Commun. Comput. Phys.
doi: 10.4208/cicp.300313.130114a

Vol. **16**, No. 2, pp. 287-306
August 2014

Simulation with Fluctuating and Singular Rates

Farzin Barekat[1] and Russel Caflisch[1,*]

[1] *Mathematics Department, University of California at Los Angeles, Los Angeles, CA 90095-1555, USA.*

Received 30 March 2013; Accepted (in revised version) 13 January 2014

Communicated by Bo Li

Available online 17 April 2014

Abstract. In this paper we present a method to generate independent samples for a general random variable, either continuous or discrete. The algorithm is an extension of the Acceptance-Rejection method, and it is particularly useful for kinetic simulation in which the rates are fluctuating in time and have singular limits, as occurs for example in simulation of recombination interactions in a plasma. Although it depends on some additional requirements, the new method is easy to implement and rejects less samples than the Acceptance-Rejection method.

AMS subject classifications: 65C05, 82B80, 82D10

Key words: Sampling, Monte Carlo, Acceptance/Rejection method, plasma, recombination.

1 Introduction

Kinetic transport for a gas or plasma involves particle interactions such as collisions, excitation/deexcitation and ionization/recombination. Simulation of these interactions is most often performed using the Direct Simulation Monte Carlo (DSMC) method [1] or one of its variants, in which the actual particle distribution is represented by a relatively small number of numerical particles, each of which is characterized by state variables, such as position x and energy E. Interactions between the numerical particles are performed by random selection of the interacting particles and the interaction parameters, depending on the interaction rates. Correctly sampling these interactions involves several computational challenges: First the number N of particles can be large (e.g., $N=10^6$) and the number of possible interaction events can be even larger (e.g., N^k for $k=2$ or 3). Second, the interaction probabilities vary throughout the simulation since interactions

*Corresponding author. *Email addresses:* fbarekat@math.ucla.edu (F. Barekat), caflisch@math.ucla.edu (R. Caflisch)

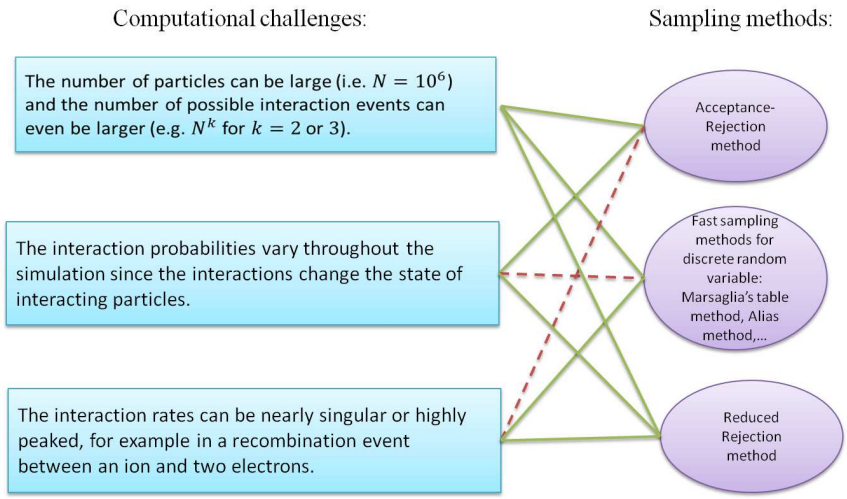

Figure 1: This figure illustrates the computational challenges involved in sampling interactions of numerical particles, and how different methods can handle them. Broken line represents challenges for which the method becomes computationally inefficient, whereas, the solid line represents the challenges for which the method is still computationally efficient.

change the state of the interacting particles. These two difficulties are routinely overcome using Acceptance-Rejection sampling. Third, the interaction rates can be nearly singular, for example in a recombination event between an ion and two electrons (described in more detail in Section 5). This creates a wide range of interaction rates that makes Acceptance-Rejection computationally intractable. Fig. 1 illustrates these challenges and how different methods can handle them. The sampling method presented here, which we call Reduced Rejection, was developed to overcome the challenges of a large number of interaction events with fluctuating and singular rates.

Simulation of kinetics requires sampling methods that generate independent samples. This rules out Markov Chain Monte Carlo schemes, such as Metropolis-Hastings, Gibbs sampling, and Slice sampling. Although these methods are very powerful and are used very often, this paper focuses on sampling methods that generate independent samples.

There are several efficient algorithms for simulation of discrete random variables, notably Marsaglia's Table method [11] and the Alias method [19, 20]. However, these methods require pre-processing time and, therefore, are not efficient for sampling from a random variable whose probability function changes during the simulation. For continuous random variables there are several different algorithms; nevertheless, each of these algorithms has its own constraints. For example, Inverse Transform Sampling method requires knowledge of the cumulative distribution function and evaluation of its inverse, Box-Muller only applies to a normal distribution, and Ziggurat algorithm [12] can be used for random variables that have monotone decreasing (or symmetric unimodal) density function.

The algorithm of choice for general (both continuous and discrete) random vari-

ables that generates independent samples and does not require preprocessing time is Acceptance-Rejection method (see for example [3]). Let $q(x)$ be a real-valued function on the sample space. Let $I[q]$ denote the expectation of function $q(x)$. By sampling according to function $q(x)$ we mean to sample using the probability distribution function $q(x)/I[q]$. We say function $q(x)$ *encloses* function $p(x)$ if $p(x)\leq q(x)$ for all x in the sample space. The idea of Acceptance-Rejection method is to find a proposal function $q(x)$ that encloses function $p(x)$. Suppose we already have a mechanism to sample according to $q(x)$, then Acceptance-Rejection algorithm enables us to sample according to $p(x)$. In most cases the constant function is used as the proposal function $q(x)$. The main drawback of Acceptance-Rejection method is that it might reject many samples. Indeed the ratio of the number of rejected samples to the number of accepted samples is approximately equal to the ratio of the area between curves $q(x)$ and $p(x)$ to the area under the curve $p(x)$.

For many given distributions, finding a good proposal function that encloses it without leading to many rejected samples is difficult. One extension to Acceptance-Rejection method is Adaptive Rejection Sampling [8]. The basic idea of Adaptive Rejection Sampling is to construct proposal function $q(x)$ that encloses the given distribution by concatenating segments of one or more exponential distributions. As the algorithm proceeds, it successively updates the proposal function $q(x)$ to correspond more closely to the given distribution. Another extension to Acceptance-Rejection method is Economical method [5]. This method is basically a generalization of Alias method for continuous distributions. In this method, one needs to define a specific transformation that maps $\{x:p(x)>q(x)\}$ to $\{x:p(x)\leq q(x)\}$. Although this method produces no rejection, finding the required transformation is difficult in general.

In the Reduced Rejection method we sample according to a given function $p(x)$ based on a proposal function $q(x)$. In contrast to the Acceptance-Rejection method, Reduced Rejection sampling does not require $q(x)$ to enclose $p(x)$ (i.e. it allows $p(x)>q(x)$ for some x). On the other hand, Reduced Rejection sampling requires some extra knowledge about the functions $p(x)$ and $q(x)$.

