THIELE CENTRE
FOR APPLIED MATHEMATICS IN NATURAL SCIENCE

Markov Bridges, Bisection and Variance Reduction

Søren Asmussen and Asger Hobolth

11

Research Report

No. 01 I January 2011

Markov Bridges, Bisection and Variance Reduction

Søren Asmussen[*] and Asger Hobolth[**]

[*]Department of Mathematical Sciences, Aarhus University, asmus@imf.au.dk
[**]Bioinformatics Research Center, Aarhus University, asger@birc.au.dk

Abstract

Time-continuous Markov jump processes is a popular modelling tool in disciplines ranging from computational finance and operations research to human genetics and genomics. The data is often sampled at discrete points in time, and it can be useful to simulate sample paths between the datapoints. In this paper we firstly consider the problem of generating sample paths from a continuous-time Markov chain conditioned on the endpoints using a new algorithm based on the idea of bisection. Secondly we study the potential of the bisection algorithm for variance reduction. In particular, examples are presented where the methods of stratification, importance sampling and quasi Monte Carlo are investigated.

1 Introduction

Let $X = \{X(t) : t \geq 0\}$ be a Markov process in continuous time with discrete or general state space E. A *Markov bridge* with parameters T, a, b is then a stochastic process with time parameter $t \in [0, T]$ and having the distribution of $\{X(t) : 0 \leq t \leq T\}$ conditioned on $X(0) = a$, $X(T) = b$.

Markov bridges occur in a number of disciplines ranging from computational finance and operations research to human genetics and genomics. In many applications, it is of relevance to simulate sample paths of such bridges. In particular, the case of diffusions has received extensive attention. An early reference is [27] and later selected ones [5, 6], and [8]. Also some special Lévy processes have been considered, see [3, 25, 26], and [20]. A more theoretical discussion of Markov bridges is in [15].

The present paper is concerned with the CTMC (continuous time Markov chain) case where the state space E of X is finite. This case is of course much simpler than diffusions or Lévy processes, but nevertheless, it occurs in some important applications. With E finite, there is a simple description of the process in terms of the rate (intensity) matrix $Q = (q_{ij})_{i,j \in E}$: a jump $i \to j \neq i$ occurs at rate q_{ij}. Equivalently, state i has an exponential holding time with mean $1/q_i$ where $q_i = -q_{ii} = \sum_{j \neq i} q_{ij}$, and upon exit, the new state equals $j \neq i$ with probability $\theta_{ij} = q_{ij}/q_i$.

We focus here on a bisection algorithm first presented in [2]. The details are surveyed in Section 4, but the key is two fundamental observations. Firstly, if the endpoints are the same and the process does not experience any jumps, the sample path generation is finished. Secondly, if the endpoints are different and the process experiences exactly one jump, sample path generation is easy; we must basically simulate a waiting time before the jump from a truncated exponential distribution. These two fundamental observations are described in more detail in Section 4.1, but once they are in place they immediately suggest a recursive procedure for sample path generation: continue splitting the large time interval into smaller time intervals, and keep splitting until all intervals contain either zero or one jump only.

Previous algorithms for bridge sampling from CTMCs include *rejection sampling*, *uniformization* and *direct simulation* and are briefly surveyed in Section 3 (also Markov chain Monte Carlo methods have been used, e.g. [27] and [5], but we do not discuss this aspect here). Comparisons of these algorithms are in [17], and comparisons with the bisection algorithm in [2]. The overall picture is that no algorithm is universally superior in terms of fast generation of sample paths. In particular, we do not insist that the bisection idea is a major jump forward in this respect. Rather, the point of view of the present paper is to advocate its potential for variance reduction by identifying some 'most important' random variables on which to concentrate variance reduction ideas. This idea is familiar from the Brownian bridge, see for example [9, 23, 10] and [1, p. 277–280]. Here the 'most important' r.v.'s are first $X(0), X(T)$, next $X(T/2)$, then $X(T/4), X(3T/4)$ and so on. However, in our implementation of CTMC bridges a new aspect occurs since also the number of jumps in $[0,T]$, $[0,T/2]$, $[T/2,T]$ and so on play a role. The variance reduction techniques we study are stratification, importance sampling and quasi Monte Carlo.

2 Examples

2.1 Statistical inference in finite state CTMC models

For statistical purposes, the relevant parameters to estimate are often the elements q_{ij} of the rate matrix Q, or, equivalently, the q_i and the $\theta_{ij} = q_{ij}/q_i$, $j \neq i$. In the case of complete observations in $[0,T]$ (that is, the whole trajectory $\{X(t) : 0 \le t \le T\}$ is observed), there is a simple solution: the maximum likelihood estimators $\widehat{q_i}$, $\widehat{\theta}_{ij}$ of q_i and the $\theta_{ij} = q_{ij}/q_i$ are just the empirical counterparts. That is, the sufficient statistics are

$$\begin{aligned} T_i &= \text{time in state } i = \int_0^T I\big(X(t) = i\big)\,\mathrm{d}t\,, \\ N_{ij} &= \#\,(\text{jumps } i \to j) = \sum_{0\le t\le T} I\big(X(t-) = i,\, X(t) = j\big)\,, \\ N_i &= \sum_{j\neq i} N_{ij}\,, \end{aligned}$$

and the maximum likelihood estimators are

$$\widehat{q_i} = \frac{N_i}{T_i}\,, \quad \widehat{\theta}_{ij} = \frac{N_{ij}}{N_i}\,. \tag{1}$$

