PMRL Technical Report 93-4, Department of Electrical and Computer Engineering, University of Arizona, January 1993 (Also Memoranda Informatica 92-22, Computer Science Department, University of Twente) Submitted for Publication.

Adaptive Uniformization: Technical Details[*]

Aad P. A. van Moorsel
University of Twente
Tele-Informatics and Open Systems
P.O. Box 217, 7500 AE Enschede, The Netherlands
moorsel@cs.utwente.nl

William H. Sanders
University of Arizona
Department of Electrical and Computer Engineering
Tucson AZ 85721
whs@ece.arizona.edu

ABSTRACT

This paper discusses technical aspects related to adaptive uniformization [1]. We give the derivation of the schemes that we have used to compute the jump probabilities. We also give some more detailed results about the complexity of adaptive uniformization. This paper is not self contained, but is a supplement to [1].

[*]A. P. A. van Moorsel has been supported by the Netherlands Organization for Scientific Research (NWO), grants R62-385 and SIR13-1260. W. H. Sanders has been supported, in part, by the Digital Faculty Program: Incentives for Excellence.

Figure 1: The CTMC of an AU-jump process.

I Introduction

Adaptive uniformization (AU) is a generalization of standard uniformization (SU), and leads to the following expression for the transient state probability vector $\underline{\pi}(t)$

$$\underline{\pi}(t) = \underline{\pi}(0) \sum_{n=0}^{N} U_n(t) \prod_{i=0}^{n-1} P_i = \sum_{n=0}^{N} U_n(t) \underline{\pi}_n. \tag{1}$$

The computation of $U_n(t)$ is done with the acyclic Markov chain evaluator (ACE) method [3]. For the case of AU with converged rate this scheme is modified in [1]. In Section II we will give the formal derivation of this modified ACE schemes. In [1] also is reported about the computational complexity of computing (1). In Section III we will give some more details about how the computational results have been derived and about system characteristics that influence the complexity of AU and SU.

II Modified ACE schemes

The problem for which we have used the ACE scheme is the computation of the probability of exactly n jumps in the so-called AU-jump process, which is a pure birth process, with possibly non-identical rates. Let the pure birth process $B = \{B(t); t \geq 0\}$ be defined over the state space $S = \{0, 1, 2, ...\}$ with intensities λ_{n-1} for the n-th jump, $n = 1, 2, 3, ...$. See Figure 1 for the corresponding continuous time Markov chain (CTMC). $U_n(t)$ is now expressed as $U_n(t) = \Pr\{B(t) = n\}$. Let $J(n)$ be the number of *different* values in the set $\{\lambda_0, \lambda_1, ..., \lambda_n\}$ and let $\gamma_j, j = 1, ..., J(n)$, be the different parameter *values*. Further $K_j(n) + 1$ is defined as the number of parameters in $\{\lambda_0, \lambda_1, ..., \lambda_n\}$ which equal γ_j at epoch n. Finally, j_n is the value of j such that $\gamma_{j_n} = \lambda_n$.

Example. To illustrate the meaning of these parameters we give them for an example of a machine repairman model with three components (see [1]). The CTMC of the AU-jump process is depicted in Figure 2 and the corresponding parameter values are given in Table 1.

Before we derive the ACE schemes, we present the following result, which will be used throughout the paper.

Figure 2: The CTMC of the AU-jump process for the machine repairman model with three components.

n	λ_n	γ_n	$J(n)$	$K_1(n)+1$	$K_2(n)+1$	$K_3(n)+1$	j_n
0	3ν	—	1	1	—	—	1
1	$\mu+2\nu$	3ν	2	1	1	—	2
2	$\mu+\nu$	$\mu+2\nu$	3	1	1	1	3
3	$\mu+2\nu$	$\mu+\nu$	3	1	2	1	2
4	$\mu+2\nu$	—	3	1	3	1	2
5	$\mu+2\nu$	—	3	1	4	1	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: Illustration of the meaning of the parameters used in the ACE scheme for a machine repairman model with three components.

Proposition 1 ***Integral expression.***

$$\int_0^t e^{\alpha s} s^k \frac{(-\alpha)^{k+1}}{k!} ds = 1 - e^{\alpha t} \sum_{l=0}^{k} \frac{(-\alpha t)^l}{l!}, k \geq 0, \alpha \in \Re. \tag{2}$$

Derivation. For $k = 0$ (1) follows directly. For $k \geq 1$, integration by parts gives

$$\int_0^t e^{\alpha s} s^k \frac{(-\alpha)^{k+1}}{k!} ds = -\frac{(-\alpha)^k}{k!} e^{\alpha t} t^k + \int_0^t e^{\alpha s} s^{k-1} \frac{(-\alpha)^k}{(k-1)!} ds = \ldots \tag{3}$$

$$\ldots = -\sum_{l=1}^{k} \frac{(-\alpha)^l}{l!} e^{\alpha t} t^l + \int_0^t -\alpha e^{\alpha s} ds = 1 - e^{\alpha t} \sum_{l=0}^{k} \frac{(-\alpha t)^l}{l!}. \tag{4}$$

□

Note that for $\alpha < 0$ formula (4) is the Erlang probability distribution function with $k+1$ phases.

We divide this section in Section A on ACE for a generic birth process, Section B on ACE for AU with converged rate (C-ACE) and Section C on ACE for a converged AU process with all non-converged rates different (CD-ACE). In Table 2 we give the different ACE schemes that are of interest for the particular birth processes we consider for AU.

Technique	Converged rate	Different rates in non-converged part
ACE	Free	Free
C-ACE	Yes	Free
CD-ACE	Yes	Yes

Table 2: ACE schemes.

A ACE for a generic birth process

For a class of so-called right-shifted systems of differential equations, Severo [2] has derived a generic solution scheme. For a system that represents an acyclic Markov chain, Marie *et al.* [3] have derived the ACE scheme, which essentially is a special case of the solution method of Severo. For the case of a birth process, [1] gives the appropriate ACE scheme, which in turn is a special case of the ACE scheme of Marie *et al.* Since the way of presentation and the used notation in [3] differs from [1], and since the proof is constructive on itself, we repeat a proof for the generic ACE scheme in Proposition 2. The ACE scheme can also be found in [4] with a notation similar to the one we use here.

Proposition 2 ***Generic ACE.*** *$U_n(t)$ can be computed by the following ACE scheme:*

$$U_n(t) = \sum_{j=1}^{J(n)} \sum_{k=0}^{K_j(n)} a_{j,k}^{(n)} e^{-\gamma_j t} t^k, \quad \text{for } n \geq 0 \text{ and } t \geq 0, \tag{5}$$

with for $n = 0$

$$a_{1,0}^{(0)} = 1, \tag{6}$$

while for $n = 1, 2, \ldots$ the following recursive relations hold for the coefficients $a_{j,k}^{(n)}$:

$$a_{j,K_j(n)}^{(n)} = -a_{j,K_j(n)}^{(n-1)} \frac{\lambda_{n-1}}{(\gamma_j - \lambda_n)}, \qquad \text{when } j \neq j_n; \tag{7}$$

$$a_{j,k}^{(n)} = \frac{k+1}{(\gamma_j - \lambda_n)} a_{j,k+1}^{(n)} - \frac{\lambda_{n-1}}{(\gamma_j - \lambda_n)} a_{j,k}^{(n-1)}, \qquad \text{when } j \neq j_n, \text{ for } k = 0, 1, \ldots, K_j(n) - 1; \tag{8}$$

$$a_{j_n,k}^{(n)} = \frac{\lambda_{n-1}}{k} a_{j_n,k-1}^{(n-1)}, \qquad \text{for } k = 1, \ldots, K_{j_n}(n); \tag{9}$$

$$\text{and} \quad a_{j_n,0}^{(n)} = -\sum_{j=1|j\neq j_n}^{J(n)} a_{j,0}^{(n)}. \tag{10}$$

The values of $J(n)$ and $K_j(n)$ are as defined earlier in this section, and possibly non-defined terms are assumed to be equal to 0.