The Reduced Rejection sampling method can be applied to a wide range of sampling problems (for both continuous and discrete random variables) and in many examples is more efficient than customary methods (three examples are provided in Sections 4, 5 and 6). In particular, Reduced Rejection sampling requires no pre-processing time and consequently is suitable for simulations in which $p(x)$ is changing constantly (see Section 3 for an elaboration on this point and Sections 5 and 6 for examples of simulations with fluctuating $p(x)$). Also in situations where $p(x)$ has singularities or is highly peaked in certain regions, Reduced Rejection sampling can be very efficient.

The next section describes the Reduced Rejection sampling and proves its validity. Section 3 compares Reduced Rejection sampling to other methods (including other generalizations of Acceptance-Rejection), highlights advantages of Reduced Rejection sampling in comparison to other methods, and points out some of the challenges in applying Reduced Rejection sampling. In Section 4, Reduced Rejection sampling is demonstrated

on a simple example. In Section 5, Reduced Rejection sampling is applied to an example motivated from plasma physics, for which other sampling methods cannot be used efficiently. In Section 6, we make some comments on how to apply Reduced Rejection in the context of stochastic chemical kinetics. In the appendix, we provide flow charts for the Reduced Rejection algorithm.

2 Reduced Rejection sampling

Consider a sample space Ω with Lebesgue measure μ on Ω, and two functions $q,p:\Omega\to\mathbb{R}$. Denote

$$I[q]=\int_\Omega q(x)d\mu(x),\qquad I[p]=\int_\Omega p(x)d\mu(s).$$

By sampling from Ω according to $p(x)$ we mean sampling from Ω using probability distribution function $p(x)/I[p]$. Partition sample space Ω into two sets $\mathcal{S}$ and $\mathcal{L}$:

$$\mathcal{L}=\{x\in\Omega\colon p(x)>q(x)\},\qquad \mathcal{S}=\{x\in\Omega\colon p(x)\le q(x)\}.$$

Reduced Rejection sampling is a method for sampling from Ω according to $p(x)$ using an auxiliary function $q(x)$. It depends on the following:

- The values of $I[q]$, $I[p]$ and $\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)$. Note that the last value is needed only for "Algorithm II", see Subsection 2.1.
- A mechanism to sample from Ω according to $q(x)$.
- A mechanism to sample from $\mathcal{L}$ according to $p(x)-q(x)$.

Whereas the Acceptance-Rejection method for sampling from $p(x)$ requires a function $q(x)$ that encloses $p(x)$ (i.e., $0\le p(x)\le q(x)$ for all $x\in\Omega$), the Reduced Rejection sampling algorithm is a generalization of the Acceptance-Rejection method, that allows $p(x)>q(x)$ for some x. The Reduced Rejection sampling algorithm is detailed in Section 2.1, and its validity as a method for sampling from Ω according to $p(x)$ is demonstrated in Section 2.2.

2.1 The Reduced Rejection sampling algorithm

The Reduced Rejection sampling method consists of two algorithms (i.e., two different algorithms) depending on the relative values of $I[p]$ and $I[q]$. The outcome of each algorithm is a value z that is an independent sample from Ω according to $p(x)$.

Algorithm I: $\mathbf{I[p]\ge I[q]}$.

Perform the following steps:

i) With probability $(I[p]-I[q])/I[p]$, sample x_0 from $\mathcal{L}$ according to $p(x)-q(x)$ and accept $z=x_0$.

ii) Otherwise (with probability $(I[q]/I[p])$, sample x_0 from Ω according to $q(x)$.

a) If $x_0 \in \mathcal{L}$, accept $z = x_0$.

b) If $x_0 \in \mathcal{S}$, accept $z = x_0$ with probability $p(x_0)/q(x_0)$.

iii) If x_0 was not accepted, then sample a new value of x_1 from $\mathcal{L}$ according to $p(x)-q(x)$ and accept $z = x_1$.

Algorithm II: $\mathbf{I[p] < I[q]}$.

Perform the following steps until a value z is accepted:

i) Sample x_0 from Ω according to $q(x)$.

ii) If $x_0 \in \mathcal{L}$, accept $z = x_0$.

iii) If $x_0 \in \mathcal{S}$, accept $z = x_0$ with probability $p_a = p(x_0)/q(x_0)$,

iv) If x_0 was not accepted, then

a) With probability p_a select x_1 from $\mathcal{L}$ according to $p(x)-q(x)$ and accept $z = x_1$, in which

$$p_a = \frac{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)} = \frac{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}{I[q]-I[p]+\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}. \tag{2.1}$$

b) Otherwise (i.e., with probability $1-p_a$), return to (i) without accepting a value of z.

Figs. 5 and 6 in Appendix 7 illustrate flow charts of Algorithms I and II.

As described in Algorithms I and II, Reduced Rejection samples from p through the following steps: On $\mathcal{L}$, treat p as a mixture $p = q+(p-q)$ and sample from q and $p-q$ with the correct probabilities; and on $\mathcal{S}$, sample from p by sampling from q and accept the sample with probability p/q. Rejected samples in $\mathcal{S}$ correspond to the region B in Fig. 2, and the region A is where q does not enclose p. If $|A| > |B|$ (i.e., Algorithm I) then

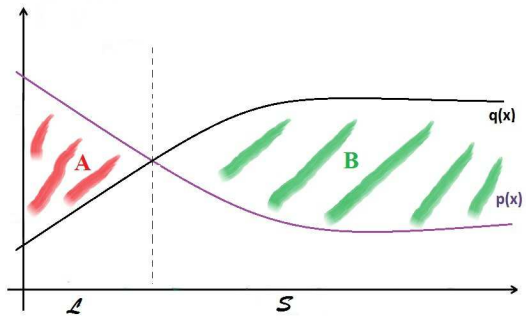

Figure 2: This figure illustrates the Reduced Rejection method. Region A is where q does not enclose p, and region B is where samples are rejected. Rejected samples from region B can be replaced by samples from region A if $|A| > |B|$; otherwise (if $|A| < |B|$), some of the rejected samples lead to repetition of the algorithm.

all of the rejected samples can be replaced by samples from A; if $|A|<|B|$ (i.e., Algorithm II) then a portion of the rejected samples can be replaced by samples from A, and for the remainder, the algorithm is repeated as in Acceptance-Rejection.

2.2 Validity of Reduced Rejection sampling

In this subsection we show the correctness of the Reduced Rejection sampling method. As the method is different for Algorithms I and II, we prove the correctness for each algorithm separately.

Proof for Algorithm I: For each $z\in\Omega$, show that the algorithm of Algorithm I returns z with probability $p(z)d\mu(z)/I[p]$.

