Markov chain simulation with fewer random samples

Dimitrios Milios, Stephen Gilmore

School of Informatics, University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB, UK

Abstract

We propose an accelerated CTMC simulation method that is exact in the sense that it produces all of the transitions involved. We call our method *Path Sampling Simulation* as it samples from the distribution of trajectories and the distribution of time given some particular trajectory. Sampling from the trajectory space rather than the transition space means that we need to generate fewer random numbers, which is an operation that is typically computationally expensive. Sampling from the time distribution involves approximating the exponential distributions that govern the sojourn times with a geometric distribution. A proper selection for the approximation parameters can ensure that the stochastic process simulated is almost identical to the simulation of the original Markov chain. Our approach does not depend on the properties of the system and it can be used as an alternative to more efficient approaches when those are not applicable.

1. Introduction

Continuous Time Markov Chains (CTMCs) have been used for many years for describing systems that exhibit stochastic behaviour. Stochastic simulation is a traditional approach for exploring the transient and steady-state properties of massive CTMCs, since it does not require an explicit representation of the state-space. There are models however, such as bio-chemical reaction networks, whose state-space is too large even for this kind of approach. The standard simulation method for CTMCs is known as the *direct method* (DM) [9]. In the case of very large models, it has a high computational cost because it simulates every possible transition happening. Several accelerated methods have been proposed that are either exact or approximate. Exact methods typically involve optimizations over the standard algorithm, such as the *next reaction method* [8], the *optimised DM* [5], the *logarithmic DM* [17] and *ER-leap* [19].

An approximate simulation method tries to skip some simulation events resulting in a significantly faster process when compared to exact methods. For example, τ*-leaping* [10] advances time by a pre-selected τ, during which many transitions may occur. Similarly in *R-leaping* [1], stochastic simulation was

accelerated by advancing by a predefined number of transition firings. *K-leap* [3] is an approach that also advances time by a specified number of events. All these methods assume that a single reaction transition causes only small changes to the state of the system. Other approaches such as [20] and [4] make use of the notion of time-scale separation. It is assumed that the model can be partitioned to two sub-systems: slow and fast. The behaviour of the fast sub-system is approximated, while only the slow sub-system is simulated. Such assumptions may not always hold for arbitrary models, meaning that either significant error is introduced or the approximate method fails to accelerate the simulation process.

The *Path Sampling Simulation* algorithm that we propose is a modification of the DM that can be characterised as almost exact, in the sense that it can be arbitrarily precise. In the case of the DM, each step requires sampling from two distributions: the state distribution and the time distribution, both conditioned on the current state. In a similar way, path sampling simulation involves sampling from the distribution of trajectories. This reduces the number of random samples generated, a fact that implies a faster simulation algorithm. The algorithm is still exact, since no transitions are skipped. The same approach is extended to sample from the time distribution given some particular trajectory. That is achieved by approximating the exponentially distributed sojourn times with a discrete random variable. Time discretization allows us to consider the time distribution as a discrete state Markov chain, and therefore employ the path sampling technique. This modification essentially renders our approach approximate, as part of the stochastic behaviour of the CTMC is suppressed. However, we have proved that our method in the limit converges to the solution of the original process, a fact that explains the term 'almost exact' used earlier in the paragraph. We show that an appropriate selection of the approximation parameters can result in a behaviour very close to the original CTMC, and in a reasonable speedup at the same time. Our implementation is based on the optimised DM (ODM), hence the ODM is used as a baseline for efficiency comparisons.

Our approach is related to *K-skip method I* in [2], or simply K-skip. While their strategy for sampling from the trajectory distribution is similar, sampling from the time distribution is different. In order to reduce the random samples that determine the sojourn times, they approximate the sum of k exponential random variables with a Gamma distribution, assuming that the exit rates are similar for subsequent states. This assumption is reasonable for many bio-chemical systems, however it may introduce errors for some models as we demonstrate in the experiments' section, while our approach can be generalised for arbitrary models. We have implemented K-skip following its description in the original paper, in order to produce some comparative results. The error parameter that we have used for K-skip is 0.01, which is the smallest value used in the original work. We also highlight some computational issues not considered in [2] that arise from the fact that one random number is used to produce an entire trajectory.

In Section 2, we introduce some concepts used throughout the paper. Sec-

tions 3 and 4 contain the theoretical details of our work. Some implementation issues are discussed in Section 5. Experimental results are presented in Section 6. Finally, we summarise the conclusions in Section 7.

