A New General-purpose Method for the Computation of the Interval Availability Distribution

Juan A. Carrasco
Departament d'Enginyeria Electrònica, Universitat Politècnica de Catalunya,
Diagonal 647, plta. 9, 08028 Barcelona, Spain

Except for formatting details, this version matches exactly the version published with the same title and author in *INFORMS Journal on Computing*, vol. 25, no. 4, Fall 2013, pp. 774–791

Abstract

We develop a new randomization-based general-purpose method for the computation of the interval availability distribution of systems modeled by continuous-time Markov chains (CTMCs). The basic idea of the new method is the use of a randomization construct with different randomization rates for up and down states. The new method is numerically stable and computes the measure with well-controlled truncation error. In addition, for large CTMC models, when the maximum output rates from up and down states are significantly different, and when the interval availability has to be guaranteed to have a level close to one, the new method is significantly or moderately less costly in terms of CPU time than a previous randomization-based state-of-the-art method, depending on whether the maximum output rate from down states is larger than the maximum output rate from up states or vice versa. Otherwise, the new method can be more costly, but a relatively inexpensive for large models switch of reasonable quality can be easily developed to choose the fastest method. Along the way, we show the correctness of a generalized randomization construct, in which arbitrarily different randomization rates can be associated with different states, for both finite CTMCs with infinitesimal generator and uniformizable CTMCs with denumerable state space.

Keywords: Engineering, probability, Markov processes, availability.

1 Introduction

The interval availability is defined as the fraction of time in a time interval in which a system is operational. There has been much interest in computing the distribution of the interval availability. Most of the work has dealt with the case in which the behavior of the system is captured by an (homogeneous) continuous-time Markov chain (CTMC) model having up (operational) and down states. Computing the distribution of the interval availability of systems modeled by a CTMC has been proved to be a challenging problem (see Carrasco 2004a, 2011; Goyal and Tantawi 1988; Ross 1983; Rubino and Sericola 1992, 1993, 1995; Sericola 1990; de Souza e Silva and Gail 1986; Takács 1957). The first effort is reported in Takács (1957), where a closed-form integral expression was obtained for a two-state CTMC model. In Ross (1983), randomization was used to obtain the distribution of the operational time in a time interval of the same two-state CTMC model. The first method able to deal with arbitrary finite CTMC models was developed by de Souza e Silva and Gail (1986) using randomization. Goyal and Tantawi (1988) developed a numerical approximate method without error bounds. Sericola (1990) obtained a closed-form solution in terms of growing size matrices. Rubino and Sericola (1992) developed an efficient numerical method for the particular case in which operational and down periods are independent, operational periods are identically distributed except, perhaps, the first one, and down periods are identically distributed. Rubino and Sericola (1993) also developed two randomization-based methods which reduce the computational requirements of the randomization-based method developed by de Souza e Silva and Gail (1986). The first of such methods is guaranteed to reduce the CPU time requirements; the second one is guaranteed to reduce the memory requirements and often also reduces the CPU time requirements. That second method was reviewed in Rubino and Sericola (1995) as Algorithm A, where it was taken as starting point to develop another method (Algorithm B) that is competitive when the number of operational states of the model is small and, furthermore, can deal with some class of CTMC models with denumerable state space. Recently, we have developed a method in Carrasco (2004a), which will be called here *regenerative transformation*, targeted at a class of CTMC models, class C_1, including both exact and bounding failure/repair CTMC models of fault-tolerant systems with increasing structure function (Barlow and Proschan 1981), exponential failure and repair time distributions, and repair in every state with failed components, with failure rates much smaller than repair rates, which can be significantly less costly in terms of CPU time than Algorithm A if the interval availability has to be guaranteed to have a level close to one. The regenerative transformation method was taken as starting point in Carrasco (2011) to develop a method, *bounding regenerative transformation*, to compute bounds for the interval availability distribution. For a class of CTMC models slightly less general than class C_1, the version that seems to be computationally less costly in terms of CPU time seems to be computationally little costly relative to the model size when that model size is large, provided the interval availability has to be guaranteed to have a level close to one. Furthermore, under additional conditions that are satisfied by both exact and bounding failure/repair CTMC models of fault-tolerant systems with increasing structure function, exponential failure and repair time distributions and repair in every state with failed components, with failure rates much smaller than repair rates, the bounds seem to be tight for any time interval for some initial

probability distributions, and for time intervals not too small in case of other distributions.

The interval availability distribution can be looked at as a particular case of the distribution of the reward earned in a time interval by a Markov reward process with reward rates associated with states, and several methods have been proposed to compute that distribution (Donatiello and Grassi 1991; Islam and Ammar 1989; Nabli and Sericola 1996; Pattipati et al. 1993; Qureshi and Sanders 1996; Smith et al. 1988; de Souza e Silva and Gail 1989; Suñé et al. 2010) or bounds for it (Carrasco 2006; Rácz et al. 2002).

Because of its numerical stability, well-controlled truncation error, and moderate memory and CPU time requirements, Algorithm A can be considered the current state-of-the-art general-purpose method for computing the interval availability distribution for finite CTMCs. In this paper, we develop a new method for computing the interval availability distribution for systems modeled with finite CTMCs. The basic idea of the new method is the use of a generalized randomization construct with different randomization rates for the up and down states. Like Algorithm A, the new method is numerically stable and computes the interval availability distribution with well-controlled truncation error. In addition, for large CTMC models, when the maximum output rates from the up and down states are significantly different, and when the interval availability has to be guaranteed to have a level close to one, the new method is significantly or moderately less costly in terms of CPU time, depending on whether the maximum output rate from down states is larger than the maximum output rate from up states, or vice versa. Otherwise, the new method can be more costly, but a relatively inexpensive switch for large models of reasonable quality in that case can be easily developed to choose the fastest method.

The rest of the paper is organized as follows. Section 2 defines formally the measure that will be computed by the new method, reviews the randomization construct on which Algorithm A and most previously proposed methods for computing the interval availability distribution are based, and reviews Algorithm A. Section 3 develops the new method and argues that it can be significantly less costly in terms of CPU time than Algorithm A. It also defines the switch between the new method and Algorithm A. Most of the effort is developed to the derivation of computationally inexpensive and good truncation points. Section 4 begins by analyzing the numerical stability of the new method using a CTMC model with closed-form solution. Then, we use two large CTMC models to illustrate that the new method can be significantly less costly in terms of CPU time than Algorithm A, confirming the analysis performed in §3. Using those two large CTMC models we also asses the quality of the switch. Section 5 presents the conclusions.