If $z\in\mathcal{S}$, then part (ii) must have been selected, z must have been sampled in (ii) and it must have been accepted in case (ii.b). Therefore, the probability of returning z is

$$\begin{aligned}&\Pr[\text{(ii) selected}]\,\Pr[z\text{ sampled in (ii)}]\,\Pr[z\text{ accepted in (ii.b)}]\\ =&\frac{I[q]}{I[p]}\times\frac{q(z)d\mu(z)}{I[q]}\times\frac{p(z)}{q(z)}=\frac{p(z)d\mu(z)}{I[p]}.\end{aligned}\tag{2.2}$$

Also note that for every $x_0\in\mathcal{S}$, after x_0 is selected in (ii.b) with probability $\frac{q(x)}{I[q]}d\mu(x)$, the probability of reaching (iii) is $\frac{q(x_0)-p(x_0)}{q(x_0)}$. Thus the total probability of reaching (iii) after selecting (ii) is

$$\begin{aligned}\Pr[\text{reaching (iii)}\mid\text{(ii) selected}]&=\int_{\mathcal{S}}\frac{q(x)-p(x)}{q(x)}\frac{q(x)}{I[q]}d\mu(x)\\ &=\frac{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)}{I[q]}.\end{aligned}\tag{2.3}$$

Next suppose that $z\in\mathcal{L}$. The probability that z is returned from (i) is

$$\begin{aligned}\Pr[z\text{ returned from (i)}]=&\Pr[\text{(i) selected}]\,\Pr[z\text{ sampled in (i)}]\\ =&\frac{(I[p]-I[q])}{I[p]}\times\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}.\end{aligned}\tag{2.4}$$

The probability that z was returned from (ii.a) is

$$\begin{aligned}\Pr[z\text{ returned from (ii.a)}]=&\Pr[\text{(ii) selected}]\,\Pr[z\text{ sampled in (ii.a)}]\\ =&\frac{I[q]}{I[p]}\times\frac{q(z)d\mu(z)}{I[q]}=\frac{q(z)d\mu(z)}{I[p]}.\end{aligned}\tag{2.5}$$

Also, using Eq. (2.3), the probability that z was returned from (iii) is

$$\begin{aligned}
&\Pr[z \text{ returned from (iii)}]\\
=&\Pr[\text{(ii) selected}]\ \Pr[\text{reaching (iii)} \mid \text{(ii) selected}]\ \Pr[z \text{ sampled from } \mathcal{L} \text{ in (iii)}]\\
=&\frac{I[q]}{I[p]}\times\left(\frac{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)}{I[q]}\right)\times\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}\\
=&\frac{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)}{I[p]}\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}.
\end{aligned}\tag{2.6}$$

Finally, using Eqs. (2.4), (2.5), and (2.6), the probability of returning z is

$$\begin{aligned}
&\Pr[z \text{ returned from (i)}]+\Pr[z \text{ returned from (ii.a)}]+\Pr[z \text{ returned from (iii)}]\\
=&\frac{(I[p]-I[q])}{I[p]}\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}+\frac{q(z)d\mu(z)}{I[p]}\\
&+\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}\frac{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)}{I[p]}\\
=&\frac{(p(z)-q(z))d\mu(z)}{I[p]\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}\left(I[p]-I[q]+\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)\right)+\frac{q(z)d\mu(z)}{I[p]}\\
=&\frac{(p(z)-q(z))d\mu(z)}{I[p]\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}\left(\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)\right)+\frac{q(z)d\mu(z)}{I[p]}\\
=&\frac{p(z)d\mu(z)}{I[p]}.
\end{aligned}\tag{2.7}$$

Hence, by (2.2) and (2.7), whether $z\in\mathcal{S}$ or $z\in\mathcal{L}$, the probability of returning z is equal to $p(z)d\mu(z)/I[p]$. This completes the proof for Algorithm I. □

Proof for Algorithm II: For each $z\in\Omega$, show that the algorithm in Algorithm II returns z with probability $p(z)d\mu(z)/I[p]$. The algorithm consists of some number of cycles, each consisting of steps (i)-(iv), until a value z is accepted. We first calculate the probability that z is accepted within one of the cycles.

Suppose that $z\in\mathcal{S}$. Then z must be sampled in (i) and accepted in (iii). Thus, the probability of returning z in (iii) is

$$\begin{aligned}
\Pr[z \text{ returned from (iii)}]=&\Pr[z \text{ sampled in (i)}]\ \Pr[z \text{ accepted in (iii)}]\\
=&\frac{q(z)d\mu(z)}{I[q]}\times\frac{p(z)}{q(z)}=\frac{p(z)d\mu(z)}{I[q]}.
\end{aligned}\tag{2.8}$$

Also note that for every $x_0\in\mathcal{S}$,which is chosen with probability $\frac{q(x_0)d\mu(x_0)}{I[q]}$, the probability that it is not accepted in (iii) is $\frac{q(x_0)-p(x_0)}{q(x_0)}$. Thus the total probability of not returning an

element of $\mathcal{S}$ in (iii), which is the same as the probability of reaching (iv), is

$$\Pr[\text{reaching (iv)}]=\int_{\mathcal{S}}\frac{q(x)-p(x)}{q(x)}\frac{q(x)}{I[q]}d\mu(x)=\int_{\mathcal{S}}\frac{q(x)-p(x)}{I[q]}d\mu(x). \tag{2.9}$$

Next suppose that $z\in\mathcal{L}$. The probability that z is accepted in (ii) is

$$\Pr[z \text{ returned from (ii)}]=\frac{q(z)d\mu(z)}{I[q]}. \tag{2.10}$$

For z to be returned from (iv.a), the algorithm must reach (iv), then go to (iv.a) and then select z in (iv.a). This has probability

$$\begin{aligned}
&\Pr[z \text{ returned from (iv.a)}]\\
=&\Pr[\text{reach (iv)}]\ \Pr[\text{go to (iv.a)}]\ \Pr[\text{z sampled in (iv.a)}]\\
=&\left(\int_{\mathcal{S}}\frac{q(x)-p(x)}{I[q]}d\mu(x)\right)\times\frac{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}{\int_{\mathcal{S}}(q(x)-p(x))d\mu(x)}\times\frac{(p(z)-q(z))d\mu(z)}{\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)}\\
=&\frac{(p(z)-q(z))d\mu(z)}{I[q]}.
\end{aligned} \tag{2.11}$$

Now using Eqs. (2.10) and (2.11), the probability of returning z in a cycle is

$$\begin{aligned}
\Pr[z \text{ returned}]=&\Pr[z \text{ returned from (ii)}]+\Pr[z \text{ returned from (iv.a)}]\\
=&\frac{q(z)d\mu(z)}{I[q]}+\frac{(p(z)-q(z))d\mu(z)}{I[q]}=\frac{p(z)d\mu(z)}{I[q]}.
\end{aligned} \tag{2.12}$$

Eqs. (2.8) and (2.12) imply that, whether $z\in\mathcal{S}$ or $z\in\mathcal{L}$, the probability that z is returned in a cycle is $p(z)d\mu(z)/I[q]$. Integrating over all the samples in Ω, we deduce that the probability that a sample is returned in a cycle is $I[p]/I[q]$. Consequently, probability that no sample point is returned in a cycle is $1-I[p]/I[q]$. Because the cycle is repeated until a sample point is returned, we conclude that the probability that the algorithm returns z is equal to

$$\sum_{k=1}^{\infty}(1-\frac{I[p]}{I[q]})^{k-1}\frac{p(z)d\mu(z)}{I[q]}=\frac{p(z)d\mu(z)}{I[p]}.$$

This completes the proof for Algorithm II. □

Note that the efficiency of Algorithm II is nominally the same as acceptance rejection, i.e. the probability of a rejection is $1-I[p]/I[q]$. Actually it can be significantly better because $I[q]$ can be smaller, since $q<p$ is allowed. Also, note that if $I[p]=I[q]$, then Algorithms I and II are the same.

