives in Section 6.4.4.) This will allow us to choose a trivial candidate distribution for the IMH algorithm where, more importantly, we can easily maximize the h/q ratio described in (3.3) in Section 3.3.1 in order to identify the lowest state l needed to carry out the algorithm. For physical reasons, it is necessary to truncate $\mathcal{M}$ in a symmetric way, say

$$\mathcal{M}_T = \{i\,\nu\,|\,\nu = (2m_0+1)\pi T\,,\ m_0 \in \{-l, -l+1, \ldots, l-1\}\}$$

for some positive integer l.

The astute reader may have noticed by now that we are trying to compute $\sigma_2(k)$ which is a term of $\sigma(k)$, given by (6.3), which is itself involved in the Green's function (6.2), which is in our expression for $\sigma_2(k)$. This is handled with standard feedback methods described in [58], [45], and [41] where one

(1) starts by setting $\sigma(k) \equiv 0$ in (6.2) so that $G(k)$ becomes

$$G(k) = G(\vec{k}, i\,\nu) = [i\,\nu - \varepsilon(\vec{k})]^{-1}, \qquad (6.17)$$

(2) carries out the entire Monte Carlo procedure to compute (6.3) which consists of the procedures of this paper for the "inner (single-diagram) sum" and the procedures of [58] for the "outer sums" which move through diagram orders and diagrams within an order,

(3) uses the resulting $\sigma(k)$ to update the Green's function before returning to step (2) and repeating the entire combined Monte Carlo procedure.

This approach is repeated until consecutive updates of the Green's function coincide within a given limit of accuracy. As this paper is concerned with computing the self energy contribution for single diagrams only, we will use a fixed Green's function. For simplicity, we use (6.17).

6.4.1 The Monte Carlo Sum

For the Monte Carlo summation, we propose as a weight function the product of the magnitudes of the Green's functions

$$|G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k - k_1)|. \tag{6.18}$$

However, since this involves the input external momentum k, it would be inefficient to focus exclusively on this weight. In order to have a single weight function for any input momentum, we again rewrite our summation goal from (6.16) to

$$\sigma_2(k) = \sum_{k_1,k_2,k_0 \in \mathcal{K}} V(k - k_1) \cdot V(k_1 - k_2) \cdot \delta_{k,k_0} \cdot G(k_1) \cdot G(k_2) \cdot G(k_2 + k_0 - k_1), \tag{6.19}$$

where δ_{k,k_0} is equal to 1 when $k_0 = k$ and equal to 0 otherwise.

Now we may write

$$\sigma_2(k) = \sum_{k_1,k_2,k_0 \in \mathcal{K}} S(k, k_0, k_1, k_2)\, W(k_0, k_1, k_2), \tag{6.20}$$

where $W(k_0, k_1, k_2)$ is the (unnormalized) weight function

$$W(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)| \tag{6.21}$$

and $S(k, k_0, k_1, k_2)$ is the score function

$$S(k, k_0, k_1, k_2) = \delta_{k,k_0}\, V(k - k_1) \cdot V(k_1 - k_2) \frac{G(k_1) \cdot G(k_2) \cdot G(k_2 + k_0 - k_1)}{|G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|}. \tag{6.22}$$

Our "target distribution" from which to draw values for the Monte Carlo summation described in Section 6.3 is

$$\pi(k_0, k_1, k_2) = \frac{1}{N_W} W(k_0, k_1, k_2) = \frac{1}{N_W} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|, \quad (6.23)$$

where

$$N_W := \sum_{k_0, k_1, k_2 \in \mathcal{B} \times \mathcal{M}_T} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|. \quad (6.24)$$

Following Section 6.3 we write

$$\begin{aligned} \sigma_2(k) &= N_W \textstyle\sum_{k_0, k_1, k_2 \in \mathcal{B} \times \mathcal{M}_T} \; S(k, k_0, k_1, k_2)\, \pi(k_0, k_1, k_2) \\ &= N_W \; \mathsf{E}\left[S(K_0, K_1, K_2)\right], \end{aligned}$$

where (K_0, K_1, K_2) is a random vector with density $\pi(k_0, k_1, k_2)$.

6.4.2 The IMH Algorithm

We will now describe how to use the IMH algorithm from Section 3.3.1 to draw n values, (k_{0i}, k_{1i}, k_{2i}), from $\pi(k_0, k_1, k_2)$. We will use these values both to estimate N_W and to plug them into the Monte Carlo approximation

$$\hat{\sigma}_2(k) = N_W \, \frac{1}{n} \sum_{i=1}^{n} S(k, k_{0i}, k_{1i}, k_{2i}). \quad (6.25)$$

(Note: Due to the large number of symbols in use, we are reclaiming the use of the letter n throughout Section 6.4 where we are considering a fixed diagram of order 2. In previous sections, n was reserved for a diagram order.)

The unnormalized target density h is

$$h(k_0, k_1, k_2) \equiv W(k_0, k_1, k_2)$$

which is given by (6.21).

We choose a simple uniform candidate distribution with density $q(k_0, k_1, k_2)$ which gives equal weight to all points $(k_0, k_1, k_2) \in (\mathcal{B} \times \mathcal{M}_T)^3$.

In order to implement the IMH algorithm, it remains to identify the "lowest point", (k_0^*, k_1^*, k_2^*), which is the point in $(\mathcal{B} \times \mathcal{M}_T)^3$ that maximizes the ratio h/q. Since q is constant for all $(k_0, k_1, k_2) \in (\mathcal{B} \times \mathcal{M}_T)^3$, we simply want to maximize

$$h(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|. \tag{6.26}$$

Consider first a single factor $|G(k)|$. Since

$$\begin{aligned} |G(k)| &= |G(k^{(1)}, \ldots, k^{(D)}, i\nu)| = |i\,\nu + \mu + 2t \sum_{d=1}^{D} cos(k^{(d)})|^{-1} \\ &= \left[\nu^2 + \left(\mu + 2t \sum_{d=1}^{D} cos(k^{(d)}) \right)^2 \right]^{-1/2}, \end{aligned}$$

we can maximize $|G(k)|$ by minimizing

$$\nu^2 + \left(\mu + 2t \sum_{d=1}^{D} cos(k^{(d)}) \right)^2 .$$

Regardless of where we truncate $\mathcal{M}$, ν^2 is minimized for $\nu = \pm \pi T$. (The IMH algorithm does not require a unique "lowest point", so we may take either value for ν.)

We minimize

$$\left(\mu + 2t \sum_{d=1}^{D} cos(k^{(d)}) \right)^2$$

with brute force by searching over the (finite) space $\mathcal{B}$. It turns out that we only need to do this once for the entire self energy calculation. This does not depend on the particular graph we are considering at any moment and also therefore, just as importantly, does not depend on the number of k's associated with a graph.

At this point we have identified a $k^* = (\vec{k}^*, i\,\pi T)$ that maximizes $|G(k)|$. We may simultaneously maximize all factors in (6.26) by setting each of k_0, k_1, and k_2 to be k^*. Due to the relationships given by (6.9), we will always be able to simultaneously maximize all factors in the weight function associated with any Feynman graph in this way.