2 Preliminaries

Let $X = \{X(t); t \geq 0\}$ be a CTMC with finite state space $\Omega = U \cup D$, where U is the subset of up states and D is the subset of down states. The interval availability complementary distribution at

time t is defined as

$$\mathrm{IAVCD}(t,p) = P\left[\frac{1}{t}\int_0^t \mathbf{1}_{X(\tau)\in U}\, d\tau > p\right],$$

where $\mathbf{1}_c$ denotes the indicator function returning value 1 if condition c is satisfied and value 0 otherwise. In other words, $\mathrm{IAVCD}(t,p)$ is the probability that the fraction of time that the system is up in the time interval $[0,t]$ is greater than p. To simplify the presentation, we will assume throughout the paper $t > 0$ and $0 < p < 1$.

Algorithm A is based on the randomization construct. Let $\mathbf{A} = (a_{i,j})_{i,j\in\Omega}$ denote the infinitesimal generator of X, where $a_{i,i} = -\lambda_i$, λ_i denoting the output rate of X from state i, and $a_{i,j} = \lambda_{i,j}$, $j \neq i$, $\lambda_{i,j}$ denoting the transition rate of X from state i to state j, let $\boldsymbol{\alpha} = (\alpha_i)_{i\in\Omega}$ denote the initial probability distribution column vector of X, where $\alpha_i = P[X(0) = i]$, $i \in \Omega$; and assume $\max_{i\in\Omega} -|a_{i,i}| > 0$. Consider any $\Lambda \geq \max_{i\in\Omega} -a_{i,i}$ and define the (homogeneous) discrete-time Markov chain (DTMC) $\widehat{X} = \{\widehat{X}_n; n = 0,1,2,\ldots\}$ with same state space and initial probability distribution as X and transition matrix $\mathbf{P} = (P_{i,j})_{i,j\in\Omega} = \mathbf{I} + \mathbf{A}/\Lambda$, $\mathbf{I}$ denoting the identity matrix. Let $Q = \{Q(t); t \geq 0\}$ be a Poisson process with arrival rate Λ independent of $\widehat{X}$. In the randomization construct, X is interpreted as the DTMC $\widehat{X}$ subordinated to the Poisson process Q, in the sense that $X(t)$ is the state in which $\widehat{X}$ is at the step given by the number of occurrences in the time interval $[0,t]$ of the Poisson process Q. In fact, we have that X and $\{\widehat{X}_{Q(t)}; t \geq 0\}$ are probabilistically identical (Kijima 1997, Theorem 4.19), and anything depending solely on the probabilistic path behavior of X can be computed using $\{\widehat{X}_{Q(t)}; t \geq 0\}$ instead. The DTMC $\widehat{X}$ is said to be randomized with rate Λ when building $\{\widehat{X}_{Q(t)}; t \geq 0\}$. Beacuse the output rate is uniformized in the randomization construct, the construct is also known as uniformization.

The randomization construct underpins the so-called randomization (uniformization) methods for the computation of transient probability distribution vectors of CTMCs, expected values of functions of the state of CTMCs at a given time, expected time averages, and variances of time averages (Grassmann 1977a, b, 1987; Gross and Miller 1984; Reibman and Trivedi 1989; Suñé and Carrasco 2005). A major advantage of most randomization methods is their numerical stability. Here, a method is considered to be numerically stable when the relative error in the computed solution, and in each component of the computed solution if the computed solution is a vector, resulting from round-off errors can be expected to be small. This results from the fact that, apart from the computation of the diagonal elements of $\mathbf{P}$ and some methods for computing Poisson probabilities, they only involve additions of positive quantities. The issue has been rigorously examined in Grassmann (1993) in relation to the computation of the transient probability vector of a finite CTMC. Computing Poisson probabilities avoiding intermediate underflows and overflows is not a trivial problem, and several methods have been proposed for computing them (Fox and Glynn 1988, Knüsel 1986, van Moorsel and Sanders 1997). In the new method and in our implementation of Algorithm A we will use the method described in Knüsel (1986, pp. 1028–29), which is reasonably efficient and numerically stable. Several variants of randomization methods have also been proposed, including selective and compressed selective randomization (Melamed and Yadin 1984), uniformization with stationarity detection (Reibman and Trivedi 1988, Sericola 1999), adaptive uniformization (van Moorsel and Sanders 1994), adaptive uniformization/standard uniformization (van

Moorsel and Sanders 1997), regenerative randomization (Carrasco 2003, 2005), and randomization with quasistationarity detection (Carrasco 2004b). Adaptive uniformization is somehow related to the new method, the difference being that in adaptive uniformization the randomization rate depends on the subset of states in which the randomized DTMC can be after a particular number of steps, whereas in the new method it depends on whether the state visited by the randomized DTMC is up or down.

Algorithm A is based on the formulation for $\mathrm{IAVCD}(t,p)$ which results from the randomization construct

$$\mathrm{IAVCD}(t,p) = \sum_{n=0}^{\infty} \sum_{k=0}^{n} \frac{(\Lambda t)^n}{n!}\, e^{-\Lambda t} \binom{n}{k} p^k (1-p)^{n-k} Y_{n,k}\,, \tag{1}$$

where $Y_{n,k} = P[\#(\widehat{X}_{0:n} \in U) > k]$ and $\#(\widehat{X}_{0:n} \in B)$ denotes the number of indices k, $0 \le k \le n$, for which $\widehat{X}_k \in B$ holds. In the method, three truncations are performed to the summations of (1). With ε being an error control parameter, the three truncations are defined by the parameters

$$N = \min\left\{ n \ge 0 \;:\; \sum_{k=n+1}^{\infty} \frac{(\Lambda t)^k}{k!}\, e^{-\Lambda t} \le \frac{\varepsilon}{2} \right\},$$

$$C'' = \begin{cases} \max\left\{ c \;:\; 0 \le c \le N \;\wedge\; \sum_{k=0}^{c} \frac{((1-p)\Lambda t)^k}{k!}\, e^{-(1-p)\Lambda t} \le \frac{\varepsilon}{4} \right\} & \text{if } e^{-(1-p)\Lambda t} \le \frac{\varepsilon}{4} \\ -1 \quad \text{if } e^{-(1-p)\Lambda t} > \frac{\varepsilon}{4} \end{cases},$$

$$C' = \begin{cases} \min\left\{ N, \min\left\{ c \ge 0 \;:\; \sum_{k=c+1}^{\infty} \frac{((1-p)\Lambda t)^k}{k!}\, e^{-(1-p)\Lambda t} \le \frac{\varepsilon}{4} \right\} \right\} & \text{if } C'' \ne -1 \\ \min\left\{ c \ge 0 \;:\; \sum_{k=c+1}^{\infty} \frac{((1-p)\Lambda t)^k}{k!}\, e^{-(1-p)\Lambda t} \le \frac{\varepsilon}{2} \right\} & \text{if } C'' = -1 \end{cases}.$$