In many applications of continuous time Markov chains, the stochastic process $\{X(t) : t \geq 0\}$ is, however, sampled at equidistant discrete points

$$t_0 = 0 < t_1 = h < t_2 = 2h < \cdots < t_{n-1} = (n-1)h < t_n = nh = T$$

in time, while the process itself is a continuous-time process. This situation is a missing data problem, for which the EM (Expectation-Maximization) algorithm is a classical tool. This algorithm is iterative, i.e. in step k it has a trial $q_i^{(k)}$, $\theta_{ij}^{(k)}$ for the parameters. To update to $k+1$, one then in (1) replaces the sufficient statistics by their conditional expectation with parameters $q_i^{(k)}$, $\theta_{ij}^{(k)}$ given the data $X_0, X_h, \ldots, X_{(n-1)h}, X(T)$. That is,

$$\widehat{q}_i^{(k+1)} = \frac{N_i^{(k)}}{T_i^{(k)}}\,, \quad \widehat{\theta}_{ij}^{(k+1)} = \frac{N_{ij}^{(k)}}{N_i^{(k)}}\,, \tag{2}$$

where

$$\begin{aligned}
T_i^{(k)} &= \mathbb{E}_{q_i^{(k)},\theta_{ij}^{(k)}}\Big[\int_0^T I\big(X(t)=i\big)\,\mathrm{d}t \,\Big|\, X_0, X_h, \ldots, X_{(n-1)h}, X(T)\Big]\,,\\
N_{ij}^{(k)} &= \mathbb{E}_{q_i^{(k)},\theta_{ij}^{(k)}}\Big[\sum_{t\leq T} I\big(X(t-)=i,\, X(t)=j\big) \,\Big|\, X_0, X_h, \ldots, X_{(n-1)h}, X(T)\Big]\,,\\
N_i^{(k)} &= \sum_{j\neq i} N_{ij}^{(k)}\,,
\end{aligned}$$

The computation of these conditional expectations is the *E-step* of the algorithm, whereas (2) is the *M-step*. The E-step is the computationally demanding one. As a consequence of the Markov assumption, knowledge of the data partitions the problem into $n = T/h$ independent problems. For example,

$$T_i^{(k)} = \sum_{m=1}^{n} \mathbb{E}_{q_i^{(k)},\theta_{ij}^{(k)}}\Big[\int_{(m-1)h}^{mh} I\big(X(t)=i\big)\,\mathrm{d}t \,\Big|\, X\big((m-1)h\big), X(mh)\Big]\,.$$

The computations are in principle feasible via deterministic numerical analysis, but the implementation is somewhat tedious, so it is popular to use simulation instead. Then independent sample paths $\{X(t) : (m-1)h \leq t < mh\}$ must be generated between the timepoints $(m-1)h$ and mh, conditional on the datapoints $X\big((m-1)h\big)$ and $X(mh)$. This is how the problem of simulating Markov bridges arises in the statistical context.

For more information on rate matrix estimation in partially observed finite state CTMC models we refer to the paper by [21] and references therein.

2.2 Applications in genetics

A DNA string is a word from the alphabet `A`, `G`, `C`, `T`. When observing two closely related species like e.g. human and mouse, letters are equal at most sites (more than 80 %), but differ at a few as in Figure 1 where the two strings are identical except at the third site. The lines in the figure are ancestral lineages back to the common

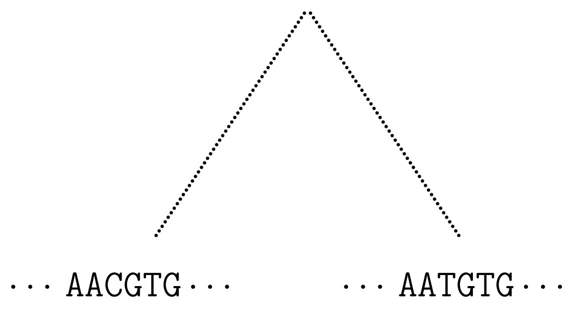

Figure 1: Related sites of DNA from human and mouse are identical at most positions. The possible states are from the DNA alphabet {A, G, C, T}.

ancestor. At each site, mutations occur, changing for example an A to a G. One is often interested in the (unobservable) complete history along the ancestral lines.

For a fixed single site, the common model assumes Markovian mutations at known exponential holding rates q_A, q_C, q_G, q_T and known transition probabilities (e.g. $\theta_{AG} = q_{AG}/q_A$ for $\mathtt{A} \to \mathtt{G}$). One further assumes time reversibility and that the ancestral lines are so long that stationarity has been reached. One can then reverse time along one of the ancestral lines, say the one starting from the human, to get a Markov process running from e.g. human to mouse and having known endpoints, see Figure 2.

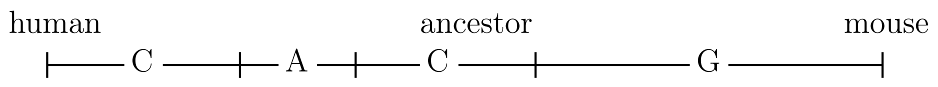

Figure 2: Example of evolution from human to mouse at a specific position. Time is reversed at the human lineage when compared to the previous figure.

An early popular model is that of [19] where the Q-matrix takes the form

	A	G	C	T
A	$-\alpha-2\beta$	α	β	β
G	α	$-\alpha-2\beta$	β	β
C	β	β	$-\alpha-2\beta$	α
T	β	β	α	$-\alpha-2\beta$

One readily computes the stationary distribution $\pi = (1,1,1,1)/4$ and checks the conditions of detailed balance ($\pi_G q_{GT} = \pi_T q_{TG}$ etc.) so that the model is indeed time reversible. Plugging in specific values of α, β, T one can then simulate the Markov bridge from human to mouse to obtain information on the ancestral history. One possible application is to put a prior on the length $T/2$ of the ancestral lines and use the simulations to compute a posterior.

Endpoint conditioned CTMC's are thus a crucial modelling tool for the evolution of DNA sequences. At the nucleotide level the states for the DNA substitution process state space is 4 as described above. At the amino acid level the state space size is 20 and at the codon level the size is 61. The ancestry is usually represented by a binary tree where a node corresponds to two DNA sequences finding a common ancestor.

For a given set of DNA sequences observed at the leaves of a given tree we can determine the probabilities of the states in the inner nodes using Felsenstein's tree peeling algorithm [13], and therefore the basic setup is very similar to and endpoint conditioned CTMC. For more information on the use of CTMC methodology in evolutionary models of DNA sequences we refer to [11, Chapter 13 and 14], [14, Chapter 13 and 14] and references therein.

Single site analysis is not completely satisfactory because there is dependence among neighboring sites. A simple and popular model assumes that the Q-matrix at site j only depends on the states at sites $j-1$ and $j+1$ as in Figure 3.