2. Preliminaries

A CTMC can be represented as a triple $(S, Q, \boldsymbol{\pi}_0)$, where S is a finite set of states, $Q \in \mathbb{R}^{|S|\times|S|}$ is a generator matrix, and $\boldsymbol{\pi}_0$ is the initial probability distribution over S. Each $s \in S$ is associated with an exponentially distributed random variable $L_s \sim Exp(\lambda_s)$, where $\lambda_s = \sum_{s' \neq s} Q_{ss'}$ is the rate of exiting state s. The jump chain of a CTMC is discrete-time Markov with probability matrix P where:

$$P_{ss'} = \begin{cases} Q_{ss'}/Q_s, & s \neq s' \text{ and } Q_s \neq 0 \\ 0, & s \neq s' \text{ and } Q_s = 0 \\ 0, & s = s' \text{ and } Q_s \neq 0 \\ 1, & s = s' \text{ and } Q_s = 0 \end{cases}, \quad \text{where } Q_s = \sum_{s' \neq s} Q_{ss'} \tag{1}$$

A transition in a CTMC is associated with two random variables that depend on the current state s: X_s that determines the next state and L_s that determines the amount of time spent in s. The DM involves sampling from X_s and L_s to generate the next event. The distribution of X_s is multinomial conditional on s, and its probability mass function is given by the s-th row of the jump matrix P. We assume an ordering of states such as $s < s'$, if s corresponds to a row of the transition matrix with a smaller index than s'. If s_{k-1} is the state of the system after $k-1$ transitions, sampling from $X_{s_{k-1}}$ involves using a uniform sample $U \sim \mathcal{U}(0,1)$ and selecting the next state s_k with probability:

$$Pr(X_{s_{k-1}} = s_k) = Pr(a_{s_k} < U \leq b_{s_k}) \tag{2}$$

where b_{s_k} is the cumulative probability of state s_k given s_{k-1}, while a_{s_k} is the cumulative probability of state that precedes s_k in the ordering:

$$a_{s_k} = \sum_{s_k' < s_k} P_{s_{k-1}s_k'} \qquad \text{and} \qquad b_{s_k} = \sum_{s_k' \leq s_k} P_{s_{k-1}s_k'} \tag{3}$$

In order to sample from $L_{s_{k-1}}$, we have to draw a new uniform sample $U \sim \mathcal{U}(0,1)$ and calculate the time $t_{s_{k-1}}$ spent in s_{k-1} as follows:

$$t_{s_{k-1}} = -\frac{\ln(1-U)}{\lambda_{s_{k-1}}} \tag{4}$$

From a trajectory point of view, the random variables are different. Formally, a trajectory is a sequence of states and transitions. Assuming that there is at most one transition from any state to another, a trajectory is completely characterised by its sequence of states. Let $\mathcal{T}_k$ be a collection that stands for the family of trajectories of length k. Therefore, we define $X_{\mathcal{T}_k}$ as the variable

that represents the k-length trajectory distribution. This should not be confused with the state distribution after k transitions, since we also need to keep track of the entire state sequence. The trajectory distribution is only dependent on the initial state distribution. Given some particular trajectory, namely $s_{0:k}$, its duration is represented by the $L_{s_{0:k}}$ random variable. Ideally, we would like to use a single uniform sample for each to determine the state history and the time of the system after k transitions. Exact stochastic simulation algorithms actually sample from those distributions indirectly by generating one state at each event. In the sections that follow, we discuss how we can directly sample from $X_{\mathcal{T}_k}$ and $L_{s_{0:k}}$.

3. Sampling from the Trajectory Distribution

The sampling from the trajectory distribution discussed in this section can be applied to both discrete and continuous time processes. Without loss of generality, we can assume that there is one initial state in some Markov chain. This will be the root of a tree whose paths represent all the possible trajectories. Each path of a tree with k levels corresponds to a sequence of $k+1$ states or k transitions. Then, the probability of a trajectory can be defined as the product of the transitions involved:

$$Pr(X_{\mathcal{T}_k} = s_{0:k}) = \prod_{n=1}^{k} P_{s_{n-1}s_n} \tag{5}$$

In fact, $X_{\mathcal{T}_k}$ follows a multinomial distribution with $|\mathcal{T}_k|$ parameters. Sampling from the trajectory distribution requires us to compute its cumulative distribution function, which means that we have to define an ordering of the possible trajectories.

Definition 1 (Ordering of Trajectories)**.** *Given an ordering of states, we define an ordering of trajectories such as* $s_{0:k} < s_{0:k}{}'$ *if one of the following holds:*

i. $s_{0:k-1} < s_{0:k-1}{}'$ *or*

ii. $s_{0:k-1} = s_{0:k-1}{}'$ *and* $s_k < s_k{}'$

Therefore, we can calculate the cumulative probabilities for the trajectories. Given a uniform random variable $U \sim \mathcal{U}(0,1)$, we can choose directly a sample from the trajectory space. The relationship between U and $X_{\mathcal{T}_k}$ is shown in the following equation:

$$Pr(X_{\mathcal{T}_k} = s_{0:k}) = Pr(a_{s_{0:k}} < U \leq b_{s_{0:k}}) \tag{6}$$

The term $b_{s_{0:k}}$ is defined as the cumulative probability of the $s_{0:k}$ trajectory. In the same way, $a_{s_{0:k}}$ will be the cumulative probability of the path that precedes