This gives the approximate value for $\mathrm{IAVCD}(t,p)$

$$\mathrm{IAVCD}^a_{N,C',C''}(t,p) = \sum_{n=0}^{N} \sum_{k=\max\{0,n-C'\}}^{\min\{n,N-C''-1\}} \frac{(\Lambda t)^n}{n!}\, e^{-\Lambda t} \binom{n}{k} p^k (1-p)^{n-k} Y_{n,k}\,,$$

which satisfies

$$\mathrm{IAVCD}(t,p) = \mathrm{IAVCD}^a_{N,C',C''}(t,p) + e_{N,C',C''}(t,p)\,,$$

with $0 \le e_{N,C',C''}(t,p) \le \varepsilon$.

Let $\widehat{X}^i$ denote a version of $\widehat{X}$ with initial state $i \in \Omega$, let $Y^i_{n,k} = P[\#(\widehat{X}^i_{0:n} \in U) > k]$ and let $\mathbf{Y}_{n,k}$ be the column vector $(Y^i_{n,k})_{i\in\Omega}$. Clearly, $Y_{n,k} = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{Y}_{n,k}$, with $^{\mathrm{T}}$ denoting the transpose operator. Let $\mathbf{0}$ and $\mathbf{1}$ denote column vectors of appropriate dimensions with all elements equal to, respectively, 0 and 1; let $\mathbf{Y}^U_{n,k}$ and $\mathbf{Y}^D_{n,k}$ denote the subvectors of $\mathbf{Y}_{n,k}$ including the components associated with, respectively, the up and down states; and let $\mathbf{P}_{B,C}$ denote the submatrix of $\mathbf{P}$ collecting the components with index pairs in $B \times C$. Then, the vectors $\mathbf{Y}_{n,k}$ in the domain of (n,k) pairs for which $Y_{n,k}$ have to be obtained to compute $\mathrm{IAVCD}^a_{N,C',C''}(t,p)$ can be obtained for increasing n and, for each n, for increasing k using the recurrences $\mathbf{Y}^U_{n,k} = \mathbf{P}_{U,\Omega} \mathbf{Y}_{n-1,k-1}$, $n > 0$,

$0 < k \leq n$, $\mathbf{Y}^U_{n,0} = \mathbf{1}$, $n > 0$, $\mathbf{Y}^D_{n,k} = \mathbf{P}_{D,\Omega}\mathbf{Y}_{n-1,k}$, $n > 0$, $0 \leq k < n$, $\mathbf{Y}^D_{n,n} = \mathbf{0}$, $n > 0$, with initial conditions $\mathbf{Y}^U_{0,0} = \mathbf{1}$, $\mathbf{Y}^D_{0,0} = \mathbf{0}$.

As Λ increases, the truncation points N, C' and C'' increase and the computational cost of the method tends to increase, making $\Lambda = \max_{i\in\Omega} -a_{i,i}$ a reasonable best selection for Λ. We will assume that selection. Also, $P_k(\lambda)$ will denote the probability that a Poisson random variable with parameter λ has value k. Of course $P_0(0) = 1$.

3 The New Method

To simplify, we will exclude the cases in which $U = \emptyset$, $D = \emptyset$, $\max_{i\in U} -a_{i,i} = 0$, or $\max_{i\in D} -a_{i,i} = 0$. These are not severe restrictions, since, for $U = \emptyset$, $\mathrm{IAVCD}(t,p) = 0$, for $D = \emptyset$, $\mathrm{IAVCD}(t,p) = 1$, for $U \neq \emptyset$ and $\max_{i\in U} -a_{i,i} = 0$, all up states are absorbing and $\mathrm{IAVCD}(t,p) = P[X((1-p)t) \in U]$, which is simpler to compute, and, for $D \neq \emptyset$ and $\max_{i\in D} -a_{i,i} = 0$, all down states are absorbing and $\mathrm{IAVCD}(t,p) = P[X(pt) \in U]$, which is simpler to compute.

We will start by obtaining a new closed-form formulation for $\mathrm{IAVCD}(t,p)$. After that, we will deal with the issues necessary to derive the method from that closed-form formulation.

3.1 Formulation

To obtain the new closed-form formulation for $\mathrm{IAVCD}(t,p)$, we will consider a randomization construct in which a DTMC $\widehat{X}$ is randomized with rate $\Lambda_U = \max_{i\in U} -a_{i,i}$ in the states in U and rate $\Lambda_D = \max_{i\in D} -a_{i,i}$ in the states in D. The DTMC $\widehat{X}$ has same state space and initial probability distribution as X and transition matrix $\mathbf{P} = \mathbf{I} + \mathbf{\Lambda}^{-1}_{UD}\mathbf{A}$, where $\mathbf{\Lambda}_{UD} = \mathbf{diag}[\mathbf{1}_{i\in U}\Lambda_U + \mathbf{1}_{i\in D}\Lambda_D]_{i\in\Omega}$, $\mathbf{diag}[d_i]_{i\in\Omega}$ denoting a $|\Omega| \times |\Omega|$ diagonal matrix with diagonal elements d_i, $i \in \Omega$. Let $Y = \{Y(t); t \geq 0\}$ be the stochastic process resulting from the construct. That construct was considered in Carrasco (2004a) and can be looked at as a particular case of a more general construct in which arbitrarily different randomization rates $\Lambda_i \geq -a_{i,i}$, $\Lambda_i > 0$, $i \in \Omega$ are associated with the states of a DTMC $\widehat{X}$. In that generalized construct, $\widehat{X}$ has, of course, same state space and initial probability distribution as X and transition matrix $\mathbf{P} = \mathbf{I} + \mathbf{\Lambda}^{-1}\mathbf{A}$, $\mathbf{\Lambda} = \mathbf{diag}[\Lambda_i]_{i\in\Omega}$. Let $\{E_{i,n}, i \in \Omega, n \geq 1\}$ be a collection of exponentially distributed random variables that are mutually independent and independent of $\widehat{X}$, with $E_{i,n}$ having parameter Λ_i. Then, in that generalized construct, Y can be formally defined, respecting the fact that visits durations of $\widehat{X}$ to i are exponentially distributed with parameter Λ_i, as $Y = \{\widehat{X}_{\tau(t)}; t \geq 0\}$ with $\tau(t) = \min\{n \geq 0 : \sum_{k=0}^{n} E_{\widehat{X}_k,k+1} > t\}$. The following theorem asserts that Y is probabilistically identical to X, allowing the use of Y instead of X when computing anything depending solely on the probabilistic path behavior of X.