(Note: As $\sigma_2(k)$ and ultimately $\sigma(k)$ is complex-valued, the updated Green's function obtained through the feedback procedure described in Section 6.4 will have an imaginary component that is not simply in the set $\mathcal{M}_T$. In this case, we will have to maximize $|G(k)|$ by searching over values for $\vec{k}$ and ν together. This will not necessarily involve searching over the entire space $(\mathcal{B} \times \mathcal{M}_T)^3$ but rather it can be done with some finite upper-limit cut-off on $|i\,\nu|$ since for very large $i\,\nu$, $\sigma(k)$ is bounded by a constant independent of $i\,\nu$. This fact will be especially useful when we want to use the entire set $\mathcal{M}$ (described in Section 6.4.4) as opposed to the truncated set $\mathcal{M}_T$.)

6.4.3 Estimating N_W

In order to estimate the normalization constant N_W, we take advantage of computations already performed within the IMH algorithm. In executing the algorithm, we are drawing values uniformly from the space $(\mathcal{B} \times \mathcal{M}_T)^3$. Hence, letting $M := \#\left((\mathcal{B} \times \mathcal{M}_T)^3\right)$ denote the number of points in the space, we write N_W as

$$\begin{aligned} N_W &= \sum_{(\mathcal{B}\times\mathcal{M}_T)^3} h(k_0, k_1, k_2) \\ &= \sum_{(\mathcal{B}\times\mathcal{M}_T)^3} \frac{h(k_0, k_1, k_2)}{1/M} \frac{1}{M} \\ &= \mathsf{E}\left[\frac{h(K_0, K_1, K_2)}{1/M}\right] = M \cdot \mathsf{E}\left[h(K_0, K_1, K_2)\right], \end{aligned}$$

where (K_0, K_1, K_2) is a random vector with uniform density on $(\mathcal{B} \times \mathcal{M}_T)^3$.

We estimate this expectation by drawing n points uniformly over $(\mathcal{B} \times \mathcal{M}_T)^3$ and computing

$$\hat{N}_W = \frac{M}{n} \sum_{i=1}^{n} h(k_{0i}, k_{1i}, k_{2i})$$

where (k_{0i}, k_{1i}, k_{2i}) are the uniform draws.

6.4.4 Extending the $\mathcal{M}$-Space

In order to execute the IMH algorithm, we need to be able to identify the point that maximizes h/q where h is the (possibly unnormalized) target density and q is a candidate density with the same support as h from which we are able to simulate values. The truncation of $\mathcal{M}$ is a common practice that simplifies computational cost. This turned out to be very convenient for us as well since it provided us with a finite state space that allowed us to choose a very simple uniform candidate density which exchanged our task of maximizing h/q into the simpler task of maximizing only h. Another nice feature of this model is that we are able to make $\mathcal{M}_T$ arbitrarily large (but finite) without any adjustment, as h is always maximized at $\nu = \pm i\,\pi T$.

It is possible that one can use the full space $\mathcal{M}$ provided one chooses a candidate density q on $\mathcal{B} \times \mathcal{M}$ for which the maximizing point for h/q can be identified. If this is not the case, one alternative is to approximate the maximum either with traditional deterministic optimizers or a stochastic search approach. As illustrated in Section 3.4.1, using this "imperfect perfect simulation algorithm" may still by superior to traditional Monte Carlo methods in terms of both accuracy and speed.

6.5 Simulation Results

For each of the following two examples, we use the specific second order diagram of Section 6.4. To verify our algorithm, we start with a very low dimensional toy example where our results can be compared to those from brute force calculations.

For both examples we take $\mu = 0.5$, $t = 1.0$, and the physical temperature to be $T = 2.0$.

Example 1

We take $D = 2$, $L_1 = L_2 = 2$, and only two odd Matsubara frequencies. This means that all of the k's in (6.16) have 3 components with the first two taking values in

$$\mathcal{A} := \{0, \pi\}$$

and the third taking values in

$$\mathcal{M}_T := \{-i\,\pi T, i\pi, T\}.$$

Note that $\mathcal{B} = \mathcal{A} \times \mathcal{A}$.

We simulated 100,000 (9-dimensional) values from

$$\pi(k_0, k_1, k_2) \propto h(k_0, k_1, k_2) = |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k_0 - k_1)|.$$

The results were as follows.

(1) Exact N_W Versus the Estimate

By brute force summation, we find that the exact value of the normalizing constant N_W is 1.349741. Since there are $(2 \cdot 2 \cdot 2)^3 = 512$ possible values for (k_0, k_1, k_2), we write

$$\begin{aligned} N_W &= \sum_{(k_0,k_1,k_2)} h(k_0, k_1, k_2) \\ &= 512 \cdot \sum_{(k_0,k_1,k_2)} h(k_0, k_1, k_2) \cdot \frac{1}{512} \\ &= 512 \cdot \mathsf{E}[h(K_0, K_1, K_2)], \end{aligned}$$

where (K_0, K_1, K_2) is a random vector distributed uniformly over $(\mathcal{B} \times \mathcal{M}_T)^3$. Our estimate is then

$$\hat{N}_W = 512 \cdot \frac{1}{m} \sum_{i=1}^{m} h(k_{0i}, k_{1i}, k_{2i})$$

where (k_{0i}, k_{1i}, k_{2i}) is a particular value generated from the uniform candidate distribution in our simulation and m is the number of candidates generated. (To generate 100,000 outcomes we generate more than 100,000 candidates.)

Our simulation produced the estimate

$$\hat{N}_W = 1.349738,$$

i.e. an absolute error of $3 \cdot 10^{-6}$ and a relative error of $2 \cdot 10^{-6}$

(2) Comparing Probability Estimates for Various Regions

Using the true value of N_W, we computed the 512 probabilities associated with the 512 points in $(\mathcal{B} \times \mathcal{M}_T)^3$. We can then compute, for example

$$P(k^1_{(1)} = 0) = \frac{1}{N_W} \sum_{(k_0,k_1,k_2),k_1^{(1)}=0} h(k_0, k_1, k_2) = 0.496334.$$

In Table 6.3, we compare the true probabilities of seeing value in a given region of points with the our estimated probabilities. Although we are sampling "perfectly" from the target distribution, we are estimating probabilities based on a finite number of sampled values. All estimates consistently remain well within two standard errors of the target values.