$s_{0:k}$ according to the ordering. More formally:

$$\begin{aligned} a_{s_{0:k}} &= \sum_{s_{0:k}{}' < s_{0:k}} Pr(s_{0:k}{}') \\ b_{s_{0:k}} &= \sum_{s_{0:k}{}' \leq s_{0:k}} Pr(s_{0:k}{}') = a_{s_{0:k}} + Pr(s_{0:k}) \end{aligned} \tag{7}$$

Although sampling from the trajectory distribution is well-defined, it cannot be practically applied in its current form. The number of possible paths grows exponentially as the number of transitions increases, a fact that renders Equations (5) and (7) computationally expensive. What we can do instead, is to draw a sample from $U \sim \mathcal{U}(0,1)$ that determines the trajectory, and recursively generate the transitions involved. A recursive definition for the cumulative path probabilities would be useful for this task. Using Definition 1, the cumulative probability of the path that precedes $s_{0:k}$ can also be written recursively as follows:

$$a_{s_{0:k}} = Pr(s_{0:k-1}) \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} + a_{s_{0:k-1}} \tag{8}$$

Since the uniformly distributed sample U determines the entire k-length trajectory, it follows that it also determines all of the k transitions involved. In the DM however, the sequence of the transitions would have been determined by a sequence of uniform samples $U_n \sim \mathcal{U}(0,1)$, with $0 \leq n \leq k$. Thus, the sequence U_n is equivalent to the sample U for the trajectory. We shall next try to define the last sample U_k in terms of U, which gives rise to the following theorem:

Theorem 1. *If $U \sim \mathcal{U}(0,1)$ is used to draw a path sample $s_{0:k}$, then s_k is determined by:*

$$U_k = \frac{U - a_{s_{0:k-1}}}{b_{s_{0:k-1}} - a_{s_{0:k-1}}} \tag{9}$$

Proof. We have to show that $a_{s_k} < U_k \leq b_{s_k}$, which means that U_k will select the state s_k, according to Equation (2). Since $s_{0:k}$ was selected, Equation (6) implies:

$$\begin{aligned} a_{s_{0:k}} < U \leq b_{s_{0:k}} \Leftrightarrow & \\ & U > Pr(s_{0:k-1}) \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} + a_{s_{0:k-1}} \\ \text{and} \quad & U \leq Pr(s_{0:k-1}) \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} + a_{s_{0:k-1}} + Pr(s_{0:k}) \end{aligned}$$

We subtract $a_{s_{0:k-1}}$ from all terms, and divide everything by $Pr(s_{0:k-1})$:

$$\begin{aligned} Pr(s_{0:k-1}) \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} < U - a_{s_{0:k-1}} \leq Pr(s_{0:k-1}) \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} + Pr(s_{0:k}) & \\ \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} < \frac{U - a_{s_{0:k-1}}}{Pr(s_{0:k-1})} \leq \sum_{s_k{}' < s_k} P_{s_{k-1} s_k{}'} + \frac{Pr(s_{0:k})}{Pr(s_{0:k-1})} & \end{aligned}$$

We substitute $\frac{Pr(s_{0:k})}{Pr(s_{0:k-1})}$ with $P_{s_{k-1}s_k}$ and $Pr(s_{0:k-1})$ with $b_{s_{0:k-1}} - a_{s_{0:k-1}}$:

$$\sum_{s_k{}'<s_k} P_{s_{k-1}s_k{}'} < \frac{U - a_{s_{0:k-1}}}{b_{s_{0:k-1}} - a_{s_{0:k-1}}} \leq \sum_{s_k{}'<s_k} P_{s_{k-1}s_k{}'} + P_{s_{k-1}s_k}$$
$$\sum_{s_k{}'<s_k} P_{s_{k-1}s_k{}'} < \frac{U - a_{s_{0:k-1}}}{b_{s_{0:k-1}} - a_{s_{0:k-1}}} \leq \sum_{s_k{}'\leq s_k} P_{s_{k-1}s_k{}'}$$

which eventually yields:

$$a_{s_k} < U_k \leq b_{s_k}$$

□

Theorem 1 can be used to calculate any of the U_n samples that determine the transitions by simply considering $k = n$, with $n > 1$. For the special case where $k = 1$, the path probabilities will be equal to the transition probabilities of the first step. We could then calculate the uniform sample U_{k+1} needed for the next step and recursively update $a_{s_{0:k+1}}$ and $b_{s_{0:k+1}}$ to get the new cumulative path probabilities using Equation (8). This strategy might not be optimal though, as it requires keeping track of two cumulative probabilities. A cleaner and more efficient solution would be to write U_k in terms of the previous uniform sample U_{k-1}.