Theorem 1. *Let $X = \{X(t); t \geq 0\}$ be either a finite CTMC with state space Ω or a uniformizable CTMC with denumerable state space Ω, and let $\mathbf{A} = (a_{i,j})_{i,j\in\Omega}$ be the infinitesimal generator of X.*

Let Λ_i, $i \in \Omega$ be such that $\Lambda_i \geq -a_{i,i}$, $\Lambda_i > 0$ and the Λ_i, $i \in \Omega$ are uniformly bounded from above. Let $\widehat{X}$ be the DTMC with same state space and initial probability distribution as X and transition matrix $\mathbf{P} = \mathbf{I} + \mathbf{\Lambda}^{-1}\mathbf{A}$, $\mathbf{\Lambda} = \mathbf{diag}[\Lambda_i]_{i\in\Omega}$. Then, $Y = \{\widehat{X}_{\tau(t)}; t \geq 0\}$ with $\tau(t) = \min\{n \geq 0 : \sum_{k=1}^{n+1} E_{\widehat{X}_k,k+1} > t\}$ is probabilistically identical to X.

Proof. By the monotone convergence theorem,

$$E\left[\lim_{n\to\infty}\sum_{k=0}^{n} E_{\widehat{X}_{k,k+1}}\right] = \sum_{k=0}^{\infty}\frac{1}{\Lambda_{\widehat{X}_k}} = \infty\,,$$

$\lim_{n\to\infty}\sum_{k=0}^{n} E_{\widehat{X}_{k,k+1}}$ will be ∞ with probability one (see, for instance, Kijima 1997, p. 187, for the argument), for any arbitrary $t \geq 0$, $\tau(t)$ will be defined and finite with probability one, implying that $\tau(s)$, $0 \leq s \leq t$ will be defined and finite with probability one, and, because t s arbitrary, that $\tau(t)$, $t \geq 0$ will be defined and finite with probability one. This shows that $Y = \{\widehat{X}_{\tau(t)}; t \geq 0\}$ is defined with probability one. Because Y is defined with probability one and the stochastic behavior of either a finite CTMC with infinitesimal generator or a uniformizable CTMC with denumerable state space is defined by its initial probability distribution together with its infinitesimal generator (Kijima 1997), it is enough to prove that the initial probability distribution of Y coincides with that of X and that Y is a CTMC with infinitesimal generator $\mathbf{A}$. The first is rather obvious, since $\tau(0) = 0$ with probability one, and, then, the initial probability distribution of Y coincides with that of $\widehat{X}$, and X. For the second, it suffices to show that, for arbitraries $t \geq 0$ and $i \in \Omega$ with $P[\widehat{X}_{\tau(t)} = i] > 0$,

$$\lim_{h\to 0^+}\frac{P\big[\widehat{X}_{\tau(t+h)} = j \mid \widehat{X}_{\tau(t)} = i\big]}{h} = a_{i,j},\ \ j \in \Omega, j \neq i\,,$$

$$\lim_{h\to 0^+}\frac{P\big[\widehat{X}_{\tau(t+h)} = i \mid \widehat{X}_{\tau(t)} = i\big] - 1}{h} = a_{i,i}\,.$$

Both follow if, for arbitraries $t \geq 0$, $i \in \Omega$ and $k \geq 0$ with $P[\widehat{X}_k = i \wedge \tau(t) = k] > 0$,

$$\lim_{h\to 0^+}\frac{P\big[\widehat{X}_{\tau(t+h)} = j \mid \widehat{X}_k = i \wedge \tau(t) = k\big]}{h} = a_{i,j},\ \ j \in \Omega, j \neq i\,,$$

$$\lim_{h\to 0^+}\frac{P\big[\widehat{X}_{\tau(t+h)} = i \mid \widehat{X}_k = i \wedge \tau(t) = k\big] - 1}{h} = a_{i,i}\,.$$

But, using the memoryless property of exponential random variables and the fact that $\widehat{X}$ and the collection of random variables $\{E_{i,n}, i \in \Omega, n \geq 1\}$ are independent, we have

$$\begin{aligned}
&P\big[\widehat{X}_{\tau(t+h)} = j \mid \widehat{X}_k = i \wedge \tau(t) = k\big] \\
&\quad = P\big[E_{i,k+1} \leq h\big]P\big[\widehat{X}_{k+1} = j \mid \widehat{X}_k = i\big] + o(h) \\
&\quad = P_{i,j}\Lambda_i h + o(h) = a_{i,j}h + o(h)\,,\ \ j \in \Omega, j \neq i\,,
\end{aligned}$$

$$\begin{aligned}
&P\big[\widehat{X}_{\tau(t+h)} = i \mid \widehat{X}_k = i \wedge \tau(t) = k\big] - 1 \\
&\quad = -P\big[\widehat{X}_{\tau(t+h)} \neq i \mid \widehat{X}_k = i \wedge \tau(t) = k\big] \\
&\quad = -P\big[E_{i,k+1} \leq h\big]P\big[\widehat{X}_{k+1} \neq i \mid \widehat{X}_k = i\big] + o(h) \\
&\quad = -(1 - P_{i,i})\Lambda_i h + o(h) = a_{i,i}h + o(h)\,. \quad \square
\end{aligned}$$

With X and Y being probabilistically identical, we can base the computation of $\mathrm{IAVCD}(t,p)$ on the analysis of Y. Because the exponential visit durations in up and down states of the randomized DTMC $\widehat{X}$ are, in general, different, the use of two randomization rates will force us to count the number of visits of $\widehat{X}$ to up states during a given number of steps of $\widehat{X}$. This is also necessary in Algorithm A, so essentially no computational burden is added by the consideration of two randomization rates in the new method. Informally, using two randomization rates can be advantageous when Λ_U and Λ_D are significantly different. This is because, in Algorithm A, a randomization rate equal to $\max\{\Lambda_U, \Lambda_D\}$ is used, and the number of steps of $\widehat{X}$ that have to be considered to capture well enough the behavior of X in the time interval $[0,t]$ will tend to be significantly larger than in the method with two randomization rates. This is the intuition that has motivated the new method. On the other hand, the use of two randomization rates will necessarily complicate both the formulation of the measure and its truncation.