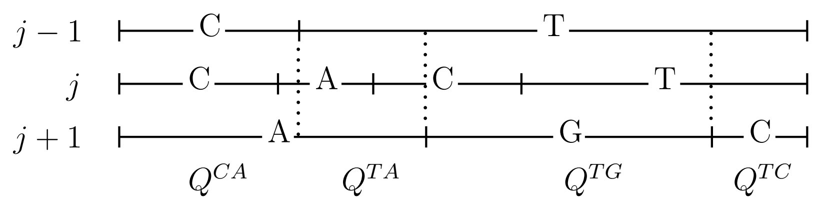

Figure 3: Illustration of neighbour dependence. The Q-matrix at site j depends on the states at the neighbouring sites. In the figure the neighbouring states at site $(j-1, j+1)$ are (C, A); (T, A); (T, G) and (T, C). When simulating site j condition on the endpoints one must take the states of the neighbouring sites into account.

Simulation of multiple sites can then be performed by Gibbs sampling, where one site at a time is updated. For updating of site j, one first simulates X at change points, i.e. times of state change of either site $j-1$ or $j+1$. These values form an inhomogeneous end-point conditioned discrete time Markov chain with easily computed transition probabilities. Once they are known, the evolution between change points are Markov bridges.

3 Previous algorithms

Reference [17] describe and analyse 3 previously suggested algorithms for end-point conditional simulation from continuous time Markov chains. The algorithms are called *rejection sampling, uniformization* and *direct simulation.* We will only briefly describe the algorithms here. For a detailed description of the algorithms we refer to [17].

Recall that our aim is to simulate a sample path $\{X(t) : 0 \leq t \leq T\}$ from a continuous time Markov chain conditional on the end-points $X(0) = a$ and $X(T) = b$. In *rejection sampling*, a sample path is simulated forward in time from $X(T) = a$, and the path is accepted if $X(T) = b$. Reference [24] describe an improvement of the naive rejection sampling approach where it is taken into account that if $a \neq b$, at least one jump must occur. Nielsens improvement is particularly important when the time interval is short and the beginning and ending states are different. Rejection sampling is inefficient if it is unlikely to end up in the desired ending state.

In *uniformization* (e.g. [12]), the number of state changes within an interval is Poisson distributed. The state changes themselves constitute a Markov chain. The price for the simple description of the number of state transitions is that virtual state changes (in which the state does not change) are permitted. Sampling from this related process is equivalent to sampling from the target continuous time Markov chain when the virtual changes are ignored. When simulating an endpoint conditioned sample path using uniformization, the number of state transitions is firstly simulated. This number follows a slightly modified Poisson distribution (the modification comes from the conditioning on the endpoints). When the number of jumps is simulated, the Markovian structure of the state transitions is utilized to simulate the types of changes that occur. Uniformization is usually very efficient, but can be slow if many virtual state changes are needed in the simulation procedure.

Finally, *direct simulation* [16] is based on analytical expressions for simulating the next state and the waiting time before the state change occurs. The expression for the waiting time distribution and corresponding cumulative distribution function are analytically available, but unfortunately not very tractable. Therefore the recursive steps of simulating the new state and corresponding waiting time is a rather time-consuming process.

4 Bisection Algorithm

The algorithm involves an initialization step and a recursive step. The recursive step is easy once the initialization step is explained. We divide the discussion of the initialization into two parts. In the first part, the end-points are the same, and in the second part the end-points are different.

4.1 The basic idea

The bisection algorithm is based on two fundamental observations:

1. If $X(0) = X(T) = a$ and there are no jumps we are done: $X(t) = a,\ 0 \le t \le T$.
2. If $X(0) = a$ and $X(T) = b \neq a$ and there is precisely one jump we are basically done: $X(t) = a,\ 0 \le t < \tau$, and $X(t) = b,\ \tau \le t \le T$.

In 2, the jump time τ is determined by the Lemma and corresponding Remark below.

The basic idea of the bisection algorithm is to formulate a recursive procedure where we finish off intervals with zero or one jumps according to the two fundamental observations above, and keep bisecting intervals with two or more jumps. The recursion ends when no intervals with two or more jumps are present. The following Lemma and Remark shows that intervals with precisely one jump are easy to handle.

We recall the notation $Q = \{q_{ab}\}$ for the instantaneous rate matrix with off-diagonal entries $q_{ab} \ge 0$ and diagonal entries $q_{aa} = -\sum_{b \neq a} q_{ab} = -q_a < 0$. We make the standard assumption that the process is irreducible and recurrent. The algorithm (as well as uniformization and direct simulation, cf. Section 3) require the transition probabilities $P_{ab}(t)$, i.e. the elements of the transition matrix $P(t) = \mathrm{e}^{Qt}$.

These can easily be computed, for example, if Q can be written in diagonal form UDU^{-1} with $D = \operatorname{diag}(\lambda_i)$; then $P(t) = U \operatorname{diag}(\mathrm{e}^{\lambda_i t})U^{-1}$. For different methods, see the classical paper [22].

Lemma 4.1. *Consider an interval of length T with $X(0) = a$, and let $b \neq a$. The probability that $X(t) = b$ and there is only one single jump (necessarily from a to b) in the interval is given by*

$$R_{ab}(T) = q_{ab} \begin{cases} \dfrac{\mathrm{e}^{-q_a T} - \mathrm{e}^{-q_b T}}{q_b - q_a} & q_a \neq q_b \\ T\mathrm{e}^{-q_a T} & q_a = q_b. \end{cases} \tag{3}$$

The density of the time of state change is

$$f_{ab}(t; T) = \frac{q_{ab}\mathrm{e}^{-q_b T}}{R_{ab}(T)} \mathrm{e}^{-(q_a - q_b)t}, \quad 0 \leq t \leq T.$$

Furthermore, the probability that $X(T) = b$ and there are at least two jumps in the interval is $P_{ab}(T) - R_{ab}(T)$.