Table 6.3: Selected true versus estimated probabilities based on a sample of size 100,000.

region	true p	est. p	abs. err.	rel. err.
$k_1^{(1)} = 0$	0.4944	0.4989	0.0044	0.0090
$k_1 = 0, \nu_2 = \pi T$	0.2472	0.2480	0.0008	0.0032
$k_0^{(1)} = 0$ or $k_1^{(2)} = 0$	0.7472	0.7471	0.0001	0.0002
50 randomly generated points	0.0963	0.0963	0.0001	0.0006
70 randomly generated points	0.1325	0.13167	0.0009	0.0065
100 randomly generated points	0.1996	0.2001	0.0004	0.0023
300 randomly generated points	0.5827	0.5828	0.0000	0.0000
375 randomly generated points	0.7397	0.7387	0.0010	0.0014
400 randomly generated points	0.7885	0.7876	0.0010	0.0012

(3) Backward Coupling Times

In 100,000 draws, the average backward coupling time (the number of time steps required to achieve a draw from the target distribution) was 1.2 time steps. The minimum and maximum backward coupling times were 1 and 8, respectively. Drawing 100,000 values from $\pi(k_0, k_1, k_2)$ took less than a second on a 450 MHz Sun SPARC processor.

(4) $\sigma_2(k)$

It remains only to evaluate the score function

$$S(k,k_0,k_1,k_2) = \delta_{k,k_0}\, V(k-k_1)\cdot V(k_1-k_2)\frac{G(k_1)\cdot G(k_2)\cdot G(k_2+k_0-k_1)}{|G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k_0-k_1)|}.$$

at the values we have drawn from $\pi(k_0,k_1,k_2)$ and a fixed external input k in order to produce the estimate for $\sigma_2(k)$ given by (6.25).

As a simple model potential, we consider

$$V_{j'j} = \begin{cases} 4t & \text{for } j=j' \\ t & \text{for } j \text{ and } j' \text{ nearest neighbors} \\ t/\sqrt{2} & \text{for } j \text{ and } j' \text{ second nearest neighbors} \\ 0 & \text{otherwise.} \end{cases}$$

Here, j and j' label the sites of the 4 points in

$$\mathcal{B} = \mathcal{L}_1 \times \mathcal{L}_2$$

and "nearest neighbors" of a site are the four nearest neighbors with periodic boundary conditions. Recall that our score function contains $V(\vec{k})$ factors where $V(\vec{k})$ denotes the Fourier transform of the interaction potential:

$$V(\vec{k}) := N^{-1}\sum_{j'j} e^{-i\vec{k}\cdot(\vec{r}_{j'}-\vec{r}_j)}V_{j'j} \tag{6.27}$$

where $\vec{r}_j$ denotes the position vector of site j and N is the total number of points in $\mathcal{B}$.

In Table 6.4 we compare the true value of $\sigma_2(0,0,i\pi T)$ with simulated values based on various sample sizes. In Table 6.5 we compare the true value of $\sigma_2(k)$ for each $k\in\mathcal{B}\times\mathcal{M}$ with simulated values based on a sample of size 100,000. It should be pointed out that, for this example, the δ_{k,k_0} term in the score function allowed only approximately 12.5% of the simulated values drawn from $\pi(k_0,k_1,k_2)$ to contribute to our estimate for $\sigma_2(k)$.

Table 6.4: Estimated value of $\sigma_2(0,0,i\pi T)$ based on various sample sizes.

True value of $\sigma_2(0,0,i\pi T) = 0.010954 + 0.997002\ i$

sample size = 100	abs. err.	rel. err.	sample size = 1000	abs. err.	rel. err.
$0.0706 + 1.8947\ i$	0.8997	0.9023	$0.0406 + 1.3259\ i$	0.3302	0.3312
$0.0835 + 1.4858\ i$	0.4941	0.4956	$0.0510 + 1.2857\ i$	0.2915	0.2923
$0.0307 + 0.8488\ i$	0.1495	0.1450	$-0.0025 + 0.9735\ i$	0.0271	0.0271
$-0.0209 + 1.2328\ i$	0.2380	0.2387	$0.0458 + 0.9932\ i$	0.0350	0.0351
$0.1544 + 1.3452\ i$	0.3766	0.3777	$0.0113 + 0.9634\ i$	0.0336	0.0337
$-0.0366 + 0.9466\ i$	0.0693	0.0695	$0.0103 + 1.3237\ i$	0.3267	0.3277

sample size = 10000	abs. err.	rel. err.	sample size = 100000	abs. err.	rel. err.
$0.0122 + 1.0467\ i$	0.0498	0.0499	$0.0108 + 0.9979\ i$	0.0009	0.0009
$0.0130 + 0.9684\ i$	0.0286	0.0287	$0.0109 + 0.9974\ i$	0.0003	0.0004
$0.0108 + 0.9959\ i$	0.0011	0.0011	$0.0107 + 0.9963\ i$	0.0007	0.0007
$0.0110 + 0.9910\ i$	0.0030	0.0060	$0.0109 + 0.9974\ i$	0.0004	0.0004
$0.0118 + 1.1340\ i$	0.1370	0.1375	$0.0110 + 0.9966\ i$	0.0004	0.0004
$0.0200 + 1.0035\ i$	0.0111	0.0112	$0.0101 + 0.9983\ i$	0.0016	0.0016

Table 6.5: True versus estimated value of $\sigma_2(k)$ based on a sample of size 100,000.

k	$\sigma_2(k)$	$\hat{\sigma}_2(k)$	abs. err.	rel. err.
$(0,0,-i\,\pi\,T)$	$0.0110 - 0.9970\ i$	$0.0109 - 0.9961\ i$	0.0009	0.0009
$(0,0,i\,\pi\,T)$	$0.0110 + 0.9970\ i$	$0.0110 + 1.0151\ i$	0.0181	0.0182
$(0,\pi,-i\,\pi\,T)$	$-0.0537 - 2.2951\ i$	$-0.0541 - 2.2978\ i$	0.0027	0.0012
$(0,\pi,i\,\pi\,T)$	$-0.0537 + 2.2951\ i$	$-0.0532 + 2.2844\ i$	0.0107	0.0047
$(\pi,0,-i\,\pi\,T)$	$-0.0537 - 2.2951\ i$	$-0.0542 - 2.2690\ i$	0.0261	0.0114
$(\pi,0,i\,\pi\,T)$	$-0.0537 + 2.2951\ i$	$-0.0531 + 2.3001\ i$	0.0051	0.0022
$(\pi,\pi,-i\,\pi\,T)$	$0.0155 - 1.1990\ i$	$0.0109 - 1.1839\ i$	0.0158	0.0132
$(\pi,\pi,i\,\pi\,T)$	$0.0155 + 1.1990\ i$	$0.0165 + 1.1952\ i$	0.0039	0.0033

Example 2

We take $D = 2$, $L_1 = L_2 = 64$, and 50 odd Matsubara frequencies. Thus, our 9-dimensional state space has approximately 8.6×10^{15} points, making a brute force approach rather unappealing.

Drawing 100,000 values from $\pi(k_0, k_1, k_2)$ took approximately 12 seconds on a 450 MZ Sun SPARC processor.