3 Comparison of Reduced Rejection and other sampling methods

One of the important features of Reduced Rejection sampling is that it requires no preprocessing time. This is particularly useful for *dynamic simulation*; i.e., simulation in which the probability distribution function $p(x)$ may change after each sample (see Section 5 for an example from plasma physics). For dynamic simulation, fast discrete sampling methods such as Marsaglia's Table method or the Alias method, are not suitable as they require preprocessing time after each change in $p(x)$. Although, the Acceptance-Rejection method requires no preprocessing time and can be used for dynamic simulation, it may require changes in $q(x)$ if $p(x)$ changes, which is usually not difficult, and it becomes very inefficient when the ratio of the area under function $p(x)$ to the area under proposal function $q(x)$ is small. Moreover, Adaptive Rejection sampling is not efficient, because the process of adapting $q(x)$ to $p(x)$ starts over whenever $p(x)$ changes.

The Reduced Rejection sampling method can be thought as an extension of the Acceptance-Rejection method. In particular when the proposal function $q(x)$ encloses $p(x)$ (i.e., $q(x)\geq p(x)$ for all $x\in\Omega$ so that $\mathcal{L}=\emptyset$) the Reduced Rejection sampling method reduces to Acceptance-Rejection method. The advantage of Reduced Rejection sampling over Acceptance-Rejection method is that the proposal function $q(x)$ does not need to enclose function $p(x)$; i.e., it allows $q(x)<p(x)$ for some x. This is very useful in dynamic simulation as it can accommodate changes in $p(x)$ without requiring changes in $q(x)$. Moreover, Reduced Rejection sampling may result in less unwanted samples than Acceptance-Rejection does, especially if $p(x)$ has singularities or is highly peaked.

There are several challenges in implementing the Reduced Rejection sampling method. The main challenge is the need to sample from set Ω according to $q(x)$ and from set $\mathcal{L}$ according to $p(x)-q(x)$, which can be performed by various sampling methods. Another challenge in using Reduced Rejection sampling is the need to know the values of $I[q]$, $I[p]$ and $\int_{\mathcal{L}}(p(x)-q(x))d\mu(x)$ (but note that the last value is only for Algorithm II). In many situations, these values are readily available or can be calculated during the simulation.

4 Example 1: Reduced Rejection sampling for a random variable with singular density

In this section, Reduced Rejection sampling method is applied to a simple problem. Let $\Omega=(0,1)$ and sample according to

$$p(x)=\frac{1}{\sqrt{x}}+\frac{1}{\sqrt[5]{1-x}} \tag{4.1}$$

which has singularities at 0 and 1. Using Inverse Transform sampling, it is easy to sample according to $1/\sqrt{x}$ or $1/\sqrt[5]{1-x}$, but Inverse Transform cannot be easily applied to (4.1)

as it requires finding the root of a eighth degree polynomial. We apply Reduced Rejection sampling to this problem by setting $q(x)=1/\sqrt{x}$. Observe that $\mathcal{L}=\Omega=(0,1)$. As mentioned earlier, Inverse Transform sampling is easily used to sample according to $q(x)$ and according to $p(x)-q(x)$. The Reduced Rejection sampling is very fast and yields no unwanted sample points. This example is equivalent to sampling from a mixture and can be extended to sampling from a probability density $p(x)$ that is a sum $p=p_1+p_2+\cdots+p_n$, if there is a method for sampling from each p_k separately and the integrals $I[p_k]$ are all known.

5 Example 2: Reduced Rejection sampling for a stochastic process with fluctuating and singular rates

In this section, we apply Reduced Rejection sampling to an idealized problem motivated by plasma physics. As discussed in Subsection 5.5, the unique features of this problem makes other sampling methods inefficient to use.

5.1 Statement of the stochastic process and the simulation algorithm

The example presented here is a simplified version of simulation for recombination by impact of two electrons with an ion, in which one of the electrons is absorbed into the atom and the other electron is scattered. For incident electron energies E_1 and E_2, the recombination rate is proportional to $(E_1E_2)^{-1/2}$ [15, 21], which can become singular if electrons of low energy are present. This is an obstacle to kinetic simulation of recombination by electron impact in a plasma.

Our goal is to simulate the evolution of the following system: Consider N particles labeled $1,\cdots,N$. To each particle i we associate a number $x_i\in(0,1)$, called the state of particle i (and corresponding to electron energy in the recombination problem). Occasionally, where it does not cause confusion, we use x_i to refer to particle i. We refer to the set $\Gamma=\{x_1,\cdots,x_n\}$ as the configuration of the system. For every pair of states x_i and x_j, $T_{i,j}$ is a random variable with an exponential distribution with parameter $(x_ix_j)^\alpha$, in which α is a fixed constant between 0 and 1. $T_{i,j}$ is the time for interaction between particle i and j which randomly occurs with rate $1/(x_ix_j)^\alpha$. After an interaction occurs, say for the pair $\{k,l\}$, the values of states x_k and x_l are replaced by new values x_k' and x_l'; consequently, the distribution of $T_{i,j}$ changes if either of i and j is equal to k or l.

We will consider a simple updating mechanisms for the states after each interaction. In the simulations presented below, the updated values of x_k' and x_l' are chosen independently and uniformly at random from $(0,1)$, without dependence on x_k and x_l. This choice has been made for simplicity and because the stationary distribution can be calculated for this choice (see Section 5.2), but we expect that Reduced Rejection sampling would work equally well for more complex interaction rules. Indeed the Algorithm 5.1 described below and the more detailed algorithm presented in Section 5.4 do not depend

on the interaction rules.

First we make some notation and observations. Set $s_i = 1/x_i^\alpha$ and $s = \sum_i s_i$. Let $T(\lambda)$ denote an exponential random variable with parameter λ (with rate $1/\lambda$); then $T(\lambda) = \mu T(\mu\lambda)$ for any scalar μ. We will use

$$\frac{s_i}{s}\frac{s_j}{s}T(1/s^2) = T(1/(s_i s_j))$$

in the following algorithm, which is a variant of the Kinetic Monte Carlo (KMC) algorithm (also known as the Residence-Time Algorithm or the N-Fold Way or the Bortz-Kalos-Lebowitz (BKL) algorithm [2]), that simulates the system described above. This algorithm chooses interactions, by choosing two particles separately out of the N number of particles, rather than choosing a pair of particles out of the N^2 number of pairs.

Algorithm 5.1.