Proof. Let $N(T)$ denote the number of jumps in the interval $[0, T]$. The first two parts of the Lemma follow from

$$\begin{aligned} R_{ab}(T) &= \mathbb{P}\big(X(T) = j, N(T) = 1 \,\big|\, X(0) = a\big) \\ &= \int_0^T q_a \mathrm{e}^{-q_a t} \frac{q_{ab}}{q_a} \mathrm{e}^{-q_b(T-t)} \,\mathrm{d}t = q_{ab}\mathrm{e}^{-q_b T} \int_0^T \mathrm{e}^{-(q_a - q_b)t} \,\mathrm{d}t, \quad a \neq b. \end{aligned}$$

The last part is clear since the case of zero jumps is excluded by $a \neq b$. □

Remark 4.2. If $q_a = q_b$, the single state change is uniformly distributed in the interval $[0, T]$. If $q_a > q_b$, the time of the state change is an exponentially distributed random variable truncated to $[0, T]$. Such a random variable V is easily simulated by inversion (e.g. [1, p. 39]. If $q_a < q_b$, we have by symmetry that $f_{ab}(t)$ is the density of the random variable $T - V$, where V is an exponentially distributed random variable with rate $q_b - q_a$ truncated to $[0, T]$. Finally, if $q_a = q_b$, the time of the state change is simply uniform on $[0, T]$. □

4.2 Initialization when the endpoints are equal

Consider the case $X(0) = X(T) = a$. We may write

$$P_{aa}(T) = P_{aa}(T/2)P_{aa}(T/2) + \sum_{c \neq a} P_{ac}(T/2)P_{ca}(T/2). \tag{4}$$

The first term can be further dissected into

$$\begin{aligned} P_{aa}(T/2) &= \mathbb{P}(X(T/2) = a | X(0) = a) \\ &= \mathbb{P}(X(T/2) = a, N(T/2) = 0 | X(0) = a) \\ &\quad + \mathbb{P}(X(T/2) = a, N(T/2) \geq 2 | X(0) = a) \\ &= \mathrm{e}^{-q_a T/2} + [P_{aa}(T/2) - \mathrm{e}^{-q_a T/2}], \end{aligned} \tag{5}$$

and similarly the second term can be written as

$$P_{ac}(T/2) = R_{ac}(T/2) + [P_{ac}(T/2) - R_{ac}(T/2)]. \tag{6}$$

With the abbreviation $e_a = \mathrm{e}^{-q_a T/2}$, $r_{ab} = R_{ab}(T/2)$, $p_{ab} = P_{ab}(T/2)$ we obtain Table 1 when substituting (5) and (6) into (4).

Table 1: Possible scenarios when the endpoints $X(0) = a$ and $X(T) = a$ are the same.

case	number of jumps in first interval	number of jumps in second interval	(unconditional) probability	notation
1	0	0	$e_a e_a$	α_1
2	0	≥ 2	$e_a(p_{aa} - e_a)$	α_2
3	≥ 2	0	$(p_{aa} - e_a)e_a$	α_3
4	≥ 2	≥ 2	$(p_{aa} - e_a)(p_{aa} - e_a)$	α_4
5	1	1	$r_{ac} r_{ba}$	$\alpha_{5,c}$
6	1	≥ 2	$r_{ac}(p_{ca} - r_{ca})$	$\alpha_{6,c}$
7	≥ 2	1	$(p_{ac} - r_{ac})r_{ca}$	$\alpha_{7,c}$
8	≥ 2	≥ 2	$(p_{ac} - r_{ac})(p_{ca} - r_{ca})$	$\alpha_{8,c}$

Note that in case 1-4 we have $X(T/2) = a$, and in case 5-8 we have $X(T/2) = c \neq a$. Of course we have

$$P_{aa}(T) = \sum_{i=1}^{4} \alpha_i + \sum_{i=5}^{8} \sum_{c \neq a} \alpha_{i,c}.$$

In the initialization step, we select one of the cases with probabilities proportional to the corresponding α-value. In case the algorithm enters case 1 we are done. In case the algorithm enters case 5 we are almost done; we just need to simulate two waiting times according to Remark 4.2: one waiting time in the interval $[0, T/2]$ with beginning state a and ending state c, and another in the interval $[T/2, T]$ with beginning state c and ending state a.

In case the algorithm enters one or more intervals where the number of jumps are ≥ 2, further simulation is needed (case 2, 3, 4, 6, 7, 8), and we move on to the recursion step explained below. However, we only need to pass intervals to the next level of the algorithm if the number of jumps are larger or equal to two. If the selected case is case 2, for example, we only need to pass the second interval $[T/2, T]$ and the endpoints $X(T/2) = a$ and $X(T) = a$. Similarly, if the selected case is case 6 we use Remark 4.2 to simulate the waiting time to state c in the first interval (and keep the type and time of the state change in the memory), but we only pass on the second interval $[T/2, T]$ and the endpoints $X(T/2) = c$ and $X(T) = a$ to the next level.

4.3 Initialization when the endpoints are different

Now consider the case when the end-points $X(0) = a$ and $X(T) = b \neq a$ are different. This time we get

$$P_{ab}(T) = P_{aa}(T/2)P_{ab}(T/2) + P_{ab}(T/2)P_{bb}(T/2) + \sum_{c\neq(a,b)} P_{ac}(T/2)P_{cb}(T/2).$$

Using the same notation as previously, we get the 12 cases in Table 2.

Table 2: Possible scenarios when the endpoints $X(0) = a$ and $X(T) = b \neq a$ are different.

case	number of jumps in first interval	number of jumps in second interval	(unconditional) probability	notation
1	0	1	$e_a r_{ab}$	β_1
2	0	≥ 2	$e_a(p_{ab} - r_{ab})$	β_2
3	≥ 2	1	$(p_{aa} - e_a)r_{ab}$	β_3
4	≥ 2	≥ 2	$(p_{aa} - e_a)(p_{ab} - r_{ab})$	β_4
5	1	0	$r_{ab}e_b$	β_5
6	1	≥ 2	$r_{ab}(p_{bb} - e_b)$	β_6
7	≥ 2	0	$(p_{ab} - r_{ab})e_b$	β_7
8	≥ 2	≥ 2	$(p_{ab} - r_{ab})(p_{bb} - e_b)$	β_8
9	1	1	$r_{ac}r_{cb}$	$\beta_{9,c}$
10	1	≥ 2	$r_{ac}(p_{cb} - r_{cb})$	$\beta_{10,c}$
11	≥ 2	1	$(p_{ac} - r_{ac})r_{cb}$	$\beta_{11,c}$
12	≥ 2	≥ 2	$(p_{ac} - r_{ac})(p_{cb} - r_{cb})$	$\beta_{12,c}$

Note that we can merge case 1 and case 5 (corresponding to one jump):

$$e_a r_{ab} + r_{ab}e_b = R_{ab}(T).$$

It clearly holds that

$$P_{ab}(T) = \sum_{i=1}^{8} \beta_i + \sum_{i=9}^{12} \sum_{c\neq(a,b)} \beta_{i,c}.$$

In case 1–4 we have $X(T/2) = a$, in case 5–8 we have $X(T/2) = b \neq a$, and in case 9–12 we have $X(T/2) = c \neq (a, b)$.