Derivation. First, for $n = 0$ we have $\gamma_1 = \lambda_0$, $J(0) = 1$, $K_1(0) + 1 = 1$ and thus

$$U_0(t) = \sum_{j=1}^{J(0)} \sum_{k=0}^{K_j(0)} a_{j,k}^{(0)} e^{-\gamma_j t} t^k = a_{1,0}^{(0)} e^{-\gamma_j t} = e^{-\gamma_1 t}. \tag{11}$$

By induction we show that expression (5) holds for all $n = 1, 2, \ldots$. Then we show that if we have the coefficients on epoch $n-1$, the coefficients for epoch n obey the given recursive relations.

From Kolmogorov's forward equation we see that

$$U_n(t) = \int_0^t U_{n-1}(s)\lambda_{n-1}e^{-\lambda_n(t-s)}ds. \tag{12}$$

Assume that

$$U_{n-1}(t) = \sum_{j=1}^{J(n-1)} \sum_{k=0}^{K_j(n-1)} a_{j,k}^{(n-1)} e^{-\gamma_j t} t^k, \text{ for some } n \geq 1, \tag{13}$$

then it follows from (12) that

$$U_n(t) = \int_0^t \sum_{j=1}^{J(n-1)} \sum_{k=0}^{K_j(n-1)} a_{j,k}^{(n-1)} e^{-\gamma_j s} s^k \lambda_{n-1} e^{-\lambda_n(t-s)} ds \tag{14}$$

$$= \sum_{j=1}^{J(n-1)} \sum_{k=0}^{K_j(n-1)} a_{j,k}^{(n-1)} \lambda_{n-1} e^{-\lambda_n t} \int_0^t e^{-(\gamma_j - \lambda_n)s} s^k ds. \tag{15}$$

We evaluate the term $e^{-\lambda_n t} \int_0^t e^{-(\gamma_j-\lambda_n)s} s^k ds$ for $j = 1, 2, \ldots, J(n-1)$, subsequently for j such that $\lambda_n = \gamma_j$ and such that $\lambda_n \neq \gamma_j$. For $\lambda_n = \gamma_j$ we have

$$e^{-\lambda_n t} \int_0^t e^{-(\gamma_j-\lambda_n)s} s^k ds = e^{-\lambda_n t} \frac{t^{k+1}}{k+1}. \tag{16}$$

For $\lambda_n \neq \gamma_j$ we have, by Proposition 1 for $\alpha = -(\gamma_j - \lambda_n)$ (note that Proposition 1 holds both when $(\gamma_j - \lambda_n)$ is positive and negative)

$$e^{-\lambda_n t} \int_0^t e^{-(\gamma_j-\lambda_n)s} s^k ds = e^{-\lambda_n t} \frac{k!}{(\gamma_j - \lambda_n)^{k+1}} (1 - e^{-(\gamma_j-\lambda_n)t} \sum_{l=0}^{k} \frac{((\gamma_j - \lambda_n)t)^l}{l!}) \tag{17}$$

$$= \frac{k!}{(\gamma_j - \lambda_n)^{k+1}} e^{-\lambda_n t} - \frac{k!}{(\gamma_j - \lambda_n)^{k+1}} e^{-\gamma_j t} \sum_{l=0}^{k} \frac{(\gamma_j - \lambda_n)^l t^l}{l!}. \tag{18}$$

When there exists no $j_n \leq J(n-1)$ such that $\gamma_{j_n} = \lambda_n$, it should hold that $J(n) = J(n-1)+1$; $K_j(n) = K_j(n-1)$ for all $j = 1, 2, \ldots, J(n-1)$; and $K_{j_n}(n)+1 = 1$ (we define in this case $K_{j_n}(n-1) = -1$). A new term for $j = J(n), k = 0$ and $e^{-\lambda_n t}$ appears in (18). If there exists a $j_n \leq J(n-1)$ such that $\gamma_{j_n} = \lambda_n$, it should hold that $J(n) = J(n-1)$; $K_j(n) = K_j(n-1)$ for all $j \neq j_n$,; and $K_{j_n}(n) = K_{j_n}(n-1)+1$. In (16) a new term for $j = j_n, k = K_{j_n}(n)$ with $e^{-\lambda_n t} t^{K_{j_n}(n)}$ is then introduced. Therefore, the number of terms, i.e. the values of $J(n)$ and $K_j(n)$ for $j = 1, 2, \ldots, J(n)$, in the expression for $U_n(t)$ are according to the proposition. So, when for $n = 1, 2, \ldots$, the formula (5) for $U_{n-1}(t)$ holds, it does for $U_n(t)$. That for $n = 0$, $U_0(t)$ obeys (5) completes the induction argument. We now show that the coefficients obey the given recursive relations.

Combining (16) and (18) the following expression for $U_n(t)$ is formed from (15)

$$U_n(t) = \sum_{k=0}^{K_{j_n}(n-1)} a_{j_n,k}^{(n-1)} \frac{\lambda_{n-1}}{k+1} e^{-\lambda_n t} t^{k+1} \tag{19}$$

$$+ \sum_{j=1|j\neq j_n}^{J(n-1)} \sum_{k=0}^{K_j(n-1)} a_{j,k}^{(n-1)} \lambda_{n-1} \frac{k!}{(\gamma_j - \lambda_n)^{k+1}} e^{-\lambda_n t} \tag{20}$$

$$- \sum_{j=1|j\neq j_n}^{J(n-1)} \sum_{k=0}^{K_j(n-1)} a_{j,k}^{(n-1)} \lambda_{n-1} \frac{k!}{(\gamma_j - \lambda_n)^{k+1}} e^{-\gamma_j t} \sum_{l=0}^{k} \frac{(\gamma_j - \lambda_n)^l t^l}{l!}. \tag{21}$$

From (19) it follows that, for $k = 1, 2, ..., K_{j_n}(n)$,

$$a_{j,k}^{(n)} = a_{j,(k-1)}^{(n-1)} \frac{\lambda_{n-1}}{k}. \tag{22}$$

Note that $K_{j_n}(n) = K_{j_n}(n-1)+1$. From (21), observing that in (21) there is a term $e^{-\gamma_j t} t^{k^*}$ whenever l sums at least to k^*, i.e. there is a new term for $k = k^*, k^*+1, ..., K_j(n-1)$, $0 \leq k^* \leq K_j(n-1)$. So, for $j \neq j_n$

$$a_{j,k}^{(n)} = - \sum_{r=k}^{K_j(n-1)} a_{j,r}^{(n-1)} \lambda_{n-1} \frac{r!}{(\gamma_j - \lambda_n)^{r+1}} \frac{(\gamma_j - \lambda_n)^k}{k!}. \tag{23}$$