Theorem 2. *If $U_{k-1} \sim \mathcal{U}(0,1)$ is used to draw a state sample s_{k-1}, then s_k is determined by*

$$U_k = \frac{U_{k-1} - a_{s_{k-1}}}{b_{s_{k-1}} - a_{s_{k-1}}} \tag{10}$$

Proof. Given a uniform sample U that determines the path, the samples U_k and U_{k-1} can be written as specified in (9). If we solve w.r.t. U in both cases, we obtain the following equality:

$$U_k Pr(s_{0:k-1}) + a_{s_{0:k-1}} = U_{k-1} Pr(s_{0:k-2}) + a_{s_{0:k-2}}$$

which yields:

$$U_k = \frac{U_{k-1} Pr(s_{0:k-2})}{Pr(s_{0:k-1})} - \frac{a_{s_{0:k-1}} - a_{s_{0:k-2}}}{Pr(s_{0:k-1})} \tag{11}$$

Using (8), the numerator of the second fraction above can be written as:

$$a_{s_{0:k-1}} - a_{s_{0:k-2}} = Pr(s_{0:k-2}) a_{s_{k-1}} + a_{s_{0:k-2}} - Pr(s_{0:k-3}) a_{s_{k-2}} - a_{s_{0:k-3}}$$

We also know that $a_{s_{0:k-2}} = Pr(s_{0:k-3}) a_{s_{k-2}} + a_{s_{0:k-3}}$ because of (8), so we can rewrite (11) as:

$$U_k = \frac{U_{k-1} Pr(s_{0:k-2})}{Pr(s_{0:k-1})} - \frac{a_{s_{k-1}} Pr(s_{0:k-2})}{Pr(s_{0:k-1})}$$

According to the definition of path probabilities in (5), we have $Pr(s_{0:k-1}) = Pr(s_{0:k-2})P_{s_{k-2}s_{k-1}}$, which implies:

$$U_k = \frac{U_{k-1} - a_{s_{k-1}}}{P_{s_{k-2}s_{k-1}}}$$

Finally, we can write $P_{s_{k-2}s_{k-1}}$ as a difference of cumulative probabilities to obtain Equation (10). □

Starting from some initial transition, we can recursively generate an entire sequence of random samples that are uniformly distributed between 0 and 1. If the previous step utilised a sample $U_{k-1} \sim \mathcal{U}(0,1)$, we know that $a_{k-1} < U_{k-1} \leq b_{k-1}$. If we define U_k according to (10), it is easy to show that $0 < U_k \leq 1$, which means that $U_k \sim \mathcal{U}(0,1)$. Although this sequence is produced deterministically, we have shown that it corresponds to the uniform sample needed to sample from the trajectory distribution.

Thus, assuming that the quantities $a_{s_{k-1}}$ and $b_{s_{k-1}} - a_{s_{k-1}}$ have to be calculated anyway, generating a sample at each step requires a subtraction followed by a division, as Equation (10) implies. This procedure is more efficient than most of the random generator algorithms, in particular the ones that produce high quality random numbers.

4. Sampling from the Time Distribution

4.1. Time Discretization

If we select some particular trajectory $s_{0:k}$, the duration of the total of the transitions involved is represented by a $L_{s_{0:k}}$ random variable. In the case of CTMCs this will be the sum of k exponentially distributed independent random variables that determine the duration of each transition, or more formally:

$$L_{s_{0:k}} = \sum_{i=0}^{k} L_{s_i} \tag{12}$$

where $L_{s_i} \sim Exp(\lambda_{s_i})$. Therefore, $L_{s_{0:k}}$ will follow hypo-exponential distribution with k parameters, or equivalently $L_{s_{0:k}} \sim Hypo(\lambda_{s_0}, \ldots, \lambda_{s_k})$. To sample directly from $L_{s_{0:k}}$ is only feasible for special cases such as the Gamma distribution, which is a hypo-exponential with k similar parameters. It is possible to transform $L_{s_{0:k}}$ to a Gamma distributed variable by applying uniformization [13]. This approach is problematic though, as the probability matrix of the embedded DTMC will contain self-loops, in contrast to the original jump chain as defined in (1). This means that the uniformised CTMC will involve a larger number of events, a fact that could actually slow the simulation down.

Our attempt of sampling from the hypo-exponential $L_{s_{0:k}}$ efficiently will focus on approximating the exponentially distributed sojourn times with a discrete random variable. The use of a discrete distribution implies that we divide time into intervals, since it involves discrete time-steps rather than continuous.

Thus, while a continuous distribution indicates the probability of a transition happening up to a time t, a discrete one indicates the transition probability up to the n-th interval.