In the initialization step, we select one of the cases with probabilities proportional to the corresponding β-value. If the algorithm enters one or more intervals where the number of jumps are larger than two, further simulation is needed (case 2, 3, 4, 6, 7, 8, 10, 11, 12). If the algorithm enters a case where the number of jumps is less than one in both intervals (case 1, 5, 9), we are essentially done.

Entering an interval with ≥ 2 jumps means that further simulation is needed. In the next subsection, we discuss this recursive part of the bisection algorithm.

4.4 Recursion and termination

When an interval with ≥ 2 jumps is entered, further simulation is needed. However, it is straightforward to calculate the probabilities for the various scenarios; the (unconditional) probabilities are given by Table 1 with case 1 removed if the end-points of the interval are the same, and by Table 2 with case 1 and 5 removed if the end-points of the interval are different. (The values entering in Table 1 and Table 2 should also be calculated for half as long a time interval). The algorithm terminates when no intervals with ≥ 2 jumps are present.

4.5 Implementation

In the bisection algorithm, we simulate state changes every time we have an interval with precisely one jump. Thus we organize the program such that at every level of the recursion we simulate (and record) the time and type of state change when an interval with one jump is present, and pass the time-points and end-points of intervals with ≥ 2 state changes to the next level.

5 Numerical examples

This is a collection of the results from three experiments where bisection ideas are used for variance reduction in time-continuous Markov jump processes. The three experiments are (i) Stratification, (ii) Importance sampling and (iii) Quasi Monte Carlo. We consider a $N \times N$ rate matrix Q where the rates are given by $Q_{1,2} = \lambda$, $Q_{n,n+1} = \mu$, $n = 2, \ldots, N-1$, $Q_{N,1} = \mu$, and all other rates are zero, cf. Fig. 4. We took $\mu = N$.

$$\begin{tikzcd}
1 \arrow[r, "\lambda"] & 2 \arrow[r, "\mu"] & 3 \arrow[r, "\mu"] & 4 \arrow[r, "\mu"] & \cdots \arrow[r, "\mu"] & N-1 \arrow[r, "\mu"] & N \arrow[llllll, bend right, "\mu"']
\end{tikzcd}$$

Figure 4: Transition diagram for the cyclic example

Our target is to determine the probability p_λ of exactly one cycle in the time interval $[0, 1]$ conditioned on the initial state $X(0) = 1$ and final state $X(1) = 1$. We stress that neither the model nor the problem of estimating p_λ are chosen because of their intrinsic interest but in order to investigate the potential of the bisection algorithm for variance reduction in a simple example. Indeed, the value of p_λ is the probability of making exactly N jumps conditional on the end-points $X(0) = X(1) = 1$. This probability can be calculated using the algorithm of [28] which for convenience is reproduced in the Appendix. In Table 3 we provide the exact probabilities of p_λ in our experiments.

One possibility of estimating p_λ is of course the crude Monte Carlo method, to just generate sample paths $\{X(t) : 0 \leq t \leq 1\}$ conditional on $X(0) = 1$ and $X(1) = 1$ (we have previously discussed several algorithms, including bisection, for obtaining such samples). Let Z_r, $r = 1, \ldots, R$ indicate if exactly one cycle is obtained in the

Table 3: Exact probability p_λ for one cycle for various state space sizes N and various ratios λ/N.

N	λ/N						
	0.10	0.45	0.80	1.00	1.20	3.10	5.00
4	0.138405	0.567199	0.766990	0.800363	0.803456	0.639064	0.562409
7	0.191982	0.818845	0.946474	0.948949	0.941157	0.857991	0.818518
10	0.253733	0.940944	0.987026	0.984950	0.981193	0.950914	0.935423
15	0.373870	0.992560	0.998395	0.997843	0.997230	0.992588	0.990111
20	0.506338	0.999121	0.999775	0.999687	0.999597	0.998920	0.998555
30	0.745579	0.999988	0.999995	0.999993	0.999992	0.999977	0.999970

R sample paths where R is the number of replications. Clearly $Z_r \sim \mathrm{Bin}(1, p_\lambda)$ and so the crude Monte Carlo estimator is $\bar{Z} = \sum_{r=1}^{R} Z_r/R$ with variance

$$\sigma^2 = \mathrm{Var}(\bar{Z}) = \frac{p_\lambda(1-p_\lambda)}{R}.$$

5.1 Stratification

First consider a proportional stratification procedure [1, V.7] where the R replicates are allocated according to the probability of the midpoint s, $s = 1, \ldots, N$ of the process. More precisely, let $p_s = \mathbb{P}(X(1/2) = s | X(0) = 1, X(1) = 1)$ be the probability of the midpoint being s and allocate $R_s = Rp_s$ replicates to this stratum. We use $\sum_{s=1}^{N} p_s \bar{Z}_s$ as an estimator of p_λ, where $\bar{Z}_s = \sum_{i=1}^{R_s} Z_{s,i}/R_s$ and $Z_{s,i}$ indicates if the ith sample in the sth stratum contains exactly one cycle.

Letting

$$p_{\lambda,s} = \mathbb{P}\big(\text{exactly one cycle} \,\big|\, X(0) = 1, X(1/2) = s, X(1) = 0\big)\,,$$

we obtain the stratum variance

$$\sigma^2_{\text{Str}} = \sum_{s=1}^{N} p_s^2 \frac{p_{\lambda,s}(1-p_{\lambda,s})}{R_s}.$$

We now see that the ratio between the two variances is given by

$$\frac{\sigma^2_{\text{Str}}}{\sigma^2} = \sum_{s=1}^{N} p_s \frac{p_{\lambda,s}(1-p_{\lambda,s})}{p_\lambda(1-p_\lambda)},$$

where we have used $R_s = Rp_s$.