We will obtain a closed-form expression for $\mathrm{IAVCD}(t,p)$ in terms of probabilities of sets of realizations of $Y = \{\widehat{X}_{\tau(t)}; t \geq 0\}$ with $\tau(t) = \min\{n \geq 0 : \sum_{k=0}^{n}(\mathbf{1}_{\widehat{X}_k \in U} E_{k+1}^U + \mathbf{1}_{\widehat{X}_k \in D} E_{k+1}^D) > t\}$, E_k^U, $k = 1,2,3,\ldots$ denoting exponential random variables with parameter Λ_U and E_k^D, $k = 1,2,3,\ldots$ denoting exponential random variables with parameter Λ_D, with all random variables independent among them and independent of $\widehat{X}$. Then, conditioning on the number of steps given by $\widehat{X}$ at time t, on the number of up states visited by $\widehat{X}$ on those steps, and on whether the last visited state was up or down, using the theorem of total probability and the fact that the random variables E_k^U and E_k^D are independent of $\widehat{X}$, we have

$$
\begin{aligned}
&\mathrm{IAVCD}(t,p) \\
&= \sum_{n=0}^{\infty}\sum_{k=1}^{n+1} P\left[\#(\widehat{X}_{0:n} \in U) = k \wedge \widehat{X}_n \in U \wedge \sum_{i=1}^{k-1} E_i^U + \sum_{i=1}^{n+1-k} E_i^D \leq t \right. \\
&\qquad\qquad \left. \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\qquad \times P\left[t - \sum_{i=1}^{n+1-k} E_i^D > pt \;\middle|\; \#(\widehat{X}_{0:n} \in U) = k \wedge \widehat{X}_n \in U \right. \\
&\qquad\qquad \left. \wedge \sum_{i=1}^{k-1} E_i^U + \sum_{i=1}^{n+1-k} E_i^D \leq t \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\quad + \sum_{n=0}^{\infty}\sum_{k=0}^{n} P\left[\#(\widehat{X}_{0:n} \in U) = k \wedge \widehat{X}_n \in D \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n-k} E_i^D \leq t \right. \\
&\qquad\qquad \left. \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\qquad \times P\left[\sum_{i=1}^{k} E_i^U > pt \;\middle|\; \#(\widehat{X}_{0:n} \in U) = k \wedge \widehat{X}_n \in D \right. \\
&\qquad\qquad \left. \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n-k} E_i^D \leq t \wedge \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{n=0}^{\infty}\sum_{k=1}^{n+1} P[\#(\widehat{X}_{0:n}\in U)=k \ \wedge\ \widehat{X}_n\in U] \\
&\qquad \times P\left[\sum_{i=1}^{k-1} E_i^U + \sum_{i=1}^{n+1-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\qquad \times P\left[t - \sum_{i=1}^{n+1-k} E_i^D > pt \;\Big|\; \sum_{i=1}^{k-1} E_i^U + \sum_{i=1}^{n+1-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\quad + \sum_{n=0}^{\infty}\sum_{k=0}^{n} P[\#(\widehat{X}_{0:n}\in U)=k \ \wedge\ \widehat{X}_n\in D] \\
&\qquad \times P\left[\sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&\qquad \times P\left[\sum_{i=1}^{k} E_i^U > pt \;\Big|\; \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t\right] \\
&= \sum_{n=0}^{\infty}\sum_{k=1}^{n+1} \Omega_{n,k}^U F_{n,k}^U(t,p) + \sum_{n=0}^{\infty}\sum_{k=0}^{n} \Omega_{n,k}^D F_{n,k}^D(t,p)\,, \qquad (2)
\end{aligned}
$$

with

$$\Omega_{n,k}^U = P[\#(\widehat{X}_{0:n}\in U)=k \ \wedge\ \widehat{X}_n\in U]\,, \qquad (3)$$

$$\Omega_{n,k}^D = P[\#(\widehat{X}_{0:n}\in U)=k \ \wedge\ \widehat{X}_n\in D]\,, \qquad (4)$$

$$F_{n,k}^U(t,p) = P\left[\sum_{i=1}^{k-1} E_i^U + \sum_{i=1}^{n+1-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t \ \wedge\ t - \sum_{i=1}^{n+1-k} E_i^D > pt\right],$$

and

$$F_{n,k}^D(t,p) = P\left[\sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n-k} E_i^D \le t \ \wedge\ \sum_{i=1}^{k} E_i^U + \sum_{i=1}^{n+1-k} E_i^D > t \ \wedge\ \sum_{i=1}^{k} E_i^U > pt\right].$$

The following theorem gives integral expressions for $F_{n,k}^U(t,p)$, $n\ge 0$, $1\le k\le n+1$ and $F_{n,k}^D(t,p)$, $n\ge 0$, $0\le k\le n$.

Theorem 2. *For $n\ge 0$,*

$$F_{n,n+1}^U(t,p) = \frac{(\Lambda_U t)^n}{n!}\, e^{-\Lambda_U t}\,.$$

For $n\ge 0$, $F_{n,0}^D(t,p)=0$. For $n\ge 1$ and $1\le k\le n$, $F_{n,k}^U(t,p)=\Lambda_D I_{n,k}(t,p)$, $F_{n,k}^D(t,p)=\Lambda_U I_{n,k}(t,p)$, where

$$I_{n,k}(t,p) = \int_0^{(1-p)t} \frac{(\Lambda_U(t-x))^{k-1}}{(k-1)!}\, e^{-\Lambda_U(t-x)}\, \frac{(\Lambda_D x)^{n-k}}{(n-k)!}\, e^{-\Lambda_D x}\, dx\,.$$

Proof. See the online supplement (available at `http://dx.doi.org/10.1287/ijoc.1120.0539`). □

Three issues have to be solved to complete the method. The first one is the truncation of the infinite summations in (2) with control of the truncation error. The second one is the computation of the quantities $\Omega^U_{n,k}$ and $\Omega^D_{n,k}$. The third one is the efficient evaluation with numerical stability of the integrals $I_{n,k}(t,p)$. We will deal with those issues in that order. After that, a comparison of the computational costs of the new method and Algorithm A will be made.