Note that $K_j(n) = K_j(n-1)$ for $j \neq j_n$. Now, when $l = K_j(n)$ we take $r = k = K_j(n)$ in (23), leading to

$$a_{j,K_j(n)}^{(n)} = -a_{j,K_j(n)}^{(n)} \frac{\lambda_{n-1}}{(\gamma_j - \lambda_n)}, \tag{24}$$

and, for $k = 0, 1, ..., K_j(n) - 1, j \neq j_n$, to

$$a_{j,k}^{(n)} = -a_{j,k}^{(n-1)} \frac{\lambda_{n-1}}{(\gamma_j - \lambda_n)} + a_{j,k+1}^{(n)} \frac{k+1}{(\gamma_j - \lambda_n)}. \tag{25}$$

Note particulary the varying index n and $n-1$ in de coefficients in this deduction. Finally, by taking $k = 0$ in (23) we obtain from (20)

$$a_{j_n,0}^{(n)} = - \sum_{j=1|j\neq j_n}^{J(n-1)} a_{j,0}^{(n)}. \tag{26}$$

Now (22), (24), (25) and (26) equal (9), (7), (8) and (10) respectively, so the recursive relations for the coefficients obey Proposition 2.

$\square$

Conclusion. The ACE scheme can be used to compute the probabilities $U_n(t)$ in AU. ACE has order complexity $O(N^2)$, N being the number of jumps in the AU process. An important aspect is the numerical stability of the ACE scheme, which we however do not address in this report. According to [4] ACE can only be used for birth processes with until 30 different paramater values. For AU in its most generic form this can form a serieus drawback.

Figure 3: The CTMC for an AU-jump process with converged rate.

B C-ACE for AU with converged rate

In this subsection we derive the scheme for AU with converged rate. In Figure 3 a corresponding birth process, or AU-jump process, is depicted. So, from some epoch $m \geq 1$ on the uniformization rate remains μ. We first give a recursive relation for the hypo-exponential density in Proposition 3. This we use in Proposition 4 to derive an expression for $U_n(t)$ with $n = m + l$, for values $l \geq 0$. A recursive scheme for the computation of $U_{m+l}(t)$ is then provided in Proposition 5 and 6.

Proposition 3 ***Hypo-exponential distribution.*** *The density $f_H^{(n)}(t), t \geq 0$ for a hypo-exponential random variable with $n \geq 1$ phases, is given by*

$$f_H^{(n)}(t) = \sum_{j=1}^{J^-(n)} \sum_{k=0}^{K_j(n)} b_{j,k}^{(n)} e^{-\gamma_j t} t^k, \tag{27}$$

with parameters as in Section A, except that $J^-(n)$ now denotes the number of different parameter values in the set $\{\lambda_0, \lambda_1, ..., \lambda_{n-1}\}$.

Derivation. Define $F_H^{(n)}(t)$ to be the hypo-exponential distribution with n phases, so $dF_H^{(n)}(t)/dt = f_H^{(n)}(t)$. Now, if we take $\lambda_n = 0$, it follows from the definition of $U_n(t)$ and Proposition 2, for $n = 1, 2, ...$, that

$$F_H^{(n)}(t) = \sum_{j=1}^{J(n)} \sum_{k=0}^{K_j(n)} a_{j,k}^{(n)} e^{-\gamma_j t} t^k. \tag{28}$$

Then we have for $f_H^{(n)}(t)$

$$f_H^{(n)}(t) = \frac{F_H^{(n)}(t)}{dt} = \sum_{j=1}^{J(n)} \sum_{k=0}^{K_j(n)} a_{j,k}^{(n)}(-\gamma_j) e^{-\gamma_j t} t^k + \sum_{j=1}^{J(n)} \sum_{k=1}^{K_j(n)} a_{j,k}^{(n)} k e^{-\gamma_j t} t^{k-1}. \tag{29}$$

Now we can take, depending on the value of k, the following expressions for the coefficients $b_{j,k}^{(n)}$ of (27)

$$b_{j,0}^{(n)} = -a_{j,0}^{(n)} \gamma_j, \tag{30}$$

$$b_{j,k}^{(n)} = -a_{j,k}^{(n)} \gamma_j + a_{j,k+1}^{(n)}(k+1), \text{ for } k = 1, 2, ..., K_j(n) - 1, \tag{31}$$

$$b_{j,K_j(n)}^{(n)} = -a_{j,k}^{(n)} \gamma_j. \tag{32}$$

Notice that we have assumed $\lambda_n = 0$, and thus $b_{j_n,0}^{(n)} = -a_{j_n,0}^{(n)} \lambda_n = 0$, which justifies the parameter $J^-(n)$ in (27).

$\square$

The preceding deduction does not construct the recursive scheme to compute the coefficients $b_{j,k}^{(n)}$ of (3), but only shows that they exist. A constructive approach, similar to the derivation of the coefficients in Proposition 2, is possible too.

Proposition 4 ***Expression for*** $U_{m+l}(t)$. *Let the hypo-exponential density with m phases be given by the expression (27). For AU with converged rate μ after m jumps, $U_{m+l}(t)$ is given, for $l = 0, 1, 2, ..,$ by*

$$U_{m+l}(t) = \sum_{k=0}^{K_{j_\mu}(m)} \sum_{r=0}^{k} b_{j_\mu,k}^{(m)} (-1)^r \frac{k!}{(k-r)!r!} \frac{\mu^l}{l!(r+l+1)} e^{-\mu t} t^{k+l+1} \tag{33}$$

$$+ \sum_{j=1|j \neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (-1)^r \frac{k!}{(k-r)!r!} \frac{\mu^l}{(\mu-\gamma_j)^{r+l+1}} \frac{(r+l)!}{l!} e^{-\gamma_j t} t^{k-r} \tag{34}$$

$$- \sum_{j=1|j \neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (-1)^r \frac{k!}{(k-r)!r!} \frac{\mu^l}{(\mu-\gamma_j)^{r+l+1}} \frac{(r+l)!}{l!} e^{-\mu t} t^{k-r} \sum_{v=0}^{r+l} \frac{(\mu-\gamma_j)^v t^v}{v!}, \tag{35}$$

with j_μ denoting the value of j for which $\gamma_j = \mu$.