The geometric distribution seems to be a reasonable choice for the task, since it is the discrete analogue of the exponential. Given some exponential random variable $L \sim Exp(\lambda)$, this can be approximated by a geometrically distributed $Y \sim G(p)$ that denotes the number of Bernoulli trials needed to fire a transition with probability p. The geometric distribution is normally supported in $\mathbb{N}^*$. Given the length of intervals l, we can map a geometric random variable to $\mathbb{R}^+$ by considering that it is supported in $\{1l, 2l, \ldots, nl\}$. Since Y is geometric, its expected value will be $1/p$ intervals, or l/p in terms of time units. If we make L and Y correspond to the same expected value, that is $1/\lambda = l/p$, it is easy to show that the interval length will be:

$$l = \frac{p}{\lambda} \tag{13}$$

Therefore, to determine the amount of time spent in state s_k will involve two steps:

1. Sample from $Y_{s_k} \sim G(p)$. Using a uniform sample $U \sim \mathcal{U}(0,1)$, we choose a $n \in \mathbb{N}^*$ with probability:

$$Pr(Y_{s_k} = n) = Pr(a_{Y_{s_k}} < U \leq b_{Y_{s_k}}) \tag{14}$$

   where $b_{Y_{s_k}} = Pr(Y_{s_k} \leq n)$ and $a_{Y_{s_k}} = Pr(Y_{s_k} \leq n-1)$.

2. Calculate the time spent in state s_k:

$$t_{s_k} = nl_{s_k} = n\frac{p}{\lambda_{s_k}} \tag{15}$$

The advantage of time discretization is that we can use the trajectory sampling technique presented in Section 3, and therefore reduce the random samples generated. To illustrate how this is possible, let us consider the stochastic process $\{Y_{s_k}\}_t$ that denotes the collection of geometrically distributed random variables used to approximate the sojourn times in some CTMC. If we set the same parameter p for these random variables, then they will be independent and identically distributed. We can easily verify that $\{Y_{s_k}\}_t$ is essentially a Markov process, which means that it is possible to generate an entire trajectory using a single uniform sample, as demonstrated in the previous section. The time discretization was necessary, otherwise it would not be possible to define the discrete state Markov chain $\{Y_{s_k}\}_t$, and therefore employ the path sampling technique.

One desirable property of our approach is that it gives an estimation for the duration of all of the transitions involved in a trajectory. On the contrary, the Gamma sampling used in K-skip only produces the total duration of k transitions. While both approaches use a single random number to determine the

duration of trajectories, K-skip is expected to be superior from a performance point of view. However, our method produces trajectories that are as detailed as the ones of the original Markov chain.

One other strength of our approach is that its applicability does not rely on particular model properties. The Gamma sampling used in K-skip assumes that exit rates do not change much during the k steps. This assumption, which is similar to the leap condition in τ-leaping methods, may not hold for some models meaning that it could be an extra source of error. Our method is not exact however, due to the time discretization employed. The quality of this approximation is discussed in the rest of this section.

4.2. Quality of Approximation

Since the interval length l is dependent on the parameter p of the Geometric distribution, it is rather intuitive that smaller values for p result in better approximation, as l also tends to get smaller. We are going to characterise the quality of this approximation in a rigorous manner.

Theorem 3. *Let us consider some stochastic process that approximates some CTMC featuring the same state-space S, the same transition probability matrix P, and the same initial distribution $\boldsymbol{\pi}_0$. The approximate process is only different in the sense that the sojourn times are determined by $Y_{s_k} \sim G(p)$, as described in (14) and (15). Then, the approximate process converges to the corresponding CTMC, as $p \to 0$.*

Proof. Let us define $P(t)$ as the transition probability matrix of a CTMC at time t. Given an initial state distribution vector $\boldsymbol{\pi}_0$, the distribution vector of the CTMC at time t will be:

$$\boldsymbol{\pi}_t = \boldsymbol{\pi}_0 P(t) \tag{16}$$

$P(t)$ can be calculated as a weighted sum of different powers of the probability matrix P of the underlying jump chain. The state distribution at t can then be rewritten as follows:

$$\boldsymbol{\pi}_t = \boldsymbol{\pi}_0 \sum_{k=0}^{\infty} P^k \times Pr(k \text{ steps until } t) \tag{17}$$

The modified stochastic process that resulted from this geometric approximation will have the same underlying jump chain as the original CTMC. The only term in (17) that is different in those two kinds of processes is the probability of k transitions happening until time t. This probability can be expressed as a sum of the probabilities of all trajectories being less or equal to t, weighted by the path probabilities:

$$Pr(k \text{ steps until } t) = \sum_{s_{0:k} \in \mathcal{T}_k} Pr(L_{s_{0:k}} \leq t) Pr(s_{0:k}) \tag{18}$$

In order to show that the behaviour of some CTMC as given in (17) is well approximated, it is sufficient (although not necessary) to show that the probability of k transitions until t is approximately the same for the two kinds of

processes. The modified process will have same path probabilities as its corresponding CTMC, since the jump process is the same. Thus, two corresponding processes are only different w.r.t the distribution of $L_{s_{0:k}}$. Therefore, it is sufficient (although not necessary again) to show that the cumulative distribution functions for the sojourn times are approximately the same:

$$\begin{aligned} Pr(L_{s_k} \leq t) \approx Pr(Y_{s_k} \leq n) &\Leftrightarrow 1 - e^{-\lambda t} \approx 1 - (1-p)^n \\ &\Leftrightarrow e^{-\lambda t} \approx (1-p)^n \end{aligned}$$