3.2 Truncation of the infinite summations

Let ε be an arbitrarily small error control parameter. We will perform two truncations to the summations in (2). Each truncation will introduce a nonnegative error which will be bounded from above by $\varepsilon/4$, yielding a formulation for $\mathrm{IAVCD}(t,p)$ with nonnegative truncation error $\leq \varepsilon/2$. The first truncation deletes the terms in (2) corresponding to values of $n > N'$, where N' is a truncation parameter ≥ 0. This gives, taking into account $F^D_{n,0}(t,p) = 0$, $n \geq 0$ (Theorem 2),

$$\begin{aligned}
\mathrm{IAVCD}(t,p) &= \mathrm{IAVCD}'_{N'}(t,p) + e'_{N'}(t,p)\,, \\
\mathrm{IAVCD}'_{N'}(t,p) &= \sum_{n=0}^{N'} \sum_{k=1}^{n+1} \Omega^U_{n,k} F^U_{n,k}(t,p) + \sum_{n=1}^{N'} \sum_{k=1}^{n} \Omega^D_{n,k} F^D_{n,k}(t,p)\,, \qquad (5) \\
0 \leq e'_{N'}(t,p) &= \sum_{n=N'+1}^{\infty} \sum_{k=1}^{n+1} \Omega^U_{n,k} F^U_{n,k}(t,p) + \sum_{n=N'+1}^{\infty} \sum_{k=1}^{n} \Omega^D_{n,k} F^D_{n,k}(t,p)\,.
\end{aligned}$$

Using $\Omega^U_{n,k} \geq 0$, $n \geq 1$, $1 \leq k \leq n+1$, $\Omega^D_{n,k} \geq 0$, $n \geq 1$, $1 \leq k \leq n$, $\sum_{k=1}^{n+1} \Omega^U_{n,k} + \sum_{k=1}^{n} \Omega^D_{n,k} \leq 1$, $n \geq 1$, and Theorem 2, and letting $\Lambda = \max\{\Lambda_U, \Lambda_D\}$,

$$\begin{aligned}
e'_{N'}(t,p) &\leq \sum_{n=N'+1}^{\infty} \max\left\{ \max_{1\leq k\leq n+1} F^U_{n,k}(t,p), \max_{1\leq k\leq n} F^D_{n,k}(t,p) \right\} \\
&= \sum_{n=N'+1}^{\infty} \max\left\{ \frac{(\Lambda_U t)^n}{n!}\, e^{-\Lambda_U t}, \max_{1\leq k\leq n} \Lambda I_{n,k}(t,p) \right\}.
\end{aligned}$$

Direct use of the previous upper bound for $e'_{N'}(t,p)$ to determine the truncation point N' is impractical due to the need to determine $\max_{1\leq k\leq n} \Lambda I_{n,k}(t,p)$. The following theorem gives an inexpensive upper bound for that maximum.

Theorem 3. *Let $n \geq 1$. Then,*

$$\max_{1\leq k\leq n} \Lambda I_{n,k}(t,p) \leq U_n(\Lambda_U, \Lambda_D, t, p)\,,$$

where, for $n \leq (\Lambda_U + (1-p)\Lambda_D)/\Lambda_U$,

$$U_n(\Lambda_U, \Lambda_D, t, p) = (1-p)\Lambda t\, \frac{((1-p)\Lambda_D t)^{n-1}}{(n-1)!}\, e^{-p\Lambda_U t}$$

and, otherwise,

$$U_n(\Lambda_U, \Lambda_D, t, p) = (1-p)\Lambda t\, \frac{(\Lambda_U t)^{k^*-1}}{(k^*-1)!}\, \frac{((1-p)\Lambda_D t)^{n-k^*}}{(n-k^*)!}\, e^{-p\Lambda_U t}$$

with $k^* = \lfloor (\Lambda_U/(\Lambda_U + (1-p)\Lambda_D))n \rfloor + 1$.

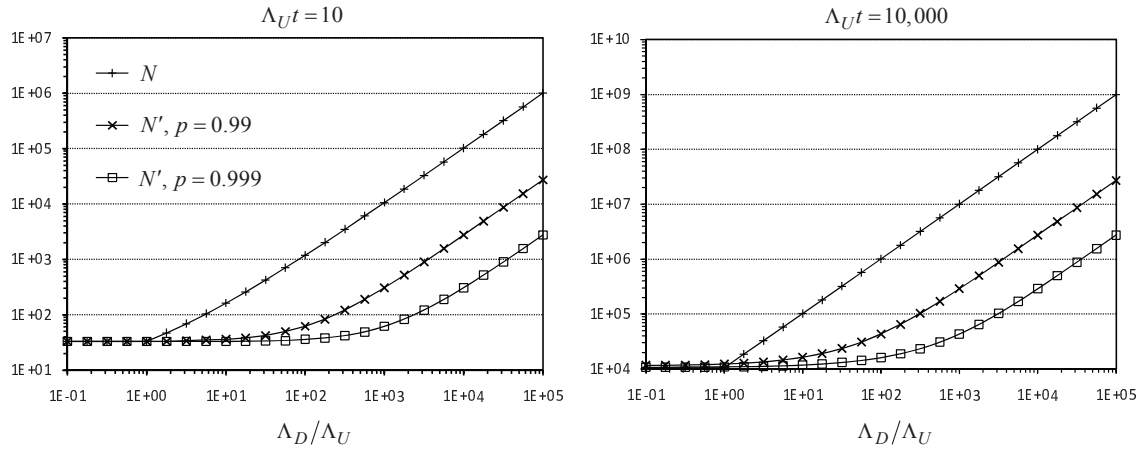

Figure 1: Truncation points N and N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$.