Derivation. We use that the birth process is constructed out of two parts; the first m jumps are such that the probability of having m jumps before time t is hypo-exponentially distributed, the last l jumps behave as a Poisson process, when looked upon seperately. Therefore we get

$$U_{m+l}(t) = \int_0^t f_H^{(m)}(t-s) \frac{(\mu s)^l}{l!} e^{-\mu s} ds \tag{36}$$

$$= \int_0^t \sum_{j=1}^{J^-(m)} \sum_{k=0}^{K_j(m)} b_{j,k}^{(m)} e^{-\gamma_j (t-s)} (t-s)^k \frac{(\mu s)^l}{l!} e^{-\mu s} ds. \tag{37}$$

Using $(t-s)^k = \sum_{r=0}^k \frac{k!}{r!(k-r)!} t^{k-r} (-s)^r$ we obtain

$$U_{m+l}(t) = \sum_{j=1}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (-1)^r \frac{\mu^l}{l!} \frac{k!}{(k-r)!r!} e^{-\gamma_j t} t^{k-r} \int_0^t s^{r+l} e^{-(\mu-\gamma_j)s} ds. \tag{38}$$

We now use Proposition 1 with $\alpha = -(\mu - \gamma_j)$, and divide (38) in terms for j such that $\mu = \gamma_j$ and $\mu \neq \gamma_j$. Note that it is possible that μ also is one of the rates in the first m non-converged jumps.

$$U_{m+l}(t) = \sum_{k=0}^{K_{j_\mu}(m)} \sum_{r=0}^{k} b_{j_\mu,k}^{(m)} (-1)^r \frac{k!}{(k-r)!r!} \frac{\mu^l}{l!} e^{-\mu t} t^{k-r} \frac{t^{r+l+1}}{(r+l+1)} \tag{39}$$

$$+ \sum_{j=1|j \neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (-1)^r \frac{k!}{(k-r)!r!} \frac{\mu^l}{l!} \frac{(r+l)!}{(\mu-\gamma_j)^{r+l+1}} e^{-\gamma_j t} t^{k-r} \tag{40}$$

$$-\sum_{j=1|j\neq j_\mu}^{J^-(m)}\sum_{k=0}^{K_j(m)}\sum_{r=0}^{k} b_{j,k}^{(m)}(-1)^r\frac{k!}{(k-r)!r!}\frac{\mu^l}{(\mu-\gamma_j)^{r+l+1}}\frac{(r+l)!}{l!}e^{-\mu t}t^{k-r}\sum_{v=0}^{r+l}\frac{(\mu-\gamma_j)^v t^v}{v!}. \quad (41)$$

From (39), (40) and (41), the equations (33), (34) and (35) in the proposition directly follow.

□

Proposition 5 ***Alternative expression for*** $U_{m+l}(t)$***.*** *Define, with* $j,k,r,l \in \aleph$, *and* $\zeta, t \in \Re$,

$$A_{j,k,r,t}^{(\zeta)}(l) = b_{j,k}^{(m)}(-1)^r\frac{k!}{(k-r)!r!}\frac{\mu^l}{(\mu-\gamma_j)^{r+l+1}}\frac{(r+l)!}{l!}e^{-\zeta t}t^{k-r}, \quad (42)$$

$$B_{k,r,t}(l) = b_{j_\mu,k}^{(m)}(-1)^r\frac{k!}{(k-r)!r!}\frac{\mu^l}{l!(r+l+1)}e^{-\mu t}t^{k+l+1}, \quad (43)$$

$$C_{j,r,t}(l) = \sum_{v=0}^{r+l}\frac{(\mu-\gamma_j)^v t^v}{v!}. \quad (44)$$

Then $U_{m+l}(t)$ *can be expressed as*

$$U_{m+l}(t) = \sum_{j=1|j\neq j_\mu}^{J^-(m)}\sum_{k=0}^{K_j(m)}\sum_{r=0}^{k}[A_{j,k,r,t}^{(\gamma_j)}(l) - A_{j,k,r,t}^{(\mu)}(l)C_{j,r,t}(l)] + \sum_{k=0}^{K_{j_\mu}(m)}\sum_{r=0}^{k}B_{k,r,t}(l). \quad (45)$$

Derivation. By inspection of (33), (34) and (35).

□

Proposition 6 ***Recursive scheme C-ACE.*** *Define, with* $j,r,l \in \aleph, t \in \Re$,

$$D_{j,r,t}(l) = \frac{(\mu-\gamma_j)^{r+l}}{(r+l)!}t^{r+l}. \quad (46)$$

Then the A, B *and* C*-terms, with appropriate indices, from Proposition 5 can be computed recursively in the parameter* l *as follows*

$$A_{j,k,r,t}^{(\zeta)}(l) = A_{j,k,r,t}^{(\zeta)}(l-1)\frac{\mu}{(\mu-\gamma_j)}\frac{(r+l)}{l}; \quad (47)$$

$$B_{k,r,t}(l) = B_{k,r,t}(l-1)\frac{\mu}{l}\frac{(r+l)}{(r+l+1)}t; \quad (48)$$

$$C_{j,r,t}(l) = C_{j,r,t}(l-1) + D_{j,r,t}(l), \quad (49)$$

with

$$D_{j,r,t}(l) = D_{j,r,t}(l-1)\frac{(\mu-\gamma_j)}{r+l}t. \quad (50)$$

Derivation. By inspection of (42), (43), (44) and (46).

□

Conclusion. When the adaptive uniformization rate in AU converges to some value μ from epoch $m \geq 1$ on, the computation of $U_{m+l}(t)$ can be done by the C-ACE scheme which is recursive in l. Therefore, the computation of $U_{m+l}(t), l = 1, 2, ...,$ will ask for a constant number of computations, independent of the actual value of l, given that $U_{m+l-1}(t)$ has been computed by the C-ACE scheme also. So, C-ACE has order of complexity $O(N)$, N being the number of jumps that has to be carried out in AU. Also of interest is the numerical stability of this C-ACE algorithm. The conclusion in [4] that for 30 different parameter values the ACE scheme does not suffer from numerical inadeqacies probably can be extrapolated to the C-ACE scheme.

C CD-ACE for AU with converged rate and all non-converged rates different

In this section we look at an AU-jump process in which the rate remains equal to μ after $m \geq 1$ epochs, and in which all the non-converged rates $\lambda_i, i = 0, 1, ..., m-1$, are *different.* The resulting recursive CD-ACE scheme for these birth processes is even simpler than the preceding C-ACE scheme. Furthermore, it is possible to use a closed-form expression for the hypo-exponential density function when all phases have different parameter values.

We first derive the CD-ACE scheme, and then give the closed-form expression for the hypo-exponential distribution with all phases having different parameters.

Proposition 7 ***CD-ACE scheme.*** *Define, with $b_j^{(m)}$ instead of $b_{j,0}^{(m)}$ in (27),*

$$A_{j,t}^{(\zeta)}(l) = b_j^{(m)} \frac{\mu^l}{(\mu - \gamma_j)^{l+1}} e^{-\zeta t} \tag{51}$$

$$B_t(l) = b_{j_\mu}^{(m)} \frac{\mu^l}{(l+1)!} e^{-\mu t} t^l \tag{52}$$

$$C_{j,t}(l) = \sum_{s=0}^{l} \frac{(\mu - \gamma_j)^s t^s}{s!}. \tag{53}$$

Then the following expression exists for $U_{m+l}(t), l = 0, 1, ...,$ when all non-converged rates are different and not equal to the converged rate μ

$$U_{m+l}(t) = \sum_{j=1|j \neq j_\mu}^{m} [A_{j,t}^{(\gamma_j)}(l) - A_{j,t}^{(\mu)}(l) C_{j,t}(l)] + B_t(l). \tag{54}$$

