

# Bisection Ideas in End-Point Conditioned Markov Process Simulation

Søren Asmussen and Asger Hobolth

Department of Mathematical Sciences, Aarhus University

Ny Munkegade, 8000 Aarhus C, Denmark

`{asmus, asger}@imf.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 consider the problem of generating sample paths from a continuous-time Markov chain conditioned on the endpoints. In particular we suggest a new algorithm, based on the idea of bisection. The bisection algorithm is compared to previously suggested sample path algorithms, and a possible role for the algorithm in phase-type distributions is discussed.

## 1 Introduction

In many applications of continuous time Markov chains, the stochastic process  $\{X(t) : t \geq 0\}$  is sampled in discrete points in time  $T_0 = 0 < T_1 < \dots < T_{n-1} < T_n = T$ , while the process itself is a continuous-time process. In order to proceed with inference and analysis it can be useful to simulate sample paths between the datapoints. As a consequence of the Markov assumption, knowledge of the data  $X(T_0), \dots, X(T_n)$  partitions the problem into  $n$  independent problems where independent sample paths  $\{X(t) : T_i < t < T_{i+1}\}$ ,  $i = 0, \dots, n-1$ , must be generated between the timepoints  $T_i$  and  $T_{i+1}$ , conditional on the datapoints  $X(T_i)$  and  $X(T_{i+1})$ . In this paper we propose a new algorithm for endpoint conditional sampling from a continuous-time Markov chain defined on a discrete and finite state space.

The bisection algorithm is presented in detail in Section 3, 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 2, 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.

In Section 4 we briefly describe three other algorithms for endpoint conditional sampling from continuous-time Markov chains. These other three algorithms are *rejection sampling*, *uniformization* and *direct simulation*. In Section 5 we compare the four algorithms. We find that the bisection algorithm does not compare favorably with the other algorithms. The reason for the rather poor performance of the bisection algorithm comes from the fact that it is often likely to have two jumps close in time. If such an event occurs, many splits of the time interval is needed before the two jumps are separated, resulting in many recursion steps.

We end the paper by discussing possible applications of the bisection algorithm in phase-type distributions and for speeding up the uniformization algorithm.

## 2 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:  $X(t) = a$ ,  $0 \leq t \leq T$ .
2. If  $X(0) = a$  and  $X(T) = b \neq a$  and there is precisely one jump we are basically done:  $X(t) = a$ ,  $0 \leq t < \tau$ , and  $X(t) = b$ ,  $\tau \leq t \leq T$ .

In 2, the jump time  $\tau$  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 use the notation  $Q = \{q_{ab}\}$  for the instantaneous rate matrix with off-diagonal entries  $q_{ab} \geq 0$  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 algorithms require the transition probabilities  $P_{ab}(t)$ , i.e. the elements of the transition matrix  $P(t) = \exp(Qt)$ . These can easily be computed, for example, if  $Q$  can be written in diagonal form  $UDU^{-1}$  with  $D = \text{diag}(\lambda_i)$ ; then  $P(t) = U \text{diag}(\exp(\lambda_i t)) U^{-1}$ . For different methods, see the classical paper by Moler and Van Loan (2003).

**Lemma 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*

$$R_{ab}(T) = q_{ab} \begin{cases} \frac{e^{-q_a T} - e^{-q_b T}}{q_b - q_a} & q_a \neq q_b \\ T e^{-q_a T} & q_a = q_b. \end{cases} \quad (1)$$

The density of the state change is

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

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

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

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

**Remark 1** 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. Asmussen and Glynn, 2007, p. 39). Finally, 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  $[0, T]$ .  $\square$

## 3 Bisection Algorithm

The algorithm involves an initialization step and a recursive step. The recursive step is easy once the initialization step is explained.

### 3.1 Initialization

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.