Proof. See the online supplement. □

Using Theorem 3, the truncation point N' can be chosen using

$$N' = \min\left\{ n \geq 0 \ : \ \sum_{m=n+1}^{\infty} \max\left\{ \frac{(\Lambda_U t)^m}{m!}\, e^{-\Lambda_U t}, U_m(\Lambda_U, \Lambda_D, t, p) \right\} \leq \frac{\varepsilon}{4} \right\}.$$

Direct computation of $U_m(\Lambda_U, \Lambda_D, t, p)$ might be problematic due to possible underflows and overflows. The problem can be solved by taking the logarithms of $(1-p)\Lambda t$, $((1-p)\Lambda_D t)^{m-1}$, $(m-1)!$, $e^{-p\Lambda_U t}$, $(\Lambda_U t)^{k^*-1}$, $(k^*-1)!$, $((1-p)\Lambda_D t)^{m-k^*}$, and $(m-k^*)!$, adding/subtracting those logarithms to obtain the logarithm of $U_m(\Lambda_U, \Lambda_D, t, p)$, and applying the exponential function to the result to obtain $U_m(\Lambda_U, \Lambda_D, t, p)$, where $\log k!$ for large k can be computed using a suitable Stirling approximation. We next analyze the quality of that truncation point N' and how it compares with the corresponding truncation point N used by Algorithm A. The truncation point N' is a function of ε, $\Lambda_U t$, Λ_D/Λ_U, and p. Figure 1 plots N and N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$, $\Lambda_U t = 10$, 10,000, and $p = 0.99$, 0.999. For $\Lambda_D > \Lambda_U$, $N' < N$. The truncation parameter N' increases smoothly with Λ_D/Λ_U till the condition $(1-p)\Lambda_D = \Lambda_U$ is satisfied and increases approximately linearly with Λ_D/Λ_U beyond that point. Figure 2 plots N/N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$, $\Lambda_U t = 10$, 10,000, and $p = 0.99$, 0.999. We can note that N/N' increases with Λ_D/Λ_U up to a value which increases as p gets closer to 1.

The behavior of the truncation point N' is not completely satisfactory. This is due to the coarseness of the upper bound for $\max_{1\leq k\leq n} \Lambda I_{n,k}(t,p)$ given by Theorem 3. We will, therefore, strive to obtain a tighter upper bound for that maximum. Consider again the integrals $I_{n,k}(t,p)$ given by Theorem 2. Because, in the integration domain, $pt \leq t - x \leq t$, for $n \geq 1$ and $1 \leq k \leq n$, where for the expression for the definite integral see, for instance, Abramowitz and Stegun (1970, 4.2.55),

$$\begin{aligned} I_{n,k}(t,p) &< \frac{(\Lambda_U t)^{k-1}}{(k-1)!}\, e^{-p\Lambda_U t} \int_0^{(1-p)t} \frac{(\Lambda_D x)^{n-k}}{(n-k)!}\, e^{-\Lambda_D x}\, dx \\ &= \frac{1}{\Lambda_D}\, \frac{(\Lambda_U t)^{k-1}}{(k-1)!}\, e^{-p\Lambda_U t} \left[1 - \sum_{m=0}^{n-k} \frac{((1-p)\Lambda_D t)^m}{m!}\, e^{-(1-p)\Lambda_D t} \right] \end{aligned}$$

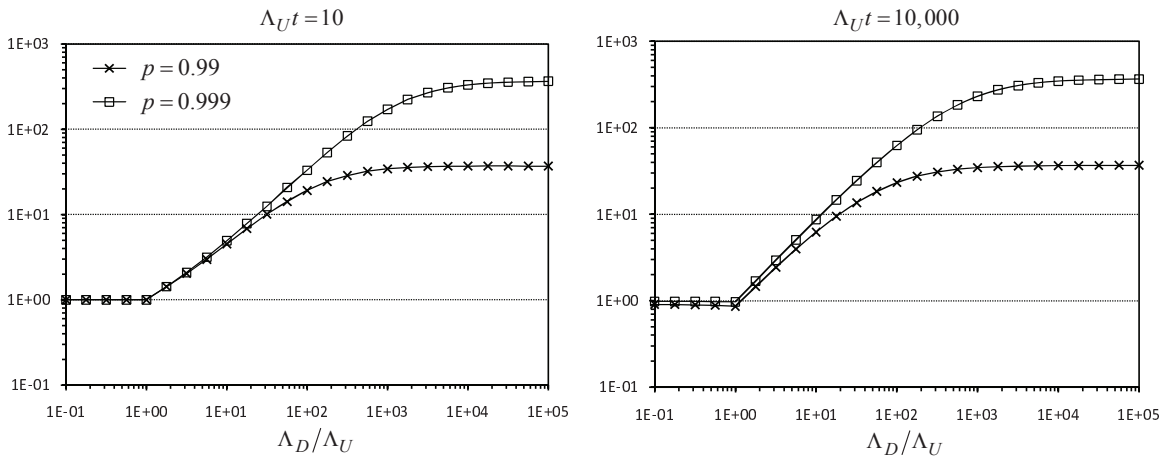

Figure 2: N/N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$.

$$= \frac{1}{\Lambda_D}\,\frac{(\Lambda_U t)^{k-1}}{(k-1)!}\,e^{-p\Lambda_U t}\sum_{m=n+1-k}^{\infty}\frac{((1-p)\Lambda_D t)^m}{m!}\,e^{-(1-p)\Lambda_D t}\,. \tag{6}$$

From the tightening process, it should be clear that the above upper bound for $I_{n,k}(t,p)$ is tight for p close enough to 1.

Noting that $(\Lambda_U t)^{k-1}/(k-1)!$ achieves its maximum at $k = k_0 = \lfloor \Lambda_U t \rfloor + 1$ and decreases toward zero for $k \geq k_0$, let, for $s = 1, 2, 3, \ldots$,

$$\Delta_l(s) = \min\left\{k \geq k_0 \;:\; \frac{\Lambda}{\Lambda_D}\,\frac{(\Lambda_U t)^{k-1}}{(k-1)!}\,e^{-p\Lambda_U t} \leq 10^{-s}\right\}, \tag{7}$$

$$\begin{aligned}\Delta_r(s) \;=\; & \min\{k \geq \lfloor (1-p)\Lambda_D t \rfloor \\ & : \frac{\Lambda}{\Lambda_D}\,\frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!}\,e^{-p\Lambda_U t}\sum_{m=k}^{\infty}\frac{((1-p)\Lambda_D t)^m}{m!}\,e^{-(1-p)\Lambda_D t} \leq 10^{-s}\right\}. \end{aligned}\tag{8}$$