1. Start from $t=0$.
2. Choose time Δt by sampling from an exponential distribution with rate s^2.
3. Choose index k with probability s_k/s.
4. Choose index l with probability s_l/s.
5. At time $t+\Delta t$ interaction between particles k and l occurs.
6. Update states x_k and x_l according to the updating mechanism and update the value of s.
7. Set $t = t+\Delta t$ and start over from 2.

We use Reduced Rejection sampling in Subsection 5.5 to perform steps 3 and 4 in the above algorithm. We also explain why other methods of sampling would be inefficient in these circumstances. To verify that our simulation is working properly, we perform the following test.

Let $g(x_1,\cdots,x_N)$ be a real-valued function on configuration space, with expectation $E[g]$ of g over configurations of the system. For a simple interaction rule and some functions g we can find the value of $E[g]$ analytically, as shown in Subsection 5.2. Consequently, the difference between the numerical and analytic results provides a measure of the accuracy of the simulation as discussed at the end of Subsection 5.5.

5.2 Theoretical results

Think of the system's evolution as a random walk over the configurations of the system. Suppose the updating process is that if states x_k and x_l interact, then states x_k' and x_l' are chosen, independently, uniformly at random from $(0,1)$. In this section, we find the stationary distribution for this random walk and the value of $E[g]$ for two functions g.

Let $P(\Gamma'|\Gamma)$ denote the probability of going from configuration Γ to configuration Γ', and $P(x_k',x_l'|x_k,x_l)$ as the probability of going from values x_k,x_l to values x_k',x_l' in an interaction. These satisfy

$$P(x_k',x_l'|x_k,x_l) = P(x_k,x_l|x_k',x_l'), \tag{5.1}$$

that is, the probability of getting x_k' and x_l' after x_k and x_l interact, is the same as probability of getting x_k and x_l after x_k' and x_l' interact. Furthermore, for this updating mechanism, the random walk is completely mixing; that is, it can go from any configuration to any other configuration (If for example the updating mechanism had additional constraints, such as $x_k'+x_l'=x_k+x_l$, then we would not have a mixing random walk since we could reach only those configurations that have the same sum of states as the starting configuration).

For every configuration $\Gamma=\{x_1\cdots,x_N\}$, set

$$\pi_\Gamma := \frac{(x_1\cdots x_N)^\alpha}{Z}\left(\sum_{i,j}\frac{1}{(x_ix_j)^\alpha}\right),$$

where Z is the normalizing constant so that $\sum_\Gamma \pi_\Gamma = 1$.

Suppose the current configuration of the system is $\Gamma=\{x_1,\cdots,x_N\}$. According to steps 3 and 4 in Algorithm 5.1, the probability of interaction occurring between states x_k and x_l is proportional to $s_ks_l = 1/(x_kx_l)^\alpha$. Let $\Gamma'=\{x_k',x_l'\}\cup\Gamma\setminus\{x_k,x_l\}$. Then

$$P(\Gamma'|\Gamma)=\frac{1/(x_kx_l)^\alpha}{\sum_{i,j}1/(x_ix_j)^\alpha}P(x_k',x_l'|x_k,x_l).$$

For the ease of explanation, relabel the states of Γ' so that $\Gamma'=\{x_1',\cdots,x_N'\}$ where $x_i'=x_i$ for $i\neq k,l$. Similarly,

$$\pi_{\Gamma'}=\frac{(x_1'\cdots x_N')^\alpha}{Z}\left(\sum_{i,j}\frac{1}{(x_i'x_j')^\alpha}\right),\quad\text{and}\quad P(\Gamma|\Gamma')=\frac{1/(x_k'x_l')^\alpha}{\sum_{i,j}1/(x_i'x_j')^\alpha}P(x_k,x_l|x_k',x_l').$$

Because of (5.1) and since $x_i'=x_i$, for $i\neq k,l$, it is straightforward to verify the detailed balance equation

$$\pi_\Gamma P(\Gamma'|\Gamma)=\pi_{\Gamma'}P(\Gamma|\Gamma').$$

Therefore, π_Γ is the (unique) stationary distribution of the random walk. Since the updating mechanism is completely mixing, the normalizing constant Z for distribution π_Γ is the integral

$$Z=\int_{(0,1)^N}(x_1\cdots x_N)^\alpha\left(\sum_{i,j}\frac{1}{(x_ix_j)^\alpha}\right)dx_1\cdots dx_N=\frac{\binom{N}{2}}{(\alpha+1)^{N-2}}.$$

Hence for any function $g(x_1,\cdots,x_N)$,

$$E[g]=\frac{(\alpha+1)^{N-2}}{\binom{N}{2}}\int_{(0,1)^N}g(x_1,\cdots,x_N)(x_1\cdots x_N)^\alpha\left(\sum_{i,j}\frac{1}{(x_ix_j)^\alpha}\right)dx_1\cdots dx_N.$$

Some tedious algebra leads to the following proposition:

Proposition 5.1. Using the above notations and assumptions:

a) $E[g]=\frac{\alpha+1}{\alpha+2}(N-2)+1$ when $g(x_1,\cdots,x_N)=x_1+\cdots+x_N$.

b) $E[g]=\frac{\alpha+1}{\alpha+3}(N-2)+\frac{2}{3}$ when $g(x_1,\cdots,x_N)=x_1^2+\cdots+x_N^2$.

5.3 Simulation issues

In this section we make some remarks about the challenges involved in simulating this system.

The main challenge of sampling in this dynamic simulation is that the s_i's are changing after each interaction. Consequently, the sampling method should require small or zero preprocessing time. For this reason, discrete sampling methods such as Marsaglia's Table method or the Alias method are not very efficient for this problem.

Next consider using Acceptance-Rejection method based on uniform sampling from 1 to n for the proposal distribution (i.e., q constant). As mentioned earlier, the changing distribution property of the problem is not very detrimental for Acceptance-Rejection. On the other hand, the singularity in the rates at $x_i=0$ can lead to a large constant for q, for which there will be many rejected samples, so that the method is inefficient. Moreover, there seems to be no other clear choice for the proposal distribution q other than a constant. Note that the sampling is from a discrete set of probabilities s_i/s with little control over their values; for example the s_i's are not monotonically ordered. This is quite different from sampling a single random variable from the density $p(x)=x^{-\alpha}$.

5.4 Use of the Reduced Rejection algorithm

In this section we explain how to use Reduced Rejection Sampling to perform steps 3 and 4 in Algorithm 5.1. Reduced Rejection sampling can be readily used in this dynamic simulation. Even though the values of the s_i's change after each interaction, they do not change drastically; in each interaction at most two of the s_i's change. Starting at time 0, we set $q_i=p_i=s_i$. After each interaction, we update the values of p_i's to $p_i=s_i$, but do not change the values of q_i's. Note that we can easily update the value of $I[p]$ after each interaction and keep track of set $\mathcal{L}=\{i:p_i>q_i\}$ by comparing the updated values of p_i's to their corresponding values of q_i's. Moreover, the size of set $\mathcal{L}$ changes by at most 2 after each interaction (but it can also decrease after some interactions).