Moreover, define

$$D_{j,t}(l) = \frac{(\mu - \gamma_j)^l}{l!} t^l, \tag{55}$$

then the following recursive relations exist

$$A_{j,t}^{(\zeta)}(l) = \frac{\mu}{(\mu - \gamma_j)} A_j^{(\zeta)}(l-1), \tag{56}$$

$$B_t(l) = \frac{\mu t}{(l+1)} B_t(l-1), \tag{57}$$

$$C_{j,t}(l) = C_{j,t}(l-1) + D_{j,t}(l), \tag{58}$$

$$D_{j,t}(l) = \frac{(\mu - \gamma_j)}{l} D_{j,t}(l-1). \tag{59}$$

Derivation. The fact that all rates λ_i are different for $i = 0, 1, ..., m-1$, implies that all $\gamma_j, j = 1, 2, ..., m$ are different. Moreover, by the way we have defined the parameters γ_j, we have $\gamma_j = \lambda_{j-1}, j = 1, 2, ..., m$. When all rates are different, we get that $K_j(m)$ is equal to 0 for all j. So, the summation over k, and consequently over r, disappears out of Equation (45) for $U_{m+l}(t)$. For the A, B and C-terms in (42), (43) and (44) $r = k = 0$ has to be filled in, thus leading to the expressions (51), (52), (53) and (54). The recursive relations (56), (57) and (58) can be found by inspection of (51), (52) and (53).

□

Proposition 8 ***Closed form expression.*** *When the uniformization rate $\lambda_i, i = 0, 1, ..., m-1$, are all different, the following closed-form solution for $U_n(t)$ with $n = 0, 1, ..., m-1$ exists*

$$U_n(t) = \sum_{j=1}^{n+1} a_j^{(n)} e^{-\gamma_j t}, \tag{60}$$

with coefficients $a_1^{(0)} = 1$, and for $n = 1, 2, ..., m-1$,

$$a_j^{(n)} = \frac{\Pi_{i=1}^{n} \gamma_i}{\Pi_{i=1|i\neq j}^{n+1} (\gamma_i - \gamma_j)}. \tag{61}$$

Derivation. For $n = 0$ we have $a_1^{(0)} = 1$, and (60) follows immediately. For the correctness for $n \geq 1$, we first notice that $U_n(t) = F_H^{(n)}(t) - F_H^{(n+1)}(t)$, when $F_H^{(n)}(t)$ is the n-phase hypo-exponential distribution function. We use that a hypo-exponential random variable with n phases and all parameter values different has a closed-form density function $f_H^{(n)}(t)$ as follows [5, p.580].

$$f_H^{(n)}(t) = \sum_{j=1}^{n} b_j^{(n)} e^{-\gamma_j t}, \tag{62}$$

with

$$b_j^{(n)} = \frac{\prod_{i=1}^{n} \gamma_i}{\prod_{i=1|i\neq j}^{n} (\gamma_i - \gamma_j)}. \tag{63}$$

Define

$$c_j^{(n)} = \frac{b_j^{(n)}}{\gamma_j} = \prod_{i=1|i\neq j}^{n} \frac{\gamma_i}{(\gamma_i - \gamma_j)}, \tag{64}$$

then we have for $F_H^{(n)}(t)$

$$F_H^{(n)}(t) = \int_0^t \sum_{j=1}^{n} b_j^{(n)} e^{-\gamma_j s} ds = \sum_{j=1}^{n} \frac{b_j^{(n)}}{\gamma_j}(1 - e^{-\gamma_j t}) = \sum_{j=1}^{n} c_j^{(n)} (1 - e^{-\gamma_j t}). \tag{65}$$

Because $F_H^{(n)}(t)$ is a distribution function the limit for $t \to \infty$ equals 1, and thus we have that for all $n \geq 1$

$$\sum_{j=1}^{n} c_j^{(n)} = 1, \tag{66}$$

and thus that

$$F_H^{(n)}(t) = 1 - \sum_{j=1}^{n} c_j^{(n)} e^{-\gamma_j t}. \tag{67}$$

Using

$$c_j^{(n+1)} - c_j^{(n)} = \prod_{i=1|i\neq j}^{n+1} \frac{\gamma_i}{(\gamma_i - \gamma_j)} - \prod_{i=1|i\neq j}^{n} \frac{\gamma_i}{(\gamma_i - \gamma_j)} \tag{68}$$

$$= \prod_{i=1|i\neq j}^{n} \frac{\gamma_i}{(\gamma_i - \gamma_j)} \left(\frac{\gamma_{n+1}}{(\gamma_{n+1} - \gamma_j)} - 1\right) = \frac{\Pi_{i=1}^{n} \gamma_i}{\Pi_{i=1|i\neq j}^{n+1} (\gamma_i - \gamma_j)} = a_j^{(n)}, \tag{69}$$

and

$$c_{n+1}^{(n+1)} = \prod_{i=1|i\neq n+1}^{n+1} \frac{\gamma_i}{(\gamma_i - \gamma_j)} = \frac{\Pi_{i=1}^{n} \gamma_i}{\Pi_{i=1|i\neq n+1}^{n+1} (\gamma_i - \gamma_j)} = a_{n+1}^{(n)}, \tag{70}$$

we finally get that

$$U_n(t) = F_H^{(n)}(t) - F_H^{(n+1)}(t) = 1 - \sum_{j=1}^{n} c_j^{(n)} e^{-\gamma_j t} - (1 - \sum_{j=1}^{n+1} c_j^{(n+1)} e^{-\gamma_j t}) \tag{71}$$

$$= \sum_{j=1}^{n} (c_j^{(n+1)} - c_j^{(n)}) e^{-\gamma_j t} + c_{n+1}^{(n+1)} e^{-\gamma_{n+1} t} = \sum_{j=1}^{n} a_j^{(n)} e^{-\gamma_j t} + a_{n+1}^{(n)} e^{-\gamma_{n+1} t} = \sum_{j=1}^{n+1} a_j^{(n)} e^{-\gamma_j t}. \tag{72}$$

□

Conclusion. We have derived a recursive scheme, called CD-ACE, for the computation of the jump probabilities $U_{m+l}(t), l = 0, 1, ...,$ when the rate in the AU process converges after m jumps, and all non-converged rates are different. The complexity of the scheme is $O(N)$, N being the number of jumps, and is thus equal to the complexity of C-ACE. However, per iteration only a small amount of computations has to be carried out, so the "constant" of the scheme is smaller. Furthermore, one can use the derived closed-form expression to compute $U_n(t)$ for $n < m$.

III Complexity aspects of AU

In [1] results for the computational complexity of AU and SU have been presented for an example of an extended machine repairmen (EMR) model. We discuss in subsection A in some more detail, compared to [1], how these results have been derived, and what assumptions and choices have been made. Then, in subsection B, we will relax from the EMR example and discuss in general how different problem aspects contribute to the computational complexity of AU.

A Derivation of the computational complexity

The following three *complexity factors*, as we will call them, determine the computational complexity of AU and SU:

- Complexity of the computation of the jump probabilities $U_n(t)$ (JPR);
- Complexity of the generation of P_n (GEN);
- Complexity of the matrix vector multiplication $\underline{\pi}_n = \underline{\pi}_{n-1} P_{n-1}$ (MVM).