#### 3.1.1 Initialization when the end-points are equal

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

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

The first term can be further dissected into

$$\begin{aligned}
P_{aa}(T/2) &= P(X(T/2) = a | X(0) = a) \\
&= P(X(T/2) = a, N(T/2) = 0 | X(0) = a) + P(X(T/2) = a, N(T/2) \geq 2 | X(0) = a) \\
&= e^{-q_a T/2} + [P_{aa}(T/2) - e^{-q_a T/2}], \tag{3}
\end{aligned}$$

and similarly the second term can be written as

$$P_{ac}(T/2) = R_{ac}(T/2) + [P_{ac}(T/2) - R_{ac}(T/2)]. \tag{4}$$

With the abbreviation  $e_a = e^{-q_a T/2}$ ,  $r_{ab} = R_{ab}(T/2)$ ,  $p_{ab} = P_{ab}(T/2)$  we obtain Table 1 when substituting (3) and (4) into (2).

case	number of jumps in first interval	number of jumps in second interval	(unconditional) probability	notation
1	0	0	$e_a e_a$	$\alpha_1$
2	0	$\geq 2$	$e_a (p_{aa} - e_a)$	$\alpha_2$
3	$\geq 2$	0	$(p_{aa} - e_a) e_a$	$\alpha_3$
4	$\geq 2$	$\geq 2$	$(p_{aa} - e_a)(p_{aa} - e_a)$	$\alpha_4$
5	1	1	$r_{ac} r_{ba}$	$\alpha_{5,c}$
6	1	$\geq 2$	$r_{ac} (p_{ca} - r_{ca})$	$\alpha_{6,c}$
7	$\geq 2$	1	$(p_{ac} - r_{ac}) r_{ca}$	$\alpha_{7,c}$
8	$\geq 2$	$\geq 2$	$(p_{ac} - r_{ac})(p_{ca} - r_{ca})$	$\alpha_{8,c}$

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

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  $\alpha$ -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 1. 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  $\geq 2$ , 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 1 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.

### 3.1.2 Initialization when the end-points 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.

case	number of jumps in first interval	number of jumps in second interval	(unconditional) probability	notation
1	0	1	$e_a r_{ab}$	$\beta_1$
2	0	$\geq 2$	$e_a(p_{ab} - r_{ab})$	$\beta_2$
3	$\geq 2$	1	$(p_{aa} - e_a)r_{ab}$	$\beta_3$
4	$\geq 2$	$\geq 2$	$(p_{aa} - e_a)(p_{ab} - r_{ab})$	$\beta_4$
5	1	0	$r_{ab}e_b$	$\beta_5$
6	1	$\geq 2$	$r_{ab}(p_{bb} - e_b)$	$\beta_6$
7	$\geq 2$	0	$(p_{ab} - r_{ab})e_b$	$\beta_7$
8	$\geq 2$	$\geq 2$	$(p_{ab} - r_{ab})(p_{bb} - e_b)$	$\beta_8$
9	1	1	$r_{ac}r_{cb}$	$\beta_{9,c}$
10	1	$\geq 2$	$r_{ac}(p_{cb} - r_{cb})$	$\beta_{10,c}$
11	$\geq 2$	1	$(p_{ac} - r_{ac})r_{cb}$	$\beta_{11,c}$
12	$\geq 2$	$\geq 2$	$(p_{ac} - r_{ac})(p_{cb} - r_{cb})$	$\beta_{12,c}$

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

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  $\beta$ -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  $\geq 2$  jumps means that further simulation is needed. In the next sub-section, we discuss the recursive part of the bisection algorithm.

## 3.2 Recursion and termination

When an interval with  $\geq 2$  jumps is entered, further simulation is needed. However, it is straight-forward 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  $\geq 2$  jumps are present.

## 3.3 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  $\geq 2$  state changes to the next level.

## 4 Previous algorithms

Hobolth and Stone (2008) 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 Hobolth and Stone (2008).

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  $X(0) = a$ , and the path is accepted if  $X(T) = b$ . Nielsen (2002) 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. Niensens 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. Fearnhead and Sherlock, 2006), 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* (Hobolth, 2008) 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.

## 5 Comparison

We conducted a small experiment to investigate the performance of the bisection algorithm compared to rejection sampling, uniformization and direct simulation.

The rate matrix is given by

$$Q = (1/10) \begin{bmatrix} -11 & 6 & 3 & 2 \\ 4 & -9 & 3 & 3 \\ 2 & 3 & -9 & 4 \\ 2 & 3 & 6 & -11 \end{bmatrix} \quad (5)$$

and  $X(0) = 1$  and  $X(T) = 2$ . The scaling of the rate matrix is such that one substitution is expected in one time unit when the beginning state is drawn from the stationary distribution.