We use Marsaglia's Table method to sample according to q_i's. Since we do not update q_i's after each interaction, the preprocessing time in Marsaglia's Table method is only required for the first sampling and not for the subsequent samplings. To sample from set $\mathcal{L}$ according to p_i-q_i, we use Acceptance-Rejection with uniform distribution for the proposal distribution. As long as the size of set $\mathcal{L}$ is not too big, the sampling from $\mathcal{L}$ is not very time consuming. To prevent $\mathcal{L}$ from getting too large, we reset the values of q_i's to $q_i=p_i=s_i$, which sets $\mathcal{L}$ to be empty, whenever the size of $\mathcal{L}$ exceeds a predetermined number M.

The size of M is important for the performance of the algorithm. If M is too small, then there are many updates of the q_i's, each of which requires preprocessing time for Marsaglia's Table method. On the other hand, if M is too big, then $\mathcal{L}$ is large and costly to sample from by Acceptance-Rejection. Our computational experience shows that setting M equal to a multiple of $\sqrt{N}$ is a good choice. It might be better for the reinitialization criterion to be based on the efficiency of the sampling from $\mathcal{L}$ (i.e., the fraction of rejected samples when using acceptance rejection on $\mathcal{L}$), rather than the size of $\mathcal{L}$.

5.5 Numerical result

We simulated the evolution of the system under the conditions outlined in Subsection 5.2 with $N=10^4$, and $\alpha=0.5$. We start with a random configuration at time $t=0$. The simulation is based the Reduced Rejection sampling method, using Marsaglia's Table method and the Acceptance-Rejection method as described above. After each interaction, we evaluate function $g(x_1,\cdots,x_N)=x_1+\cdots+x_N$ and take the average to get an estimate for $E[g]$. Each result is produced by taking an average of five independent runs. Fig. 3 compares the results for $E[g]$ from Reduced Rejection sampling with those from the Acceptance-Rejection method. The results of Fig. 3 show excellent agreement between the values of $E[g]$ as a function of the number of interactions from the two methods, which provides a validity check for Reduced Rejection sampling.

The advantage of Reduced Rejection sampling is demonstrated in Fig. 4 which shows a log-log plot of the processing time, as a function of the number of interactions, for Reduced Rejection sampling and Acceptance-Rejection. The results show that Reduced Rejection sampling is much faster than the Acceptance-Rejection method. In fact, for n interactions, the computational time scales like $\mathcal{O}(n)$ for Reduced Rejection sampling, and like $\mathcal{O}(n^{3/2})$ for Acceptance-Rejection, in the range $10^4\leq n\leq 10^6$. For small values of n, the initial pre-processing step of Marsaglia's Table method dominates the computational time. For $n>10^4$, however, the pre-processing time (including the multiple preprocessing steps due to reinitialization) is not a significant part of the computational time. The average number of reinitializations for Reduced Rejection sampling is $(0,0,0,3.7,53.1)$ for $n=(10^2,10^3,10^4,10^5,10^6)$, respectively. Another interesting advantage of the Reduced Rejection sampling is that the variance of the processing time for independent runs is much smaller in the Reduced Rejection sampling than it is in the Acceptance-Rejection method.

6 Example 3: Stochastic simulation of chemical kinetics

In this section we describe how we can use Reduced Rejection in the context of stochastic chemical kinetics. Stochastic simulation in chemical kinetics is a Monte Carlo procedure to numerically simulate the time evolution of a well-stirred chemically reacting system. The first Stochastic Simulation Algorithm, called the Direct Method, was presented in [10]. The Direct Method is computationally expensive and there have been

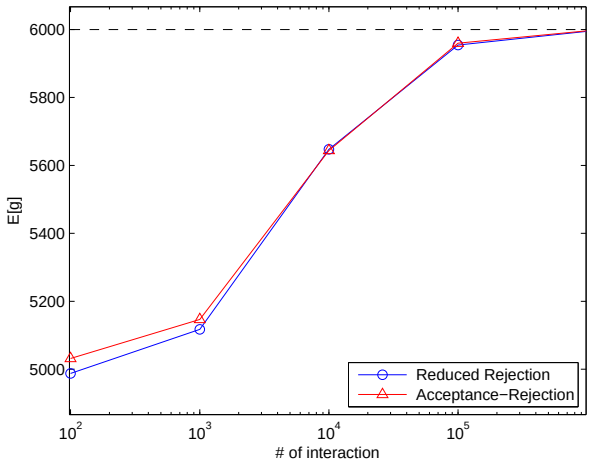

Figure 3: Theoretical (dashed line) and estimated values (solid lines) of $E[g]$ using different number of interactions. Here $g(x_1,\cdots,x_N)=x_1+\cdots+x_N$, $N=10^4$, and $\alpha=0.5$. Also $M=4000$ for the Reduced Rejection sampling. The theoretical value of $E[g]$ is 5999.8. The estimated value of $E[g]$ after 10^6 interactions using Reduced Rejection sampling and Acceptance-Rejection methods were, respectively, 5994.59 and 5996.35. The reported result is the average of 5 independent runs.

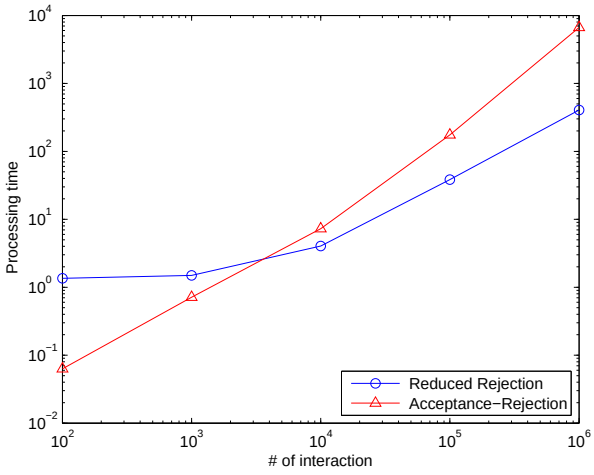

Figure 4: Loglog plot of the processing time for Acceptance-Rejection and Reduced Rejection sampling. Here $g(x_1,\cdots,x_N)=x_1+\cdots+x_N$, $N=10^4$ and $\alpha=0.5$. Also $M=4000$ for the Reduced Rejection sampling. The reported processing time is the average of 5 independent runs.

many adaptations of this algorithm to achieve greater speed in simulation. The first-reaction method, also in [10], is an equivalent formulation of the Direct Method. The next-reaction method [6] is an improvement over the first-reaction method, using a binary-tree structure to store the reaction times. The Modified Direct Method [4] and Sorting Direct

Method [14] speed up the Direct Method by indexing the reactions in such a way that reactions with larger propensity function tend to have a lower index value. Recently, some new Stochastic Simulation Algorithms, called Partial-Propensity methods, were introduced that work only for elementary chemical reactions (i.e. reactions with at most two different reactants) (see [16–18]). Nevertheless, note that it is possible to decompose any non-elementary reaction into combination of elementary reactions. There are also approximate Stochastic Simulation Algorithms, such as Tau-Leaping and Slow-Scale, that provide better computational efficiency in exchange for sacrificing some of the exactness in the Direct Method (see [9] and the references therein for more details).