In Figure 5 left we show (using exact calculations) the values of the ratio between the two variances for several values of λ and size of state space N. We see that when $\lambda \ll N$ we obtain a major reduction in the variance when stratification is applied. In the cases $\lambda \sim N$ and $\lambda \gg N$, a variance reduction is mainly obtained for large state spaces.

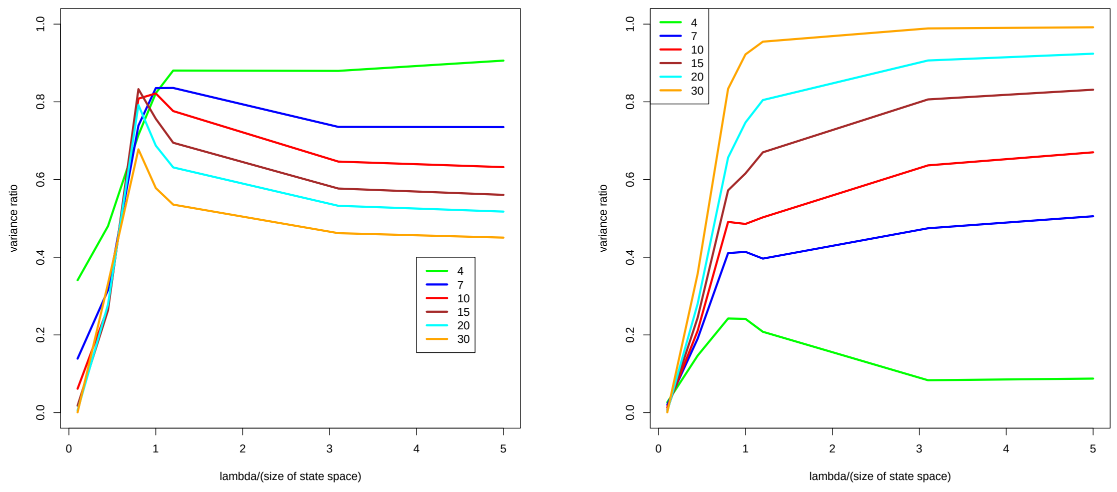

Figure 5: Variance reduction using stratification strategies. Left: Variance ratio between naive sampling and stratification according to midpoint. The variance reduction is large when λ is small and otherwise the variance reduction is moderate. Right: Variance ratio between stratification according to midpoint and stratification according to midpoint *and* information on number of jumps (0, 1 or ≥ 2). Again the variance reduction is large when λ is small.

Instead of only stratifying according to the midpoint, we can include information about the number of jumps according to Table 1. We thus include not only the midpoint but also if 0,1 or at least 2 jumps are present. We again apply a proportional stratification procedure. The variance ratio between the two stratification procedures are shown in Figure 5 right. We see that a major variance reduction is obtained for small values of λ and small state spaces.

5.2 Importance sampling

Another variance reduction mechanism is importance sampling. We choose to do importance sampling on the midpoint and include information that (a) the chain can only jump from n to $(n + 1)$ (modulo N) and (b) one cycle corresponds to exactly N jumps. Having sampled the midpoint and number of jumps in the two intervals, we proceed according to the bisection algorithm.

The N jumps are distributed in the two intervals according to a multinomial distribution with probability vector $(1/2, 1/2)$ and number of trials N, i.e. the number of jumps in the first interval follows a binomial distribution $\text{Bin}(N, 1/2)$ with parameter $1/2$ and N trials. We have outlined the proposal mechanism in Table 4 (compare with Table 1).

In Figure 6 we show the ratio between the importance sampling variance and the 'naive' sampling scheme. Even though the importance sampling scheme takes information about the type of CTMC into account it appears that we only obtain a variance reduction when the state space is smaller than 15, and even so it is modest.

Table 4: Possible number of jumps in the two intervals in the importance sampling scheme. In case 8, the last case, the value of the number of jumps k is between 2 and $N-2$.

case	number of jumps in first interval	number of jumps in second interval	sampling probability
1	0	0	0
2	0	≥ 2	$\text{Bin}(0; N, 1/2)$
3	≥ 2	0	$\text{Bin}(n; N, 1/2)$
4	≥ 2	≥ 2	0
5	1	1	0
6	1	≥ 2	$\text{Bin}(1; N, 1/2)$
7	≥ 2	1	$\text{Bin}(n-1; M, 1/2)$
8	≥ 2	≥ 2	$\text{Bin}(k; N, 1/2)$

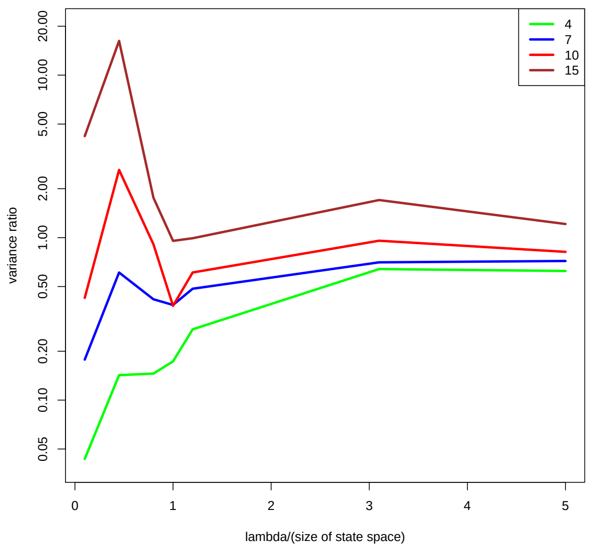

Figure 6: Variance reduction using importance sampling. The variance ratio is the ratio between the variance from importance sampling and the variance from bisection.

5.3 Quasi Monte Carlo

We finally consider a quasi Monte Carlo approach. We only draw a pseudo-random number to choose midpoint and the number of jumps in the two intervals. The remaining part of the bisection algorithm is as before. In Figure 7 we compare the two sampling schemes. It is quite clear that QMC is a very efficient strategy to improve the convergence rate for the algorithm.