Our simulation resulted in the estimate

$$\hat{N}_W = 3.483866904037023 \times 10^{10}.$$

It would be of little value in this case to attempt an estimate of $\sigma_2(0, 0, i\,\pi T)$ as the δ_{k,k_3} factor would take the value 1 with probability

$$\sum_{k_1,k_2} \pi((0, 0, i\,\pi T), k_1, k_2) \approx \frac{1}{\hat{N}_W} \sum_{k_1,k_2} h((0, 0, i\,\pi T), k_1, k_2).$$

A very rough upper bound on this is given by

$$\frac{1}{\hat{N}_W}\,(64 \cdot 64 \cdot 50)^2 \max_{k_1,k_2} h((0, 0, i\,\pi T), k_1, k_2) \leq \frac{1}{\hat{N}_W}\,(64 \cdot 64 \cdot 50)^2\,(\max_{k} |G(k)|)^3$$
$$\approx \frac{1}{\hat{N}_W}\,(64 \cdot 64 \cdot 50)^2\,0.159155^3 \approx 0.004854.$$

Here, $\max_k |G(k)|$ was obtained numerically.

However, Monte Carlo simulations give the more realistic estimate

$$\frac{1}{\hat{N}_W}\ 645603.228730 \approx 0.000019$$

In other words, almost none of the values drawn from $\pi(k_0, k_1, k_2)$ contribute to the sum. Clearly another approach is needed here, and we provide two alternatives that address this "low-score problem" by removing the δ_{k,k_0} factor. In Section 6.6.1 we use conditional averaging to change the score function while still using the current weight. In Section 6.6.2 we change the weight and at the same time reduce the dimensionality of the problem using the modified IMH algorithm given by step $3(a)'$ in Section 3.3.2.

6.6 Addressing the Low-Score Problem

6.6.1 Conditional Averaging

In this section, we change the score function (6.22) in order to remove the δ_{k,k_0} "spike" while retaining the original weight function.

Recall from (6.20) that

$$\begin{aligned}\sigma_2(k) &= \sum_{k_0,k_1,k_2\in\mathcal{K}} S(k,k_0,k_1,k_2)W(k_0,k_1,k_2)\\ &= N_W \sum_{k_0,k_1,k_2\in\mathcal{K}} S(k,k_0,k_1,k_2)\pi(k_0,k_1,k_2)\end{aligned}$$

where

$$N_W = \sum_{k_0,k_1,k_2} W(k_0,k_1,k_2).$$

Using the conditional density

$$\pi(k_0|k_1,k_2) := \frac{\pi(k_0,k_1,k_2)}{\pi(k_1,k_2)}$$

where $\pi(k_1,k_2)$ is the marginal density

$$\pi(k_1,k_2) := \sum_{k_0} \pi(k_0,k_1,k_2), \tag{6.28}$$

we may write

$$\begin{aligned}\sigma_2(k) &= N_W \sum_{k_0,k_1,k_2} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2)\pi(k_1,k_2) &(6.29)\\ &= N_W \sum_{k_1,k_2}\left[\sum_{k_0} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2)\right]\pi(k_1,k_2) &(6.30)\\ &= N_W \sum_{k_1,k_2} S'(k,k_1,k_2)\pi(k_1,k_2) &(6.31)\end{aligned}$$

where

$$S'(k,k_1,k_2) := \sum_{k_0} S(k,k_0,k_1,k_2)\pi(k_0|k_1,k_2).$$

Since we are already drawing values from $\pi(k_0,k_1,k_2)$, we take advantage of the relationship given by (6.28) to write (6.29) as

$$\sigma_2(k) = N_W \sum_{k_0,k_1,k_2} S'(k,k_1,k_2)\pi(k_0,k_1,k_2). \tag{6.32}$$

Note that we have removed k_0 from the score function and, in particular, we have removed the troublesome δ_{k,k_0} factor.

Replacing k by k_0 in (6.22), the original score function has the form

$$S(k, k_0, k_1, k_2) = \delta_{k,k_0}\, R(k_0, k_1, k_2).$$

Thus, the conditionally averaged score function $S'(k, k_1, k_2)$ may be written as

$$\begin{aligned} S'(k, k_1, k_2) &= R(k, k_1, k_2)\frac{\pi(k, k_1, k_2)}{\pi(k_1, k_2)} && (6.33) \\ &= R(k, k_1, k_2)\frac{W(k, k_1, k_2)}{W(k_1, k_2)} && (6.34) \end{aligned}$$

where

$$W(k_1, k_2) := \sum_{k'} W(k', k_1, k_2). \qquad (6.35)$$

By implementing this conditional averaging approach, we have eliminated the δ_{k,k_0} factor which enables us to make use of all values drawn from $\pi(k_0, k_1, k_2)$ while simultaneously reducing the variance of the Monte Carlo estimate for $\sigma_2(k)$. We have, however, increased computation costs through (6.35) by a factor of $L_1 \times L_2 \times \cdots \times L_d \times |\mathcal{M}_T|$, where $|\mathcal{M}_T|$ is the size of $\mathcal{M}_T$.

Example 1 Revisited

Once again, we take $D = 2$, $L_1 = L_2 = 2$ and only two odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and the physical temperature is $T = 2.0$.

From (6.32) we have that

$$\sigma(k) = N_W\, \mathsf{E}[S'(k, k_1, k_2)].$$

We use the previous estimate

$$\hat{N}_W = 1.349738$$

and average values of $S'(k, k_1, k_2)$ at values for k_1 and k_2 from draws (k_1, k_2, k_3) from $\pi(k_1, k_2, k_3)$ for various sample sizes. Results for 6 different simulation runs are shown in Table 6.6.

Example 2 Revisited

Again, we take $D = 2$, $L_1 = L_2 = 64$ and 50 odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and the physical temperature is $T = 2.0$. Using the conditionally averaged score function, we arrive at the estimate $\hat{\sigma}_2(0, 0, i\,\pi T) \approx -285.6363 - 2868.7525\,i$, which, when multiplied by the constants we have ignored to this point, contributes

$$\left(\frac{2.0}{64 \cdot 64}\right)^2 [-285.6363 - 2868.7525\,i] \approx -0.000068 - 0.000684\,i$$

to the self energy $\sigma(k)$ given by (6.3).

6.6.2 Using IMH Step $3(a)'$

In Section 6.5 we used the basic IMH algorithm to draw values from the weight function given by (6.21) as opposed to the somewhat more natural weight function given by (6.18)

$$|G(k_2)| \cdot |G(k_2 + k - k_1)|.$$

where k is the external user-input momentum vector that is fixed for any given $\sigma_2(k)$ calculation. The purpose of using (6.21) was so that it would be unnecessary to change the sampling procedure each time we wanted to change the input k. However, there are two obvious drawbacks to this approach.

(1) We increase the size of our state space. In this example, we go from having to sample values for (k_1, k_2) to having to sample values for (k_0, k_1, k_2).

(2) Even with an efficient approach to sampling from the weight function, the δ_{k,k_0} in the score function causes us to effectively discard most of the sampled values. In Section 6.5, the score function turned out to be zero approximately 87.5% of the time in Example 1 and almost 100% of the time in Example 2.

Table 6.6: Estimated value of $\sigma_2(0,0,i\,\pi\,T)$ based on various sample sizes using conditional averaging.