In Figure 1 we show the CPU-time spent by the four algorithms on simulating 100 independent sample paths. The performance of the bisection algorithm is rather poor, especially when the time between the endpoints is large. The reason being that when the time difference is large, it is rather likely to have two closely spaced jumps. When two jumps are closely spaced, many splits of the large starting time interval are needed before the two jumps are separated. The many recursion steps in the bisection algorithm is CPU time expensive.

It is not too difficult to calculate the probability of reaching a certain recursion level. Suppose, for ease of exposition, that we are at level  $\ell = 2$  where the interval  $[0, T]$  is divided into  $2^2 = 4$  time intervals. Denote the internal states  $(X(T/4), X(T/2), X(3T/4)) = (k, l, m)$ . As previously, the endpoints are  $X(0) = a$  and  $X(T) = b$ . The probability of the algorithm stopping before or at level 2 is then

$$\frac{1}{P_{ab}(T)} \sum_k \sum_l \sum_m R_{ak}(T/4) R_{kl}(T/4) R_{lm}(T/4) R_{mb}(T/4),$$

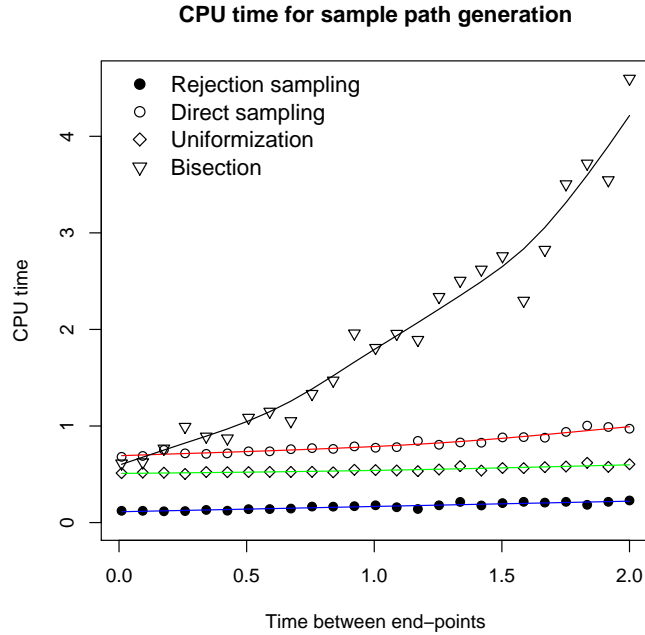


Figure 1: CPU time versus time between endpoints for the rate matrix (5). Four sample path algorithms are compared: rejection sampling, uniformization, direct simulation and the bisection algorithm. The CPU time is the cumulated time for generating 100 independent samples. The beginning state is  $X(0) = 1$ , and the ending state is  $X(T) = 2$ . Rejection sampling is very fast because the acceptance probability is high. Uniformization is also rather fast because the diagonal entries are of the same order, resulting in very few virtual state changes. Direct sampling is penalized for its time-consuming calculations of the next state and the waiting time before the state change. The bisection algorithm is slow because quite a few of the 100 independent samples have two state changes close in time. When two state changes are close in time, many recursive steps are required. The probability of having two closely spaced jumps increases when the time between the two end-points increases.

where  $R_{ak}(t), a \neq k$ , is the probability (1) that the interval  $[0, t]$  contains exactly one jump from  $a$  to  $k$ , and  $R_{aa}(t) = \exp(-q_a t)$  is the probability that the interval does not have any jumps. This sum can be efficiently calculated in a recursive manner.

In Figure 2 we show the probabilities of reaching recursion levels  $\ell = 1, \dots, 15$  for the rate matrix (5) with beginning state  $X(0) = 1$  and ending state  $X(T) = 2$ , and for various time distances  $T = 0.1, 0.5, 1.0, 1.5, 2.0$ . For the small time interval  $T = 0.1$ , the probability of reaching level 5 is smaller than 0.001, and therefore it is rather fast to use the bisection algorithm for this time interval. However, when the time interval increases, the probability of reaching a high recursion level also increases. When  $T \geq 1$  we reach a recursion level of at least 6 with probability larger than 0.005. This high probability

explains the poor performance of the bisection algorithm for large time distances.