We can now plug in the equation the interval length l to our convenience:

$$e^{-\lambda t} \approx (1-p)^{nl/l}$$

Since we are only interested for t such that $t = nl$, we can discard t and nl:

$$e^{-\lambda} \approx (1-p)^{1/l}$$

We can also substitute the l on the exponent according to (13).

$$e^{-\lambda} \approx (1-p)^{\lambda/p}$$

Finally, we can also discard λ in both sides to obtain:

$$1/e \approx (1-p)^{1/p} \tag{19}$$

The last equation is valid for values of p close to zero, as it can be easily shown by the limit:

$$\lim_{p \to 0} (1-p)^{1/p} = 1/e \tag{20}$$

□

The p parameter is a probability, so we have $0 < p \leq 1$. Equations (19) and (20) imply that smaller values of p result in much better approximation. However, a value for p that is too small can make the geometric sampling described by (14) inefficient, as the cumulative probabilities of the form $Pr(Y_{s_k} \leq n)$ will involve too many terms. Hence, we need a trade-off between approximation quality and efficiency. In the experiments that follow, we use two different values: $p = 1$ that implies deterministic time-steps that depend on the current state only, and $p = 0.1$ which we think that it is a more appropriate choice, judging by the experimental results of Section 6.

5. Implementation Issues

Although sampling from the trajectory distribution as discussed in Section 3 is theoretically correct, it gives rise to some computational issues. In most computer systems, the mantissa for the double-precision floating-point format contains 53 bits. That is why most random generators produce doubles of the form: $m \times 2^{-53}$, where m is a uniformly distributed integer. In other words,

a random generator is capable of producing 2^{53} different values. The trajectory sampling algorithm will have 2^{53} different inputs resulting in 2^{53} different trajectories at most. The number of the possible trajectories can easily exceed this value even for not so long simulation runs, since it grows exponentially with number of simulation events. Therefore, it is inevitable that we will miss a significant number of possible simulation paths.

This effect can be eliminated if we sample trajectories of some particular length k, such that the number of possible simulation paths are significantly smaller than the number of uniformly distributed doubles. A value $k = 10$ is a reasonable choice that suits most of the models that we have encountered in practice. Given 53-bits of precision for the mantissa, we have $2^{53} \approx 9 \times 10^{15}$ different possible random numbers. The largest model that we have tested is LacY [14] involving 21 bio-chemical reactions, which means that the maximum number of transitions available is also 21. If we set $k = 10$, we have $21^k \approx 1.66 \times 10^{13} \ll 2^{53}$.

Each step in path sampling simulation consists of two actions: trajectory sampling as described in Section 3, and time sampling using geometric approximation. These concepts are summarised in Algorithm 1, which involves two parameters: p that controls the granularity of the geometric distribution and the length k of the trajectories to sample. In our implementation the probabilities of the geometric distribution have been pre-calculated for efficiency.

Algorithm 1 Path Sampling Simulation

1: Initialise system state and set $0 < p \leq 1$ and $k \geq 1$
2: Draw samples $U_L \sim \mathcal{U}(0,1)$ and $U_X \sim \mathcal{U}(0,1)$
3: **while** $t < t_{final}$ **do**
4: Given M transitions, calculate the transition rates λ_m, $\forall m \in \{1, 2, \ldots, M\}$
5: Calculate $\lambda = \sum_{m=1}^{M} \lambda_m$, which is the rate of leaving the current state
6: Using sample U_L, draw n from the geometric distribution $G(p)$
7: Using sample U_X, pick transition m with probability λ_m/λ
8: Update time: $t = t + np/\lambda$
9: Update state with effect of transition m
10: **if** iteration mod $k \neq 0$ **then**
11: Update U_L and U_X according to Equation (10)
12: **else**
13: Draw samples $U_L \sim \mathcal{U}(0,1)$ and $U_X \sim \mathcal{U}(0,1)$
14: **end if**
15: **end while**

6. Experiments

The efficiency of our algorithm stems from the fact that it generates fewer random numbers. One of the most popular random generators in the literature is *Mersenne Twister* (MT) [18]. It produces high quality random numbers while it exhibits performance comparable to the most efficient algorithms of its kind, as can be seen in [15]. We have developed our algorithm in Java using a number

of open-source libraries that contain implementations of MT, namely Apache Commons, CERN Colt, JAMES II [12] and SSJ [16]. The implementations used produce doubles whose mantissa precision is 53 bits.