Next we give a brief review of stochastic simulation in chemical kinetics. An excellent reference with more detailed explanation is [9]. Using the same notation and terminology as in [9], consider a well-stirred system of molecules of N chemical species $\{S_1,\cdots,S_N\}$, which interact through M chemical reactions $\{R_1,\cdots,R_M\}$. Let $X_i(t)$ denote the number of molecules of species S_i in the system at time t. The goal is to estimate the state vector $\mathbf{X}(t)\equiv(X_1(t),\cdots,X_N(t))$ given the system is initially in state $\mathbf{X}(0)=\mathbf{x}_0$.

Similar to Section 5, when the system is in state $\mathbf{x}$, the time for reaction R_j to occur is given by an exponential distribution whose rate is the propensity function $a_j(\mathbf{x})$. When reaction R_j occurs, the state of the system changes from $\mathbf{x}$ to $\mathbf{x}+(v_{1j},\cdots,v_{Nj})$, where v_{ij} is the change in the number of S_i molecules when one reaction R_j occurs.

Estimating the propensity functions in general is not an easy task. As noted, the value of the propensity functions depend on the state of the system. For example, if R_i and R_j are, respectively, the unimolecular reaction $S_1\to product(s)$ and bimolecular reaction $S_1+S_2\to product(s)$, then $a_i(\mathbf{x})=c_ix_1$ and $a_j(\mathbf{x})=c_jx_1x_2$ for some constants c_i and c_j. Therefore, the propensity functions of the reactions are changing throughout the simulation. Moreover, if for some chemical species the magnitude of their population differ drastically from others, we expect the value of propensity functions to be very non-uniform.

For every state $\mathbf{x}$, define

$$a(\mathbf{x})=\sum_{i=1}^{M}a_i(\mathbf{x}).$$

To simulate the chemical kinetics of the system the following algorithm is used, which resembles Algorithm 5.1 in Section 5.

Algorithm 6.1.

1. Start from time $t=0$ and state $\mathbf{x}=\mathbf{x}_0$.
2. Choose time Δt by sampling from an exponential distribution with rate $a(\mathbf{x})$.
3. Choose index k with probability $a_k(\mathbf{x})/a(\mathbf{x})$.
4. At time $t+\Delta t$ reaction R_k occurs.
5. Update time $t=t+\Delta t$, state $\mathbf{x}=\mathbf{x}+(v_{1k},\cdots,v_{Nk})$ and start over from 2.

In the original Direct Method [10], step 3 in the above Algorithm 6.1 is performed by choosing number r uniformly at random in the unit interval and setting

$$k = \text{ the smallest integer satisfying } \sum_{i=1}^{k} a_i(\mathbf{x}) > r a(\mathbf{x}). \tag{6.1}$$

However, when we have many reactions with a wide range of propensity function values presented in the system, a scenario that is very common in biological models, the above procedure of using partial sums becomes computationally expensive. As noted earlier, some methods, such as the Modified Direct Method [4] and Sorting Direct Method [14], index the reactions in a smart way so that they can save on the average number of terms summed in Eq. (6.1) and consequently achieve computational efficiency.

We propose a different approach to performing step 3 in Algorithm 6.1 using the Acceptance-Rejection or Reduced Rejection method. The approach is very similar to what was done in Section 5. To be specific, we can use Acceptance-Rejection for step 3 in the following way: let

$$\bar{a}(\mathbf{x}) = \max_i a_i(\mathbf{x}).$$

Until an index is accepted, select index k uniformly at random from $\{1,\cdots,N\}$ and accept it with probability $a_k(\mathbf{x})/\bar{a}(\mathbf{x})$; otherwise, discard k and repeat. When an index is accepted step 3 in Algorithm 6.1 is completed. Typically for chemical reactions R_j, most of v_{ij}'s are zero; therefore, we can efficiently update the value of $\bar{a}(\mathbf{x})$ at each iteration of Algorithm 6.1.

However, as in Section 5, if the values of $a_i(\mathbf{x})$'s are very non-uniform (for example, when the population of some chemical species differ drastically from that of other species in the system) the Acceptance-Rejection method becomes inefficient due to rejection of many samples. In these circumstances, the Reduced Rejection algorithm can be readily used in a very similar way as it was used in Section 5. We expect that the use of the Reduced Rejection algorithm in these circumstances would greatly improve the computational efficiency of the exact Stochastic Simulation Algorithms.

7 Conclusions and future directions

In this paper we introduce a new Reduced Rejection sampling method that can be used to generate independent samples for a discrete or continuous random variable. The strength of this algorithm is most evident for applications in which Acceptance-Rejection method is inefficient; namely, the probability distribution of the random variable is highly peaked in certain regions or has singularities. It is also useful when the probabilities are fluctuating, so that discrete methods that requiring preprocessing are inefficient. In particular, the Reduced Rejection sampling method is expected to perform well on kinetic simulation of electron-impact recombination in a plasma, which is difficult to simulate by other methods.

The preliminary examples in this paper are meant to illustrate these advantages of the Reduced Rejection sampling method. They provide evidence of improvement in computation time using the Reduced Rejection sampling versus Acceptance-Rejection method. These examples also provide some insights on implementation of the method.

One possible direction for future research is the nested use of Reduced Rejection sampling methods. For the most difficult step – sampling from $\mathcal{L}$ according to $p(x)-q(x)$ – we propose to apply the Reduced Rejection sampling method again using a new proposal function. In essence, this would use one Reduced Rejection sampling method inside another Reduced Rejection sampling method.

Acknowledgments

The research of F. Barekat and R. Caflisch is supported by DOE grant DE-FG02-05ER25710.