True value of $\sigma_2(0,0,i\,\pi\,T) = 0.010954 + 0.9970023\ i$

sample size = 1000	abs. err.	rel. err.	sample size = 10000	abs. err.	rel. err.
$-0.0785 + 0.8776\ i$	0.1492	0.1496	$0.0164 + 0.9388\ i$	0.0584	0.0586
$-0.0206 + 1.2586\ i$	0.2635	0.2643	$-0.1325 + 0.8323\ i$	0.2184	0.2190
$0.2840 + 1.3479\ i$	0.4445	0.4459	$0.0073 + 0.9522\ i$	0.4495	0.0451
$-0.0105 + 1.0547\ i$	0.0616	0.0618	$0.0460 + 1.1042\ i$	0.1127	0.1131
$0.0503 + 0.7586\ i$	0.2416	0.2423	$0.1384 + 1.0787\ i$	0.1514	0.1518
$-0.0306 + 0.9241\ i$	0.0839	0.0842	$0.0108 + 1.0883\ i$	0.0913	0.0916

sample size = 10000	abs. err.	rel. err.	sample size = 100000	abs. err.	rel. err.
$0.0106 + 0.9955\ i$	0.0016	0.0016	$-0.0027 + 0.9906\ i$	0.0150	0.0151
$-0.0176 + 1.0064\ i$	0.0301	0.0302	$0.0107 + 1.0041\ i$	0.0071	0.0071
$0.0385 + 1.0074\ i$	0.0295	0.0295	$0.0114 + 1.0100\ i$	0.0130	0.0131
$0.0140 + 0.9569\ i$	0.0402	0.0431	$0.0155 + 0.9975\ i$	0.0046	0.0046
$0.0170 + 1.0235\ i$	0.0272	0.0273	$-0.0106 + 0.9805\ i$	0.0271	0.0272
$0.0247 + 0.9782\ i$	0.0233	0.0234	$0.0109 + 0.9963\ i$	0.0007	0.0008

In this section, we consider sampling directly from (6.18) in order to avoid the two drawbacks listed above. We can actually write down a single procedure that is efficient in the sense that will work for any k if we can accept the trade-off that it may become inefficient in some cases due to increased backward coupling times. (In some cases, however they may decrease since we will be making the state space smaller.)

For any fixed k, we write $\sigma_2(k)$ as

$$\begin{aligned}\sigma_2(k) &= \sum_{k_1,k_2\in\mathcal{K}} V(k-k_1)\cdot V(k_1-k_2)\cdot G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)\\ &= \sum_{k_1,k_2\in\mathcal{K}} V(k-k_1)\cdot V(k_1-k_2)\frac{G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)}{|G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|}W_k(k_1,k_2)\\ &= \sum_{k_1,k_2\in\mathcal{K}} S(k,k_1,k_2)\,W_k(k_1,k_2)\end{aligned}$$

where $S(k,k_1,k_2)$ is the score function

$$S(k,k_1,k_2) := \frac{V(k-k_1)\cdot V(k_1-k_2)\cdot G(k_1)\cdot G(k_2)\cdot G(k_2+k-k_1)}{|G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|}$$

and $W_k(k_1,k_2)$ is the weight function

$$W_k(k_1,k_2) := |G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|.$$

We wish to use the IMH algorithm to draw values from

$$\pi(k_1,k_2)\ \propto\ h(k_1,k_2)\equiv W_k(k_1,k_2).$$

We again choose a simple uniform candidate distribution this time with density $q(k_1,k_2)$ which gives equal weight to all points $(k_1,k_2)\in(\mathcal{B}\times\mathcal{M}_T)^2$.

In order to implement the IMH algorithm, it remains to identify the "lowest point", (k_1^*,k_2^*), which is the point in $(\mathcal{B}\times\mathcal{M}_T)^2$ that maximizes the ratio h/q. Since q is constant for all $(k_1,k_2)\in(\mathcal{B}\times\mathcal{M}_T)^2$, we simply want to maximize

$$h(k_1,k_2) = |G(k_1)|\cdot|G(k_2)|\cdot|G(k_2+k-k_1)|. \tag{6.36}$$

As in Section 6.4.2 we can, after a limited search, identify a point $k^*=(\vec{k}^*, i\,\pi T)$ (where $\vec{k}^*$ is two dimensional) that maximizes $|G(k)|$. This time however, it is more difficult to simultaneously maximize all factors in (6.36) as we will not necessarily be able to simply make all three arguments equal to k^* at the same time. It is not desirable to search through all k_1 and k_2 for a fixed k in order to maximize (6.36) for three reasons:

(1) The search would need to be performed again each time we change the external k.

(2) The search would need to be performed again for every graph included in the overall summation defining $\sigma(k)$.

(3) The search space will increase in size with higher order graphs and, with higher dimensions, (i.e. large D, $L_1, \ldots, L_d$), it will increase much more quickly than for the limited search described in Section 6.4.2.

Given that we do not want to maximize (6.36), we appeal to Step $3(a)'$ of the IMH algorithm from Section 3.3.2.

Since we use a uniform candidate density, the π/q ratio we need to bound is

$$\max_{k_1,k_2} |G(k_1)| \cdot |G(k_2)| \cdot |G(k_2 + k - k_1)| \leq |G(k^*)| \cdot |G(k^*)| \cdot |G(k^*)|,$$

so we take C to be $|G(k^*)|^3$.

Example 1 Revisited

Once again, we take $D = 2$, $L_1 = L_2 = 2$ and only two odd Matsubara frequencies. The parameters are $\mu = 0.5$, $t = 1.0$, and $T = 2.0$. We report estimates for $\sigma_2(\pi, 0, i\,\pi\,T)$ in Table 6.7. As expected, we have improved accuracy for small sample sizes since we are able to make use of all values drawn from the weight when evaluating the score function. The average backward coupling time for 100,000 draws was still 1.2 time steps in this case with a minimum of 1 and a maximum of 7.

Table 6.7: Estimated Value of $\sigma_2(0, 0, i\,\pi\,T)$ based on various sample sizes using the modified IMH algorithm.