We have applied our approach to simulate two different models of biochemical reaction networks. The first model is LacY, which involves 21 reactions and 22 species, as appeared in [14]. The second example is Goldbeter's oscillatory model [11] as presented in [7], which involves 7 reactions and 6 species. Both models have been simulated using ODM, K-skip, and path sampling simulation. The implementation of both K-skip and path sampling is based on ODM, hence any efficiency comparisons have ODM as a baseline.

Two different parameters were used for the geometric approximation: $p = 1$ and $p = 0.1$. Table 1 contains the running times for different random generators. The experiments have been performed in an Intel® Xeon™ E5410 @ 2.33GHz PC running Scientific Linux 6. The results imply that path sampling is about $15 \sim 20\%$ faster than ODM. A second observation is that using $p = 1$ is not significantly faster than path sampling using $p = 0.1$ for the geometric distribution. This means that there is no need to use a value for p greater than 0.1, as this would not result in a significant improvement in efficiency.

Comparing running times for K-skip and path sampling, we see that K-skip is clearly the most efficient of the two. This is because it determines the total duration of k events by sampling from a Gamma distribution, while we determine the duration of every single event happening. We note that, the speedups observed for K-skip are smaller than the values reported in [2]. This is due to the MT random number generator, which is more efficient than the ran2 algorithm used in Cai & Wen paper, as pointed out in [15]. Because we are using a more efficient random generator there is less scope to deliver speedups over the ODM. If we consider this difference, the results we have found for K-skip seem to comply with the ones reported in the original work.

Table 1: Execution times in seconds for 10^5 simulation runs

(a) LacY model, $t_{final} = 1000$

			Path Sampling	
	ODM	K-skip	$p = 1$	$p = 0.1$
Apache Commons	8759	5588	6930	6951
CERN Colt	9043	5568	6974	6915
JAMES II	9684	5490	7043	6944
SSJ	10452	5562	7248	7322

(b) Goldbeter's model, $t_{final} = 10$

			Path Sampling	
	ODM	K-skip	$p = 1$	$p = 0.1$
Apache Commons	12264	9354	10678	10615
CERN Colt	12685	9514	10662	10658
JAMES II	13531	9857	10598	10596
SSJ	14636	9269	10719	10734

A second issue that has to be explored is whether the stochastic process described by Algorithm 1 is equivalent to the original Markov chain. The convergence is ensured as $p \to 0$ when $k = 1$. The simulation will be still exact even if $k > 1$ as implied by Theorems 1 and 2. However, the use of geometric approximation means that we have a slightly altered process that approximates the original. To assess the quality of this approximation we calculate the histogram distance for the distribution of various rewards (i.e. species populations) in the models used, as it would have been impractical to compare the entire state-space distribution for models of that size.

It is important to note that the distance between the original and the approximated distribution will always be larger than zero, even if the simulation is exact, since the empirical distributions which result from simulation are always going to be different. In order to determine whether the distance calculated is significant, it has to be compared with the self-distance. According to [6], an upper bound for the average histogram self-distance given N samples is independent from the distribution and it can be calculated using $\sqrt{(4K)/(\pi N)}$, where K is the number of intervals in the histogram. For the examples that follow, we have considered $K = 50$.

Table 2 summarises the histogram distances for several species populations and time-points in the models considered. For path sampling with $p = 1$, some of the distances are slightly larger than the self-distance (the values written in italics). This implies that we have a reasonably good approximation but the error introduced by using fixed times is observable for the number of samples considered. However, the approximation quality is better when using path sampling with $p = 0.1$, as it was expected. The histogram distance from the true distribution is at the same level or smaller than the self-distance estimated almost in all cases. This means that the error observed is within the limits of the error inherently introduced by the simulation process. Those findings support the claim that path sampling simulation with parameter $p = 0.1$ for the geometric approximation is an accelerated simulation approach that is almost exact.

While K-skip proved to be more efficient than our approach for the LacY and Goldbeter models, Table 2 suggests that is not as accurate in some cases. Most of the histogram distances for the LacY model are greater than either the self-distance or the corresponding distances calculated for both versions of path sampling. It seems that the assumption that the rates of subsequent states are similar might introduce some errors, a fact that renders K-skip less appropriate for some models. Our approach generalises to systems where this assumption is not valid. Moreover, our use of the geometric approximation specifies the duration of every single event happening, which can be important for some systems.

7. Conclusions

Path sampling simulation requires fewer random samples to generate Markov chain trajectories. This is achieved by using a single random number to deter-

Table 2: Histogram distances for 10^6 simulation runs (self-distance: 0.0080)

(a) LacY model

	K-skip			Path Sampling ($p = 1$)			Path Sampling ($p = 0.1$)		
Time	lactose	PLac	product	lactose	PLac	product	lactose	PLac	product
250	0.0070	*0.0090*	0.0064	0.0062	0.0005	0.0071	0.0045	0.0011	0.0054
500	0.0040	0.0074	*0.0087*	0.0042	0.0011	*0.0083*	0.0030	0.0004	0.0071
750	0.0041	0.0077	0.0074	0.0034	0.0004	*0.0086*	0.0050	0.0001	0.0076
1000	0.0044	*0.0086*	*0.0081*	0.0040	0.0004	*0.0087*	0.0032	0.0002	0.0079