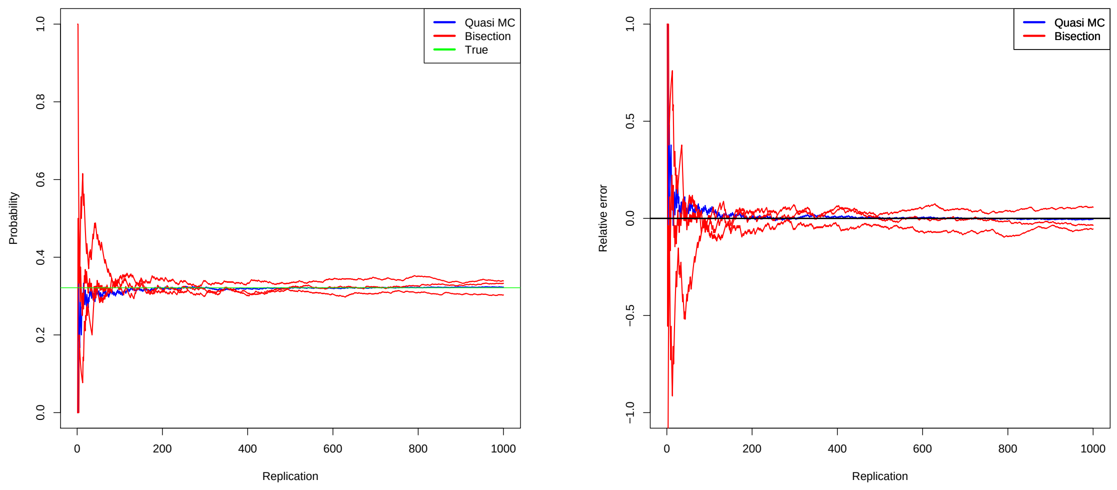

Figure 7: Quasi Monte Carlo. Right: Raw estimation of the probability for one cycle. Left: Relative error (defined as (true-obs)/true).

6 Conclusions and extensions

1. As mentioned in the Introduction, we do not believe that bisection in itself is a major improvement of existing methods for simulating CTMC bridges, but that the justification of the method rather is its potential for variance reduction. We find this potential well illustrated via the numerical examples, stressing that these are rather crude by only using variance reduction methods for the midpoint $T/2$. A substantial improvement is to be expected if in addition one incorporates $T/4$, $3T/4$ and so on. For stratification, this is unfeasible for even moderate state spaces, since the order of strata increases from $4N$ to $(4N)^3$ by just going from $T/2$ to $T/2$, $T/4$, $3T/4$. However, the situation is much better for importance sampling and quasi Monte Carlo, and in particular, such an extension could well dramatically change the somewhat disappointing behavior of importance sampling.

2. Another extension not implemented here is hybrid algorithms where the bisection is only used to generate say $X(T/2)$, $X(1/4)$, $X(3T/4)$ (and possibly the number of jumps in each of the 4 intervals), to apply variance reduction techniques ideas to these r.v.'s only and generate the rest of the sample path by some other algorithm, say rejection sampling which is superior when the endpoint conditioning is not rare.

3. A phase-type distribution is the distribution of the absorption time τ of a Markov process X^* on $\{0, 1, \ldots, M\}$, where 0 is absorbing and the states in $1, \ldots, M$ non-absorbing, and having some specified initial probabilities ξ_a, $a = 1, \ldots, M$. In simulation-based statistical estimation, one needs to generate a sample path of X^* conditioned on $\tau = T$. An algorithm is suggested in [7] and uses Gibbs sampling.

   The problem can, however, be translated to endpoint conditioned simulation. To this end, one simply computes the probability η_b that $X^*(\tau-) = b$

(this reduces to simple matrix algebra but we omit the details). One then draws a, b ccording to the ξ_a and η_b, and simulates X^* conditioned to have endpoints a, b and no transitions to state 0 in $[0, T]$.

4. Another potential application of the bisection algorithm is in combination with the uniformization algorithm. To this end, one first notes that since it is not essential to split intervals into two of exactly equal size, our algorithm applies with minor changes to discrete time Markov chains, in this case the chain at Poisson times. Doing so has the potential advantage that a segment of length K where the Markov chain is constant can be simulated in a single step instead of K steps. This is appealing in situations where the q_i are of different orders of magnitudes, since then segments with large K are likely to show up in the sample path.

7 Appendix

Consider a CTMC $\{X(t) : 0 \leq t \leq T\}$ with rate matrix Q and endpoints $X(0) = a$ and $X(T) = b$. In this Appendix we provide a recursion for the number of substitutions using the approach suggested by [28].

Consider a uniformization of the process. Let

$$R = I + \tfrac{1}{\mu} Q,$$

where $\mu = \max_c Q_c$. Furthermore, let J denote the (stochastic) number of jumps (including virtual) and N the (stochastic) number of substitutions (excluding the virtual jumps). Siepel, Pollard and Haussler's formula for the number of substitutions is based on the following fundamental observation

$$\begin{aligned} P(n, b|a, T) &= P(N(T) = n, X(T) = a|X(0) = a) \\ &= \sum_{j=n}^{\infty} P(N(T) = n, X(T) = b, J(T) = j|X(0) = a) \\ &= \sum_{j=n}^{\infty} P(N(T) = n, X(T) = b|J(T) = j, X(0) = a) P(J(T) = j|X(0) = a) \\ &= \sum_{j=n}^{\infty} P(n, b|j, a) \operatorname{Pois}(j|\mu T). \end{aligned} \tag{7}$$

Here $\operatorname{Pois}(\cdot|\mu T)$ is the Poisson distribution with rate μT. Note that $P(n, b|j, a)$ does not depend on the time interval. Also note that we can find the transition probabilities from

$$P_{ab}(T) = P(b|a, T) = \sum_{n=0}^{\infty} P(n, b|a, T).$$

This formula provides a way of calculating the transition probability without using a diagonalization of the rate matrix.

Having calculated $P(n, b|a, T)$ we can easily find the distribution for the number of endpoint-conditioned substitutions

$$P(N(T) = n|X(0) = a, X(T) = b) = \frac{P(n, b|a, T)}{P(b|a, T)}.$$

The crucial step for (7) to be useful is a fast way of calculating the quantities $P(n, b|j, a)$, and [28] provide a recursion for accomplishing this task.

First note that $P(n, b|j, a) = 0$ if $n > j$.

For $j = 0$ we have

$$P(n, b|j = 0, a) = \begin{cases} 1 & \text{if } a = b \text{ and } n = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

This provides the basis of the recursion.