True value of $\sigma_2(0, 0, i\,\pi\,T) = 0.010954 + 0.997002\,i$

sample size = 100	abs. err.	rel. err.	sample size = 1000	abs. err.	rel. err.
$0.0180 + 1.1990\,i$	0.2021	0.2027	$-0.1192 + 0.8778\,i$	0.1765	0.1770
$-0.3235 + 1.0589\,i$	0.3401	0.3411	$-0.0395 + 1.0173\,i$	0.0544	0.0545
$-0.0018 + 0.9160\,i$	0.0820	0.0823	$-0.1641 + 0.9528\,i$	0.1805	0.1810
$-0.0130 + 0.8970\,i$	0.1028	0.1031	$-0.0119 + 0.9727\,i$	0.0333	0.0335
$-0.0256 + 1.0099\,i$	0.0388	0.0389	$0.0132 + 1.0930\,i$	0.0961	0.0963
$0.0725 + 1.0963\,i$	0.1168	0.1171	$0.0035 + 0.9326\,i$	0.0645	0.0651

sample size = 10000	abs. err.	rel. err.	sample size = 100000	abs. err.	rel. err.
$0.0138 + 0.9639\,i$	0.0332	0.0333	$0.0110 + 0.9969\,i$	0.0001	0.0001
$0.0135 + 0.9993\,i$	0.0034	0.0034	$0.0107 + 0.9972\,i$	0.0003	0.0003
$0.0263 + 0.9912\,i$	0.0164	0.0165	$0.0110 + 0.9970\,i$	0.0000	0.0000
$0.0235 + 0.9404\,i$	0.0579	0.0581	$0.0106 + 0.9988\,i$	0.0019	0.0019
$-0.0120 + 1.0242\,i$	0.0356	0.0357	$0.0108 + 0.9953\,i$	0.0017	0.0017
$0.0208 + 0.9674\,i$	0.0312	0.0313	$0.0157 + 0.9922\,i$	0.0068	0.0068

Example 2 Revisited

We again consider the case where $D = 2$, $L_1 = L_2 = 64$ with 50 odd Matsubara frequencies. Using the modified IMH algorithm described at the beginning of this section, we arrive at the estimate $\hat{\sigma}_2(0, 0, i\ \pi T) \approx -286.0717 - 2876.0375\ i$ which, when multiplied by the constants we have ignored to this point, contributes

$$\left(\frac{2.0}{64 \cdot 64}\right)^2 [-286.0717 - 2876.0375\ i] \approx 0.0000068 - 0.000686\ i$$

to the self energy $\sigma(k)$ given by (6.3).

It is interesting to note that even though the state space was of size $(64 \cdot 64 \cdot 50)^2 = 41, 943, 040, 000$, roughly one-third of our 100,000 draws from the weight function, required that we only go back in time 1 step to achieve the coupling. In 50% of the draws, we had to go back 3 steps or less and 75% of the time the backward coupling time was less than or equal to 9. The average backward coupling time in 100,000 draws was 20.5. This was skewed upwards by a few backward coupling times of just over 2,000, but 90% of the draws were made in less than 33 time steps.

Chapter 7

Conclusions

7.1 Summary

Perfect sampling is variant of MCMC which eliminates the statistical error that conventional MCMC algorithms introduce, thereby avoiding the often tedious question of convergence. The drawback of perfect simulation is that these techniques are generally hard to apply to more complicated problems.

In this thesis, we presented several new ideas around perfect sampling (the **advances** in Chapter 3 and Chapter 4) and showed how these new concepts could be applied to two relevant problems (the **applications** in Chapter 5 and Chapter 6). We also believe that these techniques could facilitate the use of perfect simulation to a wider range of problems aside from the applications described in this dissertation.

Two topics for future research are mentioned in the following section.

7.2 Future Research

We list two ideas for future research. The first idea, described in Section 7.2.1, concerns the adaptive IMH algorithm from Chapter 3 and is a general question regarding the analytical assessment of conveWe state another open question for future work in Section 7.2.2, where we present a specific idea that may address and reduce the computational cost of the IMH algorithm as it is used in Chapter 5.

7.2.1 Analytical Bounds for the AIMH algorithm

In Chapter 3 in Section 3.4.2, we discussed a promising variant on the Metropolis-Hastings algorithm, the so-called adaptive IMH algorithm, that uses a self-adapting candidate and seems to converge extremely rapidly to the target distribution.

We would like to be able to quantify this convergence analytically, i.e. we would like to be able to compute bounds for

$$\|\pi - \pi_{r,N}\|,$$

where π is the target distribution, $\pi_{r,N}$ the distribution of the Metropolis-Hastings chain after r refinements of N draws each, and $\|\cdot\|$ is the total variation norm, as defined in Section 2.1.

While this is a question that has yet to be investigated, possible clues on how to approach this might be found in [17].

7.2.2 Towards a Multistage Coupler for the IMH Algorithm

In Chapter 5 in Section 5.4.3 and Section 5.4.4, we found that applying the IMH algorithm to the variable selection problem results in very large backward coupling times. While our formulation of variable selection represents a specific problem for which we employed the IMH algorithm, we believe that addressing the computational cost for this application would give insight into how to make the IMH algorithm practicable for many other (for example, Bayesian) applications. Due to the straightforward implementation of IMH, this could facilitate a larger practical impact of perfect sampling algorithms.

The following idea could help to drastically reduce the large backward coupling times for the IMH algorithm. The state space is partitioned into two clusters. In the first step, a procedure determines into which two cluster the current draw will fall. Then, in the second step, a draw from the target distribution is made **restricted** to that particular cluster. (This general idea is due to Meng [34] and also mentioned in Murdoch [42].)

To describe this idea in more detail within the framework of model selection, assume that we would like to draw from a target density $\pi(x) \propto L(x)q(x)$ and that we intend to apply an IMH algorithm with candidate density $q(x)$. Moreover, suppose that $\max\limits_x \frac{\pi(x)}{q(x)} = \max\limits_x L(x) \leq 1$.

First, the partition of the state space S is made the following way. We define a *high likelihood cluster* $M_1 := \{x \in X | L(x) \geq k\}$ and a *low likelihood cluster* $M_2 := X \setminus M_1 = \{x \in X | L(x) < k\}$ for a fixed k that is small "enough". (How k should be chosen is discussed later.)

In the first stage we perform an "IMH-like" step, but we only go back in time far enough to determine in which *cluster* the draw X_0 falls into.

First stage

(1) Draw a candidate $Q \sim q(\cdot)$ and a $U \sim \text{Uniform}(0, 1)$.

(2) If $U < \frac{L(Q)}{k}$ (we then have $U < \frac{L(Q)}{k} < \frac{L(Q)}{L(y)}$ for all $y \in M_2$, so that every element in M_2 will *accept*).

  (a) If $Q \in M_1$, then $X_0 \in M_1$, **report M_1 as the cluster**.

  (b) If $Q \in M_2$, do a full IMH-step and **report the cluster of X_0**.

(3) If $U \geq \frac{L(Q)}{k}$ (we then have $U \geq \frac{L(Q)}{k} \geq \frac{L(Q)}{L(y)}$ for all $y \in M_1$, so that every element in M_2 will *reject*):

  (a) If $X_{-1} \in M_1$, then $X_0 \in M_1$, *report M_1 as the cluster.*

  (b) If the cluster of X_{-1} is unknown, do a full IMH-step and *report the cluster of* X_0.

If we choose k small enough so that $q(M_1) \approx \pi(M_1) \approx 1$, we can expect to end up in step (2) (a) "most of the time". In that case we only have to go back one time step to determine the likelihood cluster of the current draw. To determine that $X_{-1} \in M_1$ in step (3) (a), we simply repeat the procedure with a new candidate Q and a new random number $U \sim \text{Uniform}(0, 1)$ to see if we end up in step (2) (a). In the cases where a full (regular) IMH-step has to be done, we only report the **cluster** of X_0 in order not to bias the samples.