Then, since, given $s \geq 1$,

$$\frac{\Lambda}{\Lambda_D}\,\frac{(\Lambda_U t)^{k-1}}{(k-1)!}\,e^{-p\Lambda_U t}\sum_{m=n+1-k}^{\infty}\frac{((1-p)\Lambda_D t)^m}{m!}\,e^{-(1-p)\Lambda_D t}$$

is smaller than or equal to 10^{-s} for $\Delta_l(s) \leq k \leq n$ and for $1 \leq k \leq n+1-\Delta_r(s)$, the maximum of the above for $1 \leq k \leq n$ is smaller than or equal to 10^{-s} for $n \geq \Delta(s) = \Delta_l(s) + \Delta_r(s) - 1$. Then, noting that $e^{-p\Lambda_U t} = e^{(1-p)\Lambda_U t}e^{-\Lambda_U t}$, we can bound from above $\max_{1\leq k\leq n}\Lambda I_{n,k}(t,p)$ by

$$U'_n(\Lambda_U, \Lambda_D, t, p) = \begin{cases} \dfrac{\Lambda}{\Lambda_D}\,e^{(1-p)\Lambda_U t} & \text{if } n < \Delta(1)\,, \\ 10^{-\max\{m, m\geq 1\,:\, n\geq\Delta(m)\}} & \text{if } n \geq \Delta(1)\,, \end{cases}$$

and have the new truncation point

$$\begin{aligned} N' \;=\; \min\Bigg\{ n \geq \lfloor \Lambda_U t + (1-p)\Lambda_D t \rfloor \;:\; & \sum_{m=n+1}^{\infty}\max\Bigg\{\frac{(\Lambda_U t)^m}{m!}\,e^{-\Lambda_U t}, \\ & \min\big\{U_m(\Lambda_U, \Lambda_D, t, p), U'_m(\Lambda_U, \Lambda_D, t, p)\big\}\Bigg\} \leq \frac{\varepsilon}{4}\Bigg\}. \end{aligned}\tag{9}$$

Direct computation of the left-hand side of the inequality in (7) might be problematic due to possible underflows and overflows. The problem can be solved by taking the logarithms of Λ/Λ_D, $(\Lambda_U t)^{k-1}$, $(k-1)!$, and $e^{-p\Lambda_U t}$, adding/subtracting those logarithms to obtain the logarithm of the left-hand side, and applying the exponential function to the result to obtain the value of the left-hand side. Direct computation of the left-hand side of the inequality in (8) might also be problematic. First, the computation of the sum could lead to underflow. This makes it unfeasible to compute the sum in terms of $P_m((1-p)\Lambda_D t)$. There could also be underflows and overflows in what is left. To solve those problems, we rewrite the left-hand side as

$$\frac{\Lambda}{\Lambda_D}\,\frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!}\,e^{-p\Lambda_U t}\,\frac{1}{x}\sum_{m=k}^{\infty} x\,\frac{((1-p)\Lambda_D t)^m}{m!}\,e^{-(1-p)\Lambda_D t}$$

and $\log x$ is selected so that the first term of the scaled summation is equal to one. This yields

$$\log x = (1-p)\Lambda_D t + \log k! - k\log((1-p)\Lambda_D t)\,.$$

The scaled summation can then be computed iteratively up to enough accuracy without problems. Then, the left-hand side can be computed without problems by taking the logarithms of Λ/Λ_D, $(\Lambda_U t)^{k_0-1}$, $(k_0-1)!$, $e^{-p\Lambda_U t}$, $1/x$, and the scaled summation, and applying the exponential function to the result.

The new truncation point N' is also a function of ε, $\Lambda_U t$, Λ_D/Λ_U, and p. Figure 3 plots N and the new truncation point N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$, $\Lambda_U t =$ 10, 10,000, and $p =$ 0.99, 0.999. Figure 4 plots N/N' for the new truncation point N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$, $\Lambda_U t =$ 10, 10,000, and $p =$ 0.99, 0.999. By comparing Figures 1 and 3, we can note that for $\Lambda_D(1-p) > \Lambda_U$ the new truncation point N' is significantly better than the previous truncation point N'. As Figure 4 illustrates, for small ε, the reduction factor N/N' tends for $\Lambda_D/\Lambda_U \to \infty$ to a value approximately equal to $1/(1-p)$. Informally, this is because, for large $(1-p)\Lambda_D t$, $\Delta_r(s)$ is only slightly larger than $(1-p)\Lambda_D t$ and, for $\Lambda_D/\Lambda_U \to \infty$, $\Delta_l(s)$ is much smaller than $\Delta_r(s)$, making $\Delta(s)$ and, for small ε, N' only slightly larger than $(1-p)\Lambda_D t$, whereas, for large Λt and small ε, N is only slightly larger than Λt which, for $\Lambda_D > \Lambda_U$, is equal to $\Lambda_D t$. The method will use the new truncation point N'.

The second truncation will delete the terms in (5) corresponding to values of $k < n - C'''$, where C''' is a second truncation parameter ≥ 0. This gives, using Theorem 2,

$$\mathrm{IAVCD}(t,p) = \mathrm{IAVCD}'_{N',C'''}(t,p) + e'_{N'}(t,p) + e'_{N',C'''}(t,p)\,,$$

$$\begin{aligned}
\mathrm{IAVCD}'_{N',C'''}(t,p) &= \sum_{n=0}^{N'} \sum_{k=\max\{1,n-C'''\}}^{n+1} \Omega^U_{n,k} F^U_{n,k}(t,p) + \sum_{n=1}^{N'} \sum_{k=\max\{1,n-C'''\}}^{n} \Omega^D_{n,k} F^D_{n,k}(t,p) \\
&= \sum_{n=0}^{N'} \Omega^U_{n,n+1}\,\frac{(\Lambda_U t)^n}{n!}\,e^{-\Lambda_U t} + \sum_{n=1}^{N'} \sum_{k=\max\{1,n-C'''\}}^{n} \Omega^U_{n,k}\Lambda_D I_{n,k}(t,p) \\
&\quad + \sum_{n=1}^{N'} \sum_{k=\max\{1,n-C'''\}}^{n} \Omega^D_{n,k}\Lambda_U I_{n,k}(t,p)\,, \qquad (10)
\end{aligned}$$