Appendix A: Flow charts of the Reduced Rejection algorithm

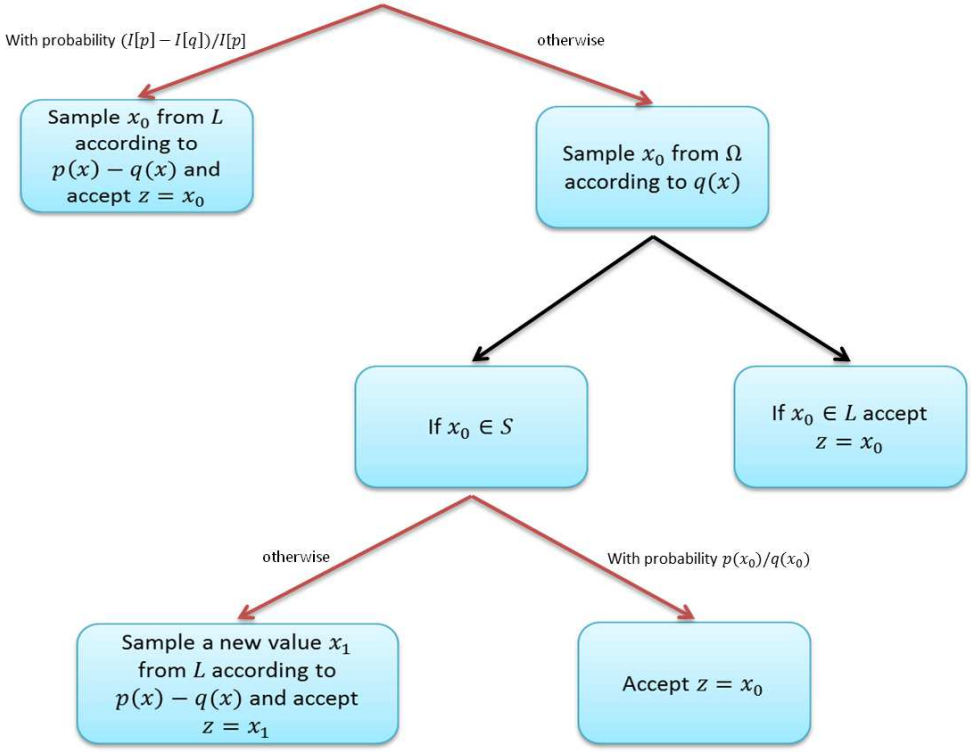

Figure 5: Flow chart of Algorithm I of the Reduced Rejection sampling method attributing to the case $I[p]\geq I[q]$.

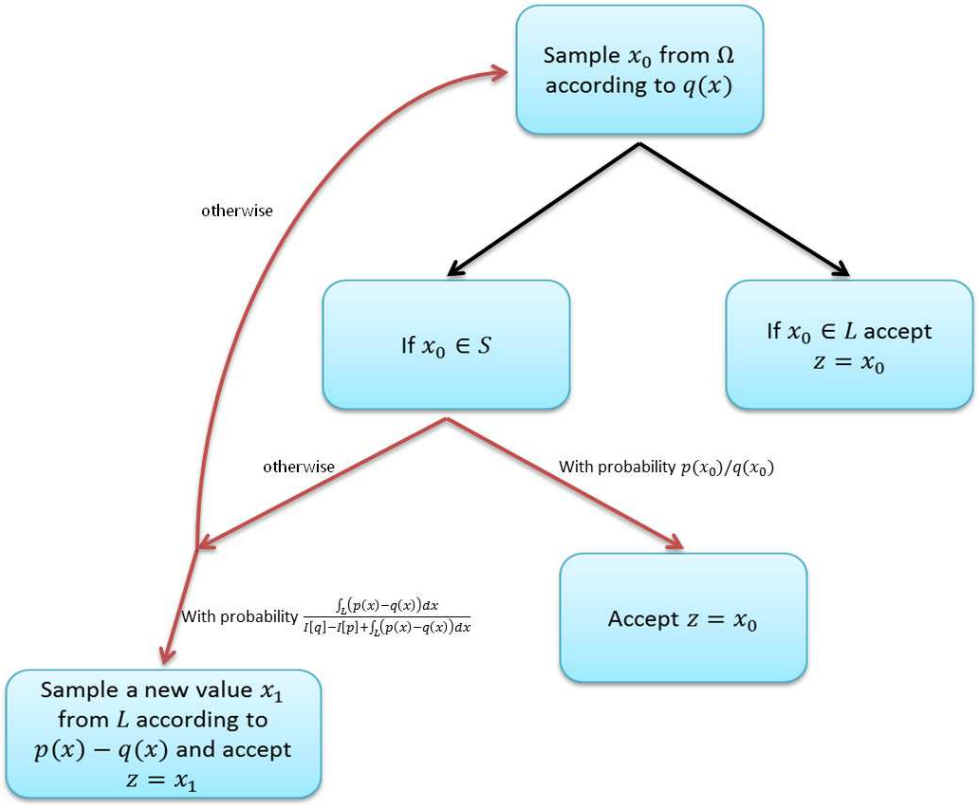

Figure 6: Flow chart of Algorithm II of the Reduced Rejection sampling method attributing to the case $I[p] < I[q]$.

References

[1] G.A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Claredon, Oxford (1994).

[2] A.B. Bortz, M.H. Kalos, and J.L. Lebowitz, A new algorithm for Monte Carlo simulation of Ising spin systems, J. Comp. Phys., 17 (1975), 10–18.

[3] R.E. Caflisch, Monte Carlo and quasi-Monte Carlo methods, Acta Numerica, 7 (1998), 1–49.

[4] Y. Cao, H. Li, and L.R. Petzold, Efficient formulation of the stochastic simulation algorithm for chemically reacting systems, J. Chem. Phys., 121 (2004), 4059–4067.

[5] I. Deak, An economical method for random number generation and a normal generator, Computing, 27 (1981), 113–121.

[6] M.A. Gibson and J. Bruck, Exact stochastic simulation of chemical systems with many species and many channels, J. Phys. Chem., 105 (2000), 1876–1889.

[7] W.R. Gilks, N.G. Best, and K.K.C. Tan, Adaptive rejection metropolis sampling, Applied Statistics, 44 (1995), 455–472.

[8] W.R. Gilks and P. Wild, Adaptive rejection sampling for Gibbs sampling, Applied Statistics, 41 (1992), 337–348.

[9] D.T. Gillespie, Stochastic simulation of chemical kinetics, Annu. Rev. Phys. Chem., 58 (2007), 35–55.

[10] D.T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, J. Comput. Phys., 22 (1976), 403–434.
[11] G. Marsaglia, W.W. Tsang, and J. Wang, Fast generation of discrete random variables, Journal of Statistical Software, 11(3) (2004), 1–11.
[12] G. Marsaglia and W.W. Tsang, A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions, SIAM J. Sci. Stat. Comput., 5(2) (1984), 349–359.
[13] G. Marsaglia, Xorshift RNGs, Journal of Statistical Software, 8(14) (2003), 1–9.
[14] J.M. McCollum, G.D. Peterson, C.D. Cox, M.L. Simpson, and N.F. Samatova, The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior, Comput. Bio. Chem., 30 (2006), 39–49.
[15] J.T. Oxenius, Kinetic Theory of Particles and Photons, Springer-Verlag, Berlin (1986).
[16] R. Ramaswamy, N. Gonzalez-Segredo, and I.F. Sbalzarini, A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks, J. Chem. Phys., 130 (2009), 244104.
[17] R. Ramaswamy and I.F. Sbalzarini, A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks, J. Chem. Phys., 132 (2010), 044102.
[18] R. Ramaswamy and I.F. Sbalzarini, A partial-propensity formulation of the stochastic simulation algorithm for chemical reaction networks with delays, J. Chem. Phys., 134 (2011), 014106.
[19] M.D. Vose, A linear algorithm for generating random numbers with a given distribution, IEEE Transaction and Software Engineering, 17(9) (1991), 972–975.
[20] A.J. Walker, An efficient method for generating discrete random variables with general distributions, ACM TOMS, 3 (1977), 253–256.
[21] Y.B. Zeldovich and Y.P. Raizer, Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena, Dover, Mineola, NY (2002).