Second stage

Once the first stage is completed, we know to which cluster to restrict the simulations in the second stage. For the multistage coupler to be useful, it is crucial that we know an efficient way to draw from $\pi|M_1$. Usually, M_1 is a compact set. That information could help in finding a way to sample within that cluster. If the cluster from the first stage turns out to be M_2, a possible way to simulate from $\pi|M_2$ is to reduce k and start another multistage coupler with M_2 as the state space.

Remark

In the context of model selection, for example, for the case Section 5.3.3 that was solved using an IMH algorithm, a multistage coupler could be applied once an efficient way to sample from $\pi|M_1$ is found.

Bibliography

[1] S. Asmussen. Applied Probability and Queues. John Wiley & Sons, New York, 1987.

[2] S.P. Brooks and G.O. Roberts. Assessing convergence of Markov chain Monte Carlo algorithms. Stat. Comput., 8:319–335, 1998.

[3] G. Casella and E. George. Explaining the Gibbs sampler. Am. Stat., 46:167–174, 1992.

[4] G. Casella, M. Lavine, and C. Robert. Explaining the perfect sampler. Working Paper 00-16, Duke University, 2000.

[5] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. Am. Stat., 49:327–335, 1992.

[6] H. Chipman, E.I. George, and R.E. McCulloch. The practical implementation of Bayesian model selection. IMS Lect. Notes Monogr. Ser. (Model Selection), 38:67–116, 2001.

[7] J. Corcoran and U. Schneider. Variants on the independent Metropolis-Hastings algorithm: approximate and adaptive IMH. In preparation, 2003.

[8] J.N. Corcoran and U. Schneider. Shift and scale coupling methods for perfect simulation. Prob. Eng. Inf. Sci., 17:277–303, 2003.

[9] J.N. Corcoran, U. Schneider, and H.-B. Schüttler. Perfect stochastic summation for high order Feynman diagrams. Submitted for publication, 2003. Preprint at `http://amath.colorado.edu/student/schneidu/feynman.html`.

[10] J.N. Corcoran and R.L. Tweedie. Perfect sampling of ergodic Harris chains. Ann. Appl. Prob., 11(2):438–451, 2001.

[11] J.N. Corcoran and R.L. Tweedie. Perfect sampling from independent Metropolis-Hastings chains. J. Stat. Plann. Inference, 104:297–314, 2002.

[12] M.K. Cowles and B.P. Carlin. Markov chain Monte Carlo convergence diagnostics: A comparative review. J. Am. Stat. Ass., 91:883–904, 1996.

[13] N. Draper and H. Smith. Applied Regression Analysis. Wiley, New York, second edition, 1981.

[14] R. P. Feynman. The Character of Physical Law. M.I.T. Press, 1967.

[15] J.A. Fill. An interruptible algorithm for perfect sampling via Markov chains. Ann. Appl. Prob., 8:131–162, 1998.

[16] S.G. Foss and R.L. Tweedie. Perfect simulation and backward coupling. Stoch. Models, 14:187–203, 1998.

[17] S.G. Foss, R.L. Tweedie, and J.N. Corcoran. Simulating the invariant measures of Markov chains using horizontal backward coupling at regeneration times. Prob. Eng. Inf. Sci., 12:303–320, 1998.

[18] P. Fulde. Electron Correlations in Molecules and Solids, Solid State Science. Springer, Berlin, Heidelberg, third edition, 1995.

[19] A.E. Gelfand and A.F.M. Smith. Sampling-based approaches to calculating marginal densities. J. Am. Stat. Ass., 85:398–409, 1990.

[20] A. Gelman. Inference and monitoring convergence. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, Markov Chain Monte Carlo in Practice, chapter 8, pages 131–143. Chapman & Hall, first edition, 1996.

[21] S. Geman and D. Geman. Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Analysis & Machine Intelligence, 6:721–740, 1984.

[22] E.I. George. The variable selection problem. Downloadable from `http://www-stat.wharton.upenn.edu/~edgeorge/Research_papers/vignette.pdf`.

[23] E.I. George and R.E. McCulloch. Variable selection via Gibbs sampling. J. Am. Stat. Ass., 88:881–889, 1993.

[24] E.I. George and R.E. McCulloch. Stochastic search variable selection. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, Markov Chain Monte Carlo in Practice, chapter 12, pages 203–214. Chapman & Hall, first edition, 1996.

[25] E.I. George and R.E. McCulloch. Approaches to Bayesian variable selection. Stat. Sin., 7:339–373, 1997.

[26] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. Introducing Markov chain Monte Carlo. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, Markov Chain Monte Carlo in Practice, chapter 1, pages 1–19. Chapman & Hall, first edition, 1996.

[27] A. Hald. Statistical Theory with Engineering. Wiley publications in statistics. Wiley, New York, 1952.

[28] On the Hald data set.
`http://www.stat.uconn.edu/~nalini/fcilmweb/example13.html`.

[29] A. Hall. On an experiment determination of π. Messeng. Math, 2:113–114, 1873.

[30] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57(1):97–109, 1970.

[31] Y. Huang and P. Djurić. Variable selection by perfect sampling. EURASIP J. Appl. Si. Pr., 1:38–45, 2002.

[32] W.S. Kendall. Perfect simulation for the area-interaction point process. In L. Accardi and C.C. Heyde, editors, Probability Towards the Year 2000, pages 218–234. Springer, New York, 1998.

[33] G.D. Mahan. Many Particle Physics. Plenum Press, New York, second edition, 1990.

[34] X.-L. Meng. Towards a more general Propp-Wilson algorithm: Multistage backward coupling. Monte Carlo Methods - Fields Inst. Commun., 26:85–93, 2000.

[35] K.L. Mengersen and Tweedie R.L. Rates of convergence of the Hastings and Metropolis algorithms. Ann. Stat., 24:101–121, 1996.

[36] N. Metropolis and S. Ulam. The Monte Carlo method. J. Am. Stat. Ass., 44(247):335–341, 1953.

[37] A. Mira, J. Møller, and G.O. Roberts. Perfect slice samplers. J. Roy. Stat. Soc., 63:593–606, 2001. 3.

[38] Model selection reading list. `http://www.stat.duke.edu/~cnk/readings.html`.

[39] J. Møller. Perfect simulation of conditionally specified models. J. Roy. Stat. Soc., 61(1):251–264, 1999.

[40] The Monte Carlo method.
`http://www.riskglossary.com/articles/monte_carlo_method.htm`.

[41] P. Monthoux and D.J. Scalapino. Self-consistent $d_{x^2-y^2}$ pairing in a two-dimensional Hubbard model. Phys. Rev., 72:1874, 1994.

[42] D.J. Murdoch. Exact sampling for Bayesian inference: Unbounded state spaces. Monte Carlo Methods - Fields Inst. Commun., 26:111–121, 2000.