To derive results for the complexity of the individual complexity factors, we only need to know the sizes of the matrices Q_n, and the necessary number of jumps, respectively, for AU and SU.

We first introduce some notation. We take $N_a(t)$ and $N_s(t)$ to be the necessary number of jumps to reach the desired accuracy ϵ with AU and SU respectively, given that the time of interest is t. Furthermore, we define $N_\Delta(t) = N_s(t) - N_a(t)$. The number of non-zero elements of Q_n is denoted by η_n, the sum of the number of diagonal elements η_n^d and the number of non-zero off-diagonal elements η_n^o. For the infinitesimal generator Q we leave out the subscript n: $\eta = \eta^d + \eta^o$. In this case η^d equals the state space size, while η^o equals the number of possible transitions. By looking only at the non-zero off-diagonal elements of the matrices, we implicitly assume a sparse matrix technique to be used. To describe the individual contribution of the three complexity factors we give the number of instructions as JPR, GEN and MVM. The subscript a denotes the number of instructions with AU, s denotes the number of instructions with SU. Again we denote dependency on the parameter t between brackets. So, we can have for example $JPR_a(t)$, the number of instructions needed to compute $U_n(t)$ for AU. Furthermore, we will use the Δ-sign to denote the difference of complexity between SU and AU, and the %-sign to denote the relative difference with respect to AU. So, for instance we get $JPR_\Delta(t) = JPR_s(t) - JPR_a(t)$ and $JPR_{\%}(t) = JPR_\Delta(t)/JPR_a(t)$.

Basic assumptions. For the computation of the complexity we have made the following starting assumptions:

- Complexity of the algorithm is solely determined by the number of arithmetic instructions necessary to carry out the calculation;
- The influence of practical problems such as underflow and overflow management, round-off errors, genericity of the algorithms, etc, on the computational complexity is not taken into account.

These assumptions make that for an actual software implementation of the algorithm the resulting complexity will most probably differ. However, for the *qualitative* judgement of the performance of the different uniformization algorithms, as was the aim in [1], this way of establishing the complexity suffies in our opinion. Moreover, implemented software will give *other* information, which not necessarily leads to a better understanding of the performance of an algorithm. Our discussion will be as much as possible implementation insensitive. We will now discuss the derivation of the results on numerical complexity for the EMR model in [1]. Subsequently the complexity factors JPR, GEN and MVM will

Factor	+ and -	* and /	Total
JPR_a, epoch $n < m$	$4n+2$	$2n$	$6n+2$
JPR_a, epoch $n \geq m$	$4m$	$4m$	$8m$
JPR_s	0	2	2
GEN_a, epoch $n \in I(t)$	η_n^d	$\eta_n^d + \eta_n^o$	$2\eta_n^d + \eta_n^o$
GEN_s, epoch $n = 0$	η^d	$\eta^d + \eta^o$	$2\eta^d + \eta^o$
MVM_a	$\eta_n^o + \eta_n^d$	η_n^o	$2\eta_n^o + \eta_n^d$
MVM_s	$\eta^o + \eta^d$	η^o	$2\eta^o + \eta^d$

Table 3: Complexity in the number of instructions at epoch n.

be discussed. The results for the number of operations at epoch n are summarized in Table 3. From left to right the columns respectively give the discussed complexity factor, the number of additions and subtractions, the number of multiplications and divisions, and the total number of operations to be carried out at epoch n.

Complexity of computing the jump probabilities $U_n(t)$

Assumptions. Computing the jump probabilities, both for AU and SU, can give numerical problems. For computing the Poisson probabilities in SU, a numerical stable scheme has been derived by Fox *et al.* [6]. For computing the coefficients of the ACE scheme, prevention of overflow and underflow problems can be achieved, in part, by implementation choices as in [4]. We will not consider the influence of these schemes on the computational complexity, as it is very problem specific, and can best be determined by implementing and running the algorithms. We estimate the computational costs when the jump probabilities are computed straightforwardly, i.e. without additional stability increasing schemes.
SU. For SU the jump probabilities are Poisson probabilities. After performing the multiplication λt once, for every step two operations are necessary to compute $T(n) = (\lambda t)^n/n! = T(n-1)\lambda t/n$. Neglecting the cost of computing $e^{-\lambda t}$, which has to be carried out only once, we have $JPR_s(t) = 2N_s(t)$ operations. Note that we do not consider the possibility to do a "left truncation" [4, 6].
AU. We derive the computational cost of the CD-ACE scheme, as this scheme can be used to derive the jump probabilities for the EMR model in [1]. For $n = 0, ..., m-1$, i.e. for steps with non-converged rates, equations (60) and (61) have to be computed. See Table 3 for the results. Although this number of operations becomes of neglicible importance for the complexity when the number of jumps increases, it is of importance for the evaluation of how AU compares to SU for values of t with $N_a(t) < m$. The computation per step after the rate has converged is determined by (56), (57), (58), (59) and (54), and is also given in Table 3.

Especially for the computation of the ACE schemes the obtained results have to be treated purely as *indications* for the complexity. Contrary to the complexity factors GEN and MVM, programming choices for AU are completely different from those for SU. An example of such a choice is the use of the recursive relation between $a_j^{(n)}$ and $a_j^{(n-1)}$ in the computation of (61), which we did assume in our instruction counting.

Complexity of generating P_i

Assumptions. We assume that the matrix Q is known in advance, and thus do not deal with the generation process itself.
SU. SU asks for η^d divisions and subtractions to compute the diagonal elements $1 - q_i/\lambda$ of P, and for η^o operations to compute the off-diagonal elements $q(i,j)/\lambda$. This has to be done once. So, $GEN_s(t) = 2\eta^d + \eta^o$, which is independent of t. In Table 3 we have denoted this as computation "at epoch $n = 0$".
AU. At every step that a new matrix P_i has to be computed it asks for $2\eta_i^d + \eta_i^o$ operations. Let $I(t)$ be the set of indices consisting of all numbers of the jumps in which a new matrix has to be computed. Then $GEN_a(t) = \sum_{i \in I(t)} (2\eta_i^d + \eta_i^o)$, which is dependent of t as it is not for all t necessary to compute all the different P_i.

Complexity of matrix vector multiplications

Assumptions. In a straightforward implementation of SU there are no considerations about active states [1], which implies that the matrix vector multiplications (MVM) is always carried out with full size matrices. The notion of active states in AU makes that it works with matrices in MVM which are as small as possible, and therefore MVM asks for less computational effort for AU than SU. On the other hand, keeping track of which states are active asks for bookkeeping, thus leading to overhead. We do not deal with this trade-off in our analysis, but because it is possible, and possibly beneficial, to incorporate the notion of active states into SU, we do so. This implies that when both methods do an equal number of steps, the complexity of MVM is equal.
SU. MVM computes $\underline{\pi}_n = \underline{\pi}_{n-1} P$ and is the most expensive complexity factor. To keep the explanation and notation concise, we first assume that in every step the full size matrices P are used. Define $\eta^o(j)$ to be the number of non-zero off-diagonal element in the j-th column of P. Note that $\sum_{j=1}^{\eta^d} \eta^o(j) = \eta^o$. As the diagonal elements of P will mainly be non-zero, we assume that the j-th column has $\eta^o(j) + 1$ non-zero elements. To compute the j-th element of $\underline{\pi}_n$ will then ask for $\eta^o(j) + 1$ multiplications and $\eta^o(j)$ additions. Summation over $j, j = 1, ..., \eta_d$, then gives a total of $\eta^o + \eta^d$ multiplications and η^o additions for every epoch in SU. So, $MVM_s(t) = N_s(t)(2\eta^o + \eta^d)$.