(b) Goldbeter's model

	K-skip			Path Sampling ($p = 1$)			Path Sampling ($p = 0.1$)		
Time	active_M	active_X	C	active_M	active_X	C	active_M	active_X	C
2	0.0065	0.0068	0.0074	0.0071	0.0039	0.0039	0.0048	0.0040	0.0036
5	0.0067	0.0059	0.0055	0.0067	0.0066	0.0054	0.0066	0.0053	0.0052
7	0.0070	*0.0088*	0.0037	0.0068	*0.0082*	0.0054	0.0080	0.0079	0.0060
10	0.0071	0.0063	0.0067	0.0055	0.0048	*0.0081*	0.0041	0.0061	0.0062

mine an entire sequence of transitions. We have proved that the random number required to select the next transition can be written in terms of the random number that selected the previous transition. This leads to a recursive update of a single random number that determines an entire simulation path. In the case of CTMCs a second random number is used to determine the length of this sequence. The same concept has been used by approximating the exponentially distributed times with a geometric distribution with parameter p that controls the quality of this approximation.

We have simulated two bio-chemical models of different nature to assess the efficiency and the accuracy of the our method. The experimental results show that our approach is about $15 \sim 20\%$ faster than the ODM, while the errors observed were found to be negligible. K-skip method I, which is a similar approach, was found to be more efficient but in some cases less accurate. Thus, path sampling simulation can be thought of as an alternative to K-skip in cases where this is possibly inappropriate.

There are also some practical considerations with respect to the length k for the trajectories to be sampled. A too large value for k might result in missing possible simulation paths, while a value too small will degenerate path sampling simulation to the ODM. We have used $k = 10$ for the experiments produced, but in the case of larger models we would have to set a smaller value for k. We think that $k = 5$ is appropriate even for very large models. For example, given a model with 500 reactions we have: $500^k \approx 3.125 \times 10^{13} \ll 2^{53}$.

Acknowledgements

The authors are supported by SynthSys, a Centre for Integrative Systems Biology (CISB) funded by BBSRC and EPSRC, reference BB/D019621/1.

References

[1] A. Auger, P. Chatelain, and P. Koumoutsakos. R-leaping: accelerating the stochastic simulation algorithm by reaction leaps. *The Journal of Chemical Physics*, 125(8):084103, 2006.

[2] X. Cai and J. Wen. Efficient exact and K-skip methods for stochastic simulation of coupled chemical reactions. *The Journal of chemical physics*, 131(6):064108, 2009.

[3] X. Cai and Z. Xu. K-leap method for accelerating stochastic simulation of coupled chemical reactions. *The Journal of chemical physics*, 126(7):074102, Feb. 2007.

[4] Y. Cao, D. T. Gillespie, and L. R. Petzold. The slow-scale stochastic simulation algorithm. *The Journal of Chemical Physics*, 122(1):14116, 2005.

[5] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *The Journal of Chemical Physics*, 121(9):4059–4067, 2004.

[6] Y. Cao and L. Petzold. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. *Journal of Computational Physics*, 212(1):6–24, Feb. 2006.

[7] F. Ciocchetta and J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065–3084, 2009.

[8] M. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry*, 104(9):1876–1889, 2000.

[9] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[10] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[11] A. Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proceedings of the National Academy of Sciences of the United States of America*, 88(20):9107–9111, 1991.

[12] J. Himmelspach and A. M. Uhrmacher. The JAMES II Framework for Modeling and Simulation. *2009 International Workshop on High Performance Computational Systems Biology*, pages 101–102, 2009.

[13] A. Jensen. Markov chains as an aid in the study of Markov processes. *Skand. Aktuarietidskrift*, 36:87–91, 1953.

[14] A. M. Kierzek. STOCKS: STOChastic Kinetic Simulations of biochemical systems with Gillespie algorithm. *Bioinformatics*, 18(3):470–481, 2002.

[15] P. L'Ecuyer. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4):1–40, 2007.

[16] P. L'Ecuyer and E. Buist. Simulation in java with SSJ. *Proceedings of the Winter Simulation Conference 2005*, pages 611–620, 2005.

[17] H. Li and L. Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. *Technical report, Department of Computer Science, University of California, Santa Barbara*, 2006.

[18] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, Jan. 1998.

[19] E. Mjolsness, D. Orendorff, P. Chatelain, and P. Koumoutsakos. An exact accelerated stochastic simulation algorithm. *The Journal of Chemical Physics*, 130(14):144110–14, Apr. 2009.

[20] C. V. Rao and A. P. Arkin. Stochastic chemical kinetics and the quasi steady-state assumption: application to the Gillespie algorithm. *The Journal of Chemical Physics*, 118(11):4999–5010, 2003.