[43] D.J. Murdoch and P.J. Green. Exact sampling from a continuous state space. Scand. J. Stat., 25(3):483–502, 1998.

[44] J.W. Negele and H. Orland. Quantum Many-Particle Systems. Addison Wesley, Redwood City, CA, 1988.

[45] C.-H. Pao and N.E. Bickers. Renormalization group acceleration of self- consistent field solutions: two-dimensional Hubbard model. Phys. Rev., 49(1586), 1994.

[46] J.G. Propp and D.B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Struct. Algorithms, 9:223–252, 1996.

[47] A.E. Raferty, D. Madigan, and J.A. Hoeting. Bayesian model averaging for linear regression models. J. Am. Stat. Ass., 92:179–191, 1997.

[48] C.P. Robert and G. Casella. Monte Carlo Statistical Methods. Springer, New York, 1999.

[49] S.M. Ross. Probability Models, chapter 4.9 – Markov Chain Monte Carlo Methods, pages 216–222. Academic Press, eighth edition, 2000.

[50] S.M. Ross. Simulation, chapter 10 – Markov Chain Monte Carlo Methods, pages 223–250. Academic Press, San Diego, third edition, 2002.

[51] U. Schneider. Informal notes on Fill's algorithm. Downloadable from `http:// amath.colorado.edu/student/schneidu/fill.html`.

[52] U. Schneider and J.N. Corcoran. Perfect sampling for Bayesian variable selection in a linear regression model. J. Stat. Plann. Inference, 2003. To appear. Preprint at `http://www.cgd.ucar.edu/stats/staff/uli/modelselect.html`.

[53] O. Stramer and R.L. Tweedie. Self-targeting candidates for Hastings- Metropolis algorithms. Technical Report 261, Department of Statistics, The University of Iowa, 1997.

[54] O. Stramer and R.L. Tweedie. Langevin-type models ii: Self-targeting candidates for mcmc algorithms. Meth. Comp. Appl. Prob., 1:307–328, 1999.

[55] E. Thönnes. A primer in perfect simulation. Lect. Notes Phys., pages 349–378, 2000.

[56] L. Tierney. Markov chains for exploring posterior distributions (with discussion). Ann. Stat., 1:1701–1762, 1994.

[57] R.L. Tweedie and J.N. Corcoran. Perfect sampling and queueing models. Proc. 38th Annual Allerton Conf. Comm. Control Comput., 2001. (to appear).

[58] A. Voight, C. Liu, Q. Wang, R.W. Robinson, and H.B. Schüttler. Stochastic Feynman diagram summation for the Anderson impurity model. Submitted for publication, 2002. Preprint at `http://www.csp.uga.edu/publications/HBS/`.

[59] D.B. Wilson. Web site for perfectly random sampling with Markov chains. `http://dimacs.rutgers.edu/~dbwilson/exact.html`.

[60] D.B. Wilson. Layered mutishift coupling for use in perfect sampling algorithms (with a primer on cftp). Fields Inst. Commun., 26:141–176, 2000.

[61] H. Woods, H.H Steinour, and H.R. Starke. Effect of composition of portland cement on heat evolved during hardening. Ind. Eng. Chem., 24(11):1207–1214, 1932.

Appendix A

Gibbs Sampling

The *Gibbs sampler* is an MCMC method that allows to get (approximate) samples from a multivariate distribution with density $\pi(x_1, \dots, x_n)$ by drawing following transition updates simulating from $(n-1)$-dimensional conditional distributions.

Since we use this technique in various sections (as Section 4.5.1, Section 4.5.2, Section 5.3.1, and Section 5.3.2), we shortly describe the simulation procedure here. There are several easy-to-read introductions and tutorials on the Gibbs sampler, including [3, 48, 49, 26, 50].

To be able to use Gibbs sampling, **one needs to be able to sample from the conditional distributions of each component given the remaining components**, i.e. to simulate from

$$\pi_{X_i|\boldsymbol{X}_{-i}}(x_i|\boldsymbol{x}_{-i}) \quad \text{where } \boldsymbol{X}_{-i} = (X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n). \tag{A.1}$$

Let $\boldsymbol{x}^{(t)} = (x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)})$ denote the values of the chain at time t. To transition to time $t+1$, one needs to follow the Gibbs updates which can be characterized as follows.

For each component $i = 1, \dots, n$, sequentially draw

$$X_i^{(t+1)} = x_i^{(t+1)} \sim \pi_{X_i|\boldsymbol{X}_{-i}}(x_i|\boldsymbol{x}^{(t+1)}), \tag{A.2}$$

where $\boldsymbol{x}_{-i}^{(t+1)} = (x_1^{(t+1)}, \dots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \dots, x_n^{(t)})$.

Remarks

(1) To update the i^{th} component for time $t+1$, we incorporate the "new" (from time $t+1$) information for the previous $i-1$ components.

(2) Note that as long as we can specify the conditional distributions in (A.1), we do not need to know the constant of proportionality for π.

(3) We refer the reader to one of the above mentioned references for a proof how the sampling scheme in (A.2) can be seen as a special case of a Metropolis-Hastings chain.

Appendix B

Hierarchical Sampling

What we call *hierarchical sampling* is a (well-known) sampling scheme to simulate from a joint distribution given by a marginal times a joint density. In this appendix, we give a quick description of the algorithm and a sketch of a proof of validity.

B.1 Sampling Scheme

Suppose we would like to draw from a joint distribution with density $\pi(x, y) = f_{X,Y}(x, y)$ and assume that we know the marginal density $f_X(x)$ and the conditional density $f_{Y|X}(y|x)$

$$\pi(x, y) = f_X(x) f_{Y|X}(y|x).$$

Furthermore, assume that we know how to sample from the marginal and the conditional distribution. Then, to sample from π, we can employ the following scheme:

(1) draw $X = x_1 \sim f_X(x)$,

(2) draw $Y = y_1 \sim Y|X = x_1 \sim f_{y|x}(y|x_1)$.

This yields $(x_1, y_1) \sim \pi(x, y)$

B.2 Validity

It might be clear intuitively why this scheme represents a valid algorithm to simulate from π, but for completeness, we sketch a proof.

The simulation steps specified above can be viewed as one iteration of a Gibbs sampler from Appendix A that has been started stationary. The output of it will therefore also follow the stationary distribution $\pi(x, y)$.

Assume that we currently have a draw $(x_0, y_0) \sim \pi(x, y)$. In the next iteration in a Gibbs sampling scheme, we do the following:

$(1)'$ draw $X = x_1 \sim X|Y = y_0$

$(2)'$ draw $Y = y_1 \sim Y|X = x_1$

Clearly, $(x_1, y_1) \sim \pi(x, y)$.

The steps for the Gibbs sampler are essentially the same as in the hierarchical sampling scheme:

In step $(1)'$, $X = x_1$ depends on the previous draw through y_0, but its marginal is $f_X(x)$, so it can be viewed as being equivalent to step (1) above. Steps (2) and $(2)'$ are identical.