For $j \geq 1$ we find $P(n, b|j, a)$ for $0 \leq n \leq j$ using the recursion

$$\begin{aligned} P(n, b|j, a) &= P(N = n, Y(j) = b|J = j, Y(0) = a) \\ &= P(N = n, Y(j) = b, Y(j-1) = b|J = j, Y(0) = a) \\ &\quad + \sum_{c \neq b} P(N = n, Y(j) = b, Y(j-1) = c|J = j, Y(0) = a) \\ &= R_{bb} P(n, b|j-1, a) + R_{cb} P(n-1, c|j-1, a), \end{aligned} \tag{9}$$

where Y is the uniformized (auxiliary) process that includes the virtual jumps.

The actual implementation of the recursion is described in the following algorithm:

1. **Initialization**. Fix a to the desired value and calculate basis of recursion using (8). Set $j = 1$.
2. **Recursion**. Define matrix $M_j(b, n)$ with number of rows equal to the size of the state space and $(j + 1)$ columns. Calculate entries $M_j(b, n) = P(n, b|a, j)$ using (9).
3. **Stopping Criteria**. If $\sum_{n,b} M_j(b, n) = 1$ to machine precision, then stop. Otherwise set $j = j + 1$ and go to 2.

References

[1] Asmussen, S. and Glynn, P.W. (2007). *Stochastic Simulation. Algorithms and Analysis* Springer-Verlag.

[2] Asmussen, S. and Hobolth, A. (2008). Bisection ideas in end-point conditioned Markov process simulation. In *Proceedings of the 7th International Workshop on Rare Event Simulation* (G. Rubino and B. Tuffin, eds.), 121–130. Rennes, France.

[3] Avramidis, A.N., L'Ecuyer, P. and Tremblay, P.-A. (2003) Efficient simulation of gamma and variance-gamma processes. In *Proceedings of the 2003 Winter Simulation Conference* (S. Chick *et al.*, eds.).

[4] Beskos, A., Papaspiliopoulos,O. and Roberts, G.O. (2006) Retrospective exact simulation of diffusion sample paths with applications. *Bernoulli* **12**, 1077–1098.

[5] Beskos, A., Papaspiliopoulos, O., Roberts, G. O. (2008a). A factorisation of diffusion measure and finite sample path constructions. *Methodol. Comput. Appl. Probab.* **10**, 85–104

[6] Beskos, A., Roberts, G.O., Stuart, A. and Voss, J. (2008b) MCMC methods for diffusion bridges. *Stoch. Dyn.* **8**, 319–350.

[7] Bladt, M., Gonzales, A. and Lauritzen, S.L. (2003). The estimation of phase-type related functionals using Markov chain Monte Carlo. *Scand. Act. J.* **4**, 280–300.

[8] Bladt, M. and Sørensen, M. (2009) Simple simulation of diffusion bridges with application to likelihood inference for diffusions. `http://www.math.ku.dk/~michael/diffusionbridge0809.pdf`

[9] Caflish, R.E. and Moskowitz, B. (1995) Modified Monte Carlo methods using quasi-random sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing* (H. Niederreiter and P.J.-S. Shine, eds.). Lecture Notes in Statistics **106**, 1–16. Springer-Verlag

[10] Caflish, R.E., Morokoff, W. and Owen, A.B. (1997) Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *J. Comput. Finance* **1**, 27–46.

[11] Ewens, W.J. and Grant, G.R. (2001). *Statistical methods in Bioinformatics.* Springer-Verlag.

[12] Fearnhead, P. and Sherlock, C. (2006). An exact Gibbs sampler for the Markov-modulated Poisson process. *J.R. Statist. Soc. B* **68**, 767–784.

[13] Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* **17**, 368–376.

[14] Felsenstein, J. (2004). *Inferring Phylogenies.* Sinauer Associates, Inc.

[15] Fitzsimmons, P., Pitman, J. and Yor, M. (1992) Markovian bridges: construction, Palm interpretation and splicing. In *Seminar on Stochastic Processes.* Progress in Probability **32**, 101–134.

[16] Hobolth, A. (2008) A Markov chain Monte Carlo expectation maximization algorithm for statistical analysis of DNA sequence evolution with neighbour-dependent substitution rates. *Journal of Computational and Graphical Statistics* **17**, 138–164.

[17] Hobolth, A. and Stone, E.A. (2009). Efficient simulation from finite-state, continuous-time Markov chains with incomplete observations. *Ann. Appl. Statist.* **3**, 1204–1231.

[18] Leobacher, G. (2006) Stratified sampling and quasi-Monte Carlo simulation of Lévy processes. *Monte Carlo Methods and Applications* **12**, 231–238

[19] Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* **16**, 111–120.

[20] Leobacher, G. (2006). Stratified sampling and quasi-Monte Carlo simulation of Levy processes. Monte-Carlo methods and applications (12), Nr. 3-4, pp. 231–238.

[21] Metzner, P., Dittmer, E., Jahnkea, T. and Schüttea, C. (2007). Generator estimation of Markov jump processes. *Journal of Computational Physics* **227**, 353–375.

[22] Moler, C. and Van Loan, C. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review* **45**, 3–49.

[23] Moskowitz, B. and Caflish, R.E. (1996) Smoothness and dimension reduction in quasi-Monte Carlo methods. *Journal of Mathematical and Computer Modeling* **23**, 37–54.

[24] Nielsen, R. (2002). Mapping mutations on phylogenies. *Syst. Biol.*, **51**, 729–739.

[25] Ribeiro, C. and Webber, N. (2003) Valuing path-dependent options in the variance-gamma model by Monte Carlo with a gamma bridge. *J. Comput. Finance* **7**, 81–100.

[26] Ribeiro, C. and Webber, N. (2006) Correction for simulation bias in Monte Carlo methods to value exotic options in models driven by Lévy processes. *Appl. Math. Finance* **13**, 333–352.

[27] Roberts, G.O. and Stramer, O. (2001) On inference for partially observed nonlinear diffusion processes using Metropolis-Hastings algorithms. *Biometrika* **88**, 603–621.

[28] Siepel, A., Pollard, K.S. and Haussler, D. (2006). New methods for detecting lineage-specific selection. *Proceedings of the 10th International Conference on Research in Computational Molecular Biology (RECOMB)*, 190–205.