Now, we did assume that minimum size matrices will be used also in SU. Then the above analysis has to be carried out for P_n at epoch n. Neglecting the few cases of zero's in the vector $\underline{\pi}_{n-1}$ we then obtain $MVM_s(t) = \sum_{n=0}^{N_s(t)} (2\eta_n^o + \eta_n^d)$.
AU. Similar to the derivation for SU we have $MVM_a(t) = \sum_{n=0}^{N_a(t)} (2\eta_n^o + \eta_n^d)$.

Conclusion

We have illustrated how the complexity results for AU and SU have been derived for the EMR model in [1]. We note here that the curves in [1] are based on the total number of operations necessary, without distinguishing between the different arithmetic operators. So, the results in the column "Total" of Table 3 have been used. To derive over-all results for the complexity of the uniformization schemes, the needed number of steps has to be computed. In [1] we have used equations (60) and (61) to derive the number of jumps for values of t for which the uniformization rate had not yet converged. For higher values of t

Metric	Value
$N_a(t)$	$\lambda(t - t_m)$
$N_s(t)$	λt
$N_\Delta(t)$	λt_m
$N_\%(t)$	$\frac{t_m}{t-t_m}$

Table 4: Number of jumps for large t.

	Instructions JPR	Instructions GEN	Instructions MVM
$Factor_a(t)$	$N_a(t)8m$	$\sum_{n\in I(\infty)}(2\eta_n^d + \eta_n^o)$	$N_a(t)(2\eta^o + \eta^d)$
$Factor_s(t)$	$N_s(t)2$	$2\eta^d + \eta^o$	$N_s(t)(2\eta^o + \eta^d)$
$Factor_\Delta(t)$	$(2 - 8m)\lambda t + 8m\lambda t_m$	$GEN_s(t) - GEN_a(t)$	$\lambda t_m(2\eta^o + \eta^d)$
$Factor_\%(t)$	$\frac{2t}{8m(t-t_m)} - 1$	$GEN_\Delta(t)/GEN_a(t)$	$\frac{t_m}{(t-t_m)}$

Table 5: Total complexity in the number of instructions, summed over all epochs.

results for the number of jumps were derived using central limit theorems. This has been explained in [1].

B The influence of problem aspects on computational complexity

We have seen in subsection A what factors contribute to the computational complexity of AU and SU. In this section we will discuss which *sources* influence the computational complexity, i.e., which elements of the problem determine the number of jumps and the number of elements in the matrices. To derive general results for the influence of problem aspects we use results based on central limit theorems. These can be found in [1], and in subsection B.1 we first will repeat some of the results. In subsection B.2 we will then discuss the influence of problem aspects on the performance of the two uniformization algorithms.

B.1 Order of complexity results

In this section we give order of complexity results when t becomes large. In subsection A detailed counting of instructions for small values of t was necessary to be able to distinguish the computational complexity of AU and SU in that area. Here we can use results from central limit theorems (see the appendix of [1]) to derive order of complexity results for large t. We give the complexity results for the following case:

- AU has a converged rate after m steps;
- The converged rate of AU equals the uniformization rate with which SU is carried out.

Let t_m be the mean time the m non-converged jumps take:

$$t_m = \sum_{i=0}^{m-1} 1/\lambda_i. \tag{73}$$

time instant	absolute difference	relative difference
$t < t^c$	gain AU	high gain AU
$t^c < t < t^*$	gain AU decreasing	gain AU decreasing
$t = t^*$	AU and SU equal	AU and SU equal
$t > t^*$	loss AU increasing	loss AU increasing
$t \to \infty$	loss AU to infinity	loss AU to constant

Table 6: Gain and loss of AU with respect to SU, as function of time t of interest.

Then we can derive for the necessary number of steps in the uniformization schemes the results of Table 4. Table 5 gives the *total* number of instructions necessary to carry out the uniformization algorithm for the corresponding needed number of epochs (note the relation between Table 5 and Table 3). In the first two rows the results for AU and SU are given, expressed in $N_a(t)$ and $N_s(t)$. In the third and fourth row the differences in complexity are given, now with the results of Table 4 filled in for $N_a(t)$ and $N_s(t)$.

We see that both $MVM_\Delta(t) \to \lambda t_m(2\eta^o - \eta^d)$ and $GEN_\Delta(t)$ converge (for $GEN_\Delta(t)$ the limiting value depends on how many different matrices have to be computed). On the other hand, $JPR_\Delta(t)$ is a decreasing function in t: $JPR_\Delta(t) = (2-8m)\lambda t + 2\lambda t_m$. When t is small, the overhead for AU due to computing the jump probabilities and the generation of the matrices will be outweighed by the fact that the number of jumps for AU is less than for SU. This makes that the matrix vector multiplication will ask less effort, as has been illustrated in [1]. The implication of this is that AU outperforms SU for small t. However, with increasing t, the gain because of MVM will be close to the limiting constant value in Table 5, while the overhead for JPR increases linear in t. Consequently, there exists a *turning point* t^* such that AU outperforms SU for $t < t^*$, while SU outperforms AU for $t > t^*$.

Let the time instant t^c denote the minimum value of t for which the number of jumps $N_a(t^c)$ is such that the converged rate has been reached. It is for values of $t < t^c$ for which using AU really pays off, as can be seen from the speed up factor of 1000 and more for the EMR model in [1]. The reason for this is that $N_a(t) \ll N_s(t)$ for $t < t^c$. In the computation of the complexity we can neglect the contribution of GEN, as generation of the matrices only has to be carried out a finite number of times, and not for every step. We therefore can derive results for the difference in complexity by only looking at JPR and MVM. For t large, the absolute difference $JPR_\Delta(t) + MVM_\Delta(t)$ goes to infinity as the term $(2-8m)\lambda t$ decreases linear in t:

$$JPR_\Delta(t) + MVM_\Delta(t) = (2-8m)\lambda t + (8m + 2\eta^o + \eta^d)\lambda t_m \to -\infty, \tag{74}$$

as can be derived from the results in Table 5. Using $JPR_a(t) + MVM_a(t) = (8m + 2\eta^o + \eta^d)\lambda(t - t_m)$ we get that the relative difference in complexity with respect to AU converges to a constant:

$$\frac{JPR_\Delta(t) + MVM_\Delta(t)}{JPR_a(t) + MVM_a(t)} = \frac{(2-8m)}{(8m + 2\eta^o + \eta^d)} \frac{t}{(t - t_m)} + \frac{t_m}{t - t_m} \to \frac{2 - 8m}{8m + 2\eta^o + \eta^d}. \tag{75}$$

Finally, we use the approximate value of the turning point t^*, which is the value of t for which $JPR_\Delta(t) = MVM_\Delta(t)$:

$$t^* = \frac{8m + 2\eta^o + \eta^d}{8m - 2} t_m. \tag{76}$$

We have to take notice of the limited validity of the central limit theorems if the value of t^* is not large compared to t_m. In that case Equation (76) cannot be used.