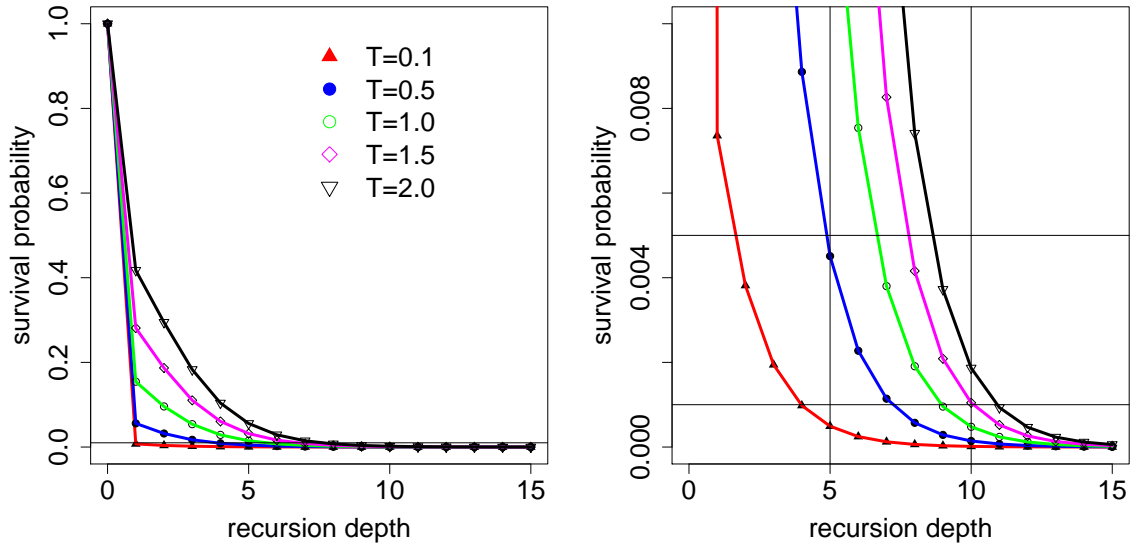


Figure 2: Probability of the bisection algorithm reaching a certain recursion level for various time distances  $T$ . The rate matrix is given by (5) and the endpoints are  $X(0) = 1$  and  $X(T) = 2$ . For small time distances  $T$  the probability of reaching a large recursion level is small, but when the time distance between the endpoints is large the probability of reaching a large recursion level is no longer small.

## 6 Extensions

A phase-type distribution is the distribution of the absorption time  $\tau$  of a Markov process  $X^*$  on  $\{0, 1, \dots, M\}$ , where 0 is absorbing and the states in  $1, \dots, M$  non-absorbing, and having some specified initial probabilities  $\pi_a$ ,  $a = 1, \dots, M$ . In simulation-based statistical estimation, one needs to generate a sample path of  $X^*$  conditioned on  $\tau = T$ . An algorithm is suggested in Bladt *et al.* (2003) and uses Gibbs sampling.

The problem can, however, be translated to endpoint conditioned simulation. To this end, one simply computes the probability  $\eta_b$  that  $X^*(\tau-) = b$  (this reduces to simple matrix algebra but we omit the details). One then draws  $a, b$  according to the  $\pi_a$  and  $\eta_b$ , and simulates  $X^*$  conditioned to have endpoints  $a, b$  and no transitions to state 0 in  $[0, T]$ .

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.

## References

- Asmussen, S. and Glynn, P.W. (2007). *Stochastic Simulation. Algorithms and Analysis* Springer-Verlag.
- 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.
- Fearnhead, P. and Sherlock, C. (2006). An exact Gibbs sampler for the Markov-modulated Poisson process. *J.R. Statist. Soc. B* **68**, 767-784.
- 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.
- Hobolth, A. and Stone, E.A. (2008). Efficient simulation from finite-state, continuous-time Markov chains with incomplete observations. *Ann. Appl. Statist.* (pending revision).
- 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.
- Nielsen, R. (2002). Mapping mutations on phylogenies. *Syst. Biol.*, **51**, 729-739.