In Table 6 a qualitative assessment about the difference in complexity between AU and SU is given for different values of t. This table summarizes the behavior of the difference in complexity as function of t.

B.2 Problem aspects

In this subsection we discuss what problem aspects influence the computational complexity, and to what extend. The discussion will especially be about the influence on the value of the turning point, and the differences between AU and SU in both absolute and relative sence. We identify three main sources that influence the computational complexity:

1. Model sources:
   - State space size and number of possible transitions;
   - Rates of transitions;
2. Implementation sources:
   - (Adapted) uniformization rates;
3. Requirements sources:
   - Desired accuracy ϵ;
   - Set of t of interest.

Model sources

The *state space size* equals η^d, while the number of *transitions* equals η^o. So, the magnitude of the model directly influences the computational complexity caused by MVM and GEN (see Table 5). On the other hand, the state space size does not influence JPR, and for this reason we have that when the state space increases, the turning point t^* will increase according to (76). By enlarging the state space, it will be very easy to construct an example in which the turning point is much higher than in the EMR model.

The *rates of the transitions* determine the uniformization rates. When the rates increase, and thus the uniformization rates increase, the necessary number of steps in both AU and SU will increase. We see from (76) that the turning point is depending on the uniformization rate through t_m (see (73)). To derive results for the turning point when the central limit theorems leading to Equation (76) are not valid demands more intricate analysis. This also is the case when we want to derive results for the complexity for small values of t. Additional experiments should provide insight in the complexity for values of t for which the central limit theorems give no accurate results. Note that the relative difference is not influenced by the uniformization rate λ, see (75), while the absolute difference increases proportional to λ, see (74). Also the absolute difference for values of $t < t^c$ will increase when the uniformization rate increases. This is because for $t < t^c$ the computational cost for AU are almost neglicible, while the cost for SU are directly determined by the necessary number of steps, which is for SU proportional to the uniformization rate.

In this respect we will shortly elaborate on the class of *stiff* CTMC's. There are several definitions of stiff models, but roughly models will be stiff when the uniformization rate λ is large. To solve the problem with SU is then not very attractive, as the necessary number of jumps $N_s(t)$ is of the order $O(\lambda t)$. Several solution methods have been developed to overcome this problem [7, 8, 9], see for an overview [10]. Most widely applicable is the method in [10] (see also [7] for an extension towards cumulative rewards). In this method it is checked whether the result of the uniformization process is close to steady-state, after which the uniformization procedure can be stopped. Dunkel *et al.* [11] show that this approach is not useful for so-called *nearly completely decomposable* models. The EMR model of [1] is, depending on the value of the coverage factor c, such a model [11]. Checking for steady-state is in this case useless, but the AU extension of uniformization can limit the computational cost. We stress here that this is especially the case for a limited class of models. When t is not too large (e.g. $t \leq t^c$), AU can then considerably speed up the solution.

Implementation sources

To get the ACE scheme as simple as possible, it is important to have a converged rate. Moreover, when the adapted uniformization rate is taken to be converged earlier, it will lessen the complexity of the jump probability computation. On the other hand it will ask for more steps, and thus for more MVM computation. For every value t of interest there is an optimal pattern of adapted uniformization rates, in the sense that it minimizes the computational complexity. For the EMR example, it typically seems to be smart to let the rates be converged after r jumps at the latest, i.e., when the first active state is reached that allows repairs and thus has a higher outgoing rate.

Requirements sources

Increasing the *accuracy* makes that the number of jumps for AU as well as SU becomes higher. It implies that $N_a(t)$ and $N_s(t)$ both are higher when t is small. So, t^c will decrease when the desired accuracy increases. But, it follows from the central limit theorem (see [1]) that for large t the number of jumps is independent of ϵ. So, for the turning point t^* the influence of ϵ will often be neglicible.

The *number of epochs* N_t for which one wants to know $\underline{\pi}(t)$ also influences the computational intensivity. Depending on the ACE scheme which can be applied, it is more or less important. In all cases it will ask for some extra instructions to be carried out for computing the jump probabilities. The overhead for computing the jump probabilities for AU will increase at most linear in N_t. So, we will see that t^* decreases when more values of t are of interest.

Conclusion on the numerical complexity of AU and SU

In this section we have described in subsection A how the complexity results for the EMR model in [1] have been derived. These complexity results are based on counting, and have several underlying assumption. The exact values of the complexity of the uniformization schemes heavily depend on the chosen assumptions. An actual software implementation can give additional insights in the complexity of the method, because practical numerical and programming aspects can then be taken into account. However, it is our opinion that for the qualitative evaluation of the difference between AU and SU, the analysis provides sufficient insight.

In subsection B.2 we have identified the problem aspects which influence the numerical complexity of AU and SU. We have seen that the computational benefit of using AU very much depends on specific problem characteristics. On the other hand we have seen that the computational overhead for AU relative to SU converges to a fixed percentage of the computational cost of SU, provided we can apply the modified ACE schemes for AU processes with converged rates. Computational overhead therefore does not have to be a bottleneck if AU is applied. Further research to the usability of AU for otherwise hard to solve problems is therefore of interest.

REFERENCES

[1] A.P.A. van Moorsel, W.H. Sanders, "Adaptive Uniformization", *Memoranda Informatica*, University of Twente, The Netherlands, 1992 (submitted for publication).

[2] N.C. Severo, "A Recursion Theorem on Solving Differential-Difference Equations and Applications to some Stochastic Processes", *J. Appl. Prob.* **6**, pp. 673-681, 1969.

[3] R.A. Marie, A.L. Reibman, K.S. Trivedi, "Transient Analysis of Acyclic Markov Chains", *Perf. Eval.* **7**, pp. 175-194, 1987.

[4] C. Lindemann, M. Malhotra, K.S. Trivedi, "Numerical Methods for Reliability Evaluation of Closed Fault-tolerant Systems", Technical Report, Duke University, 1992.

[5] K.S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, 1982.

[6] B.L. Fox, P.W. Glynn, "Computing Poisson Probabilities", *Comm. ACM* **31**, pp. 440-445, 1988.

[7] J.K. Muppala, K.S. Trivedi, "Numerical transient solution of finite Markovian queueing systems", to appear in: *Queueing and Related Models*, U.Bhat (Ed.), Oxford University Press, 1992.

[8] A. Reibman, "A Splitting Technique for Markov Chain Transient Solution", *Numerical Solution of Markov Chains*, W.J. Stewart (Ed.), Marcel Dekker, New York, 1991.

[9] B.S. Yoon, J.G. Shanthikumar, "Bounds and Approximations for the Transient Behavior of Continuous-Time Markov Chains", *Prob. Eng. Inf. Sc.* **3**, pp. 175-198, 1989.

[10] A. Reibman, K.S. Trivedi, "Numerical Transient Analysis of Markov Models", *Comput. Opns Res.* **15**, pp. 19-36, 1988.

[11] J. Dunkel, H. Stahl, "On the Transient Analysis of Stiff Markov Chains", *Proc. Third Working Conference on Dependable Computing for Critical Applications*, Mondello, Italy, Sept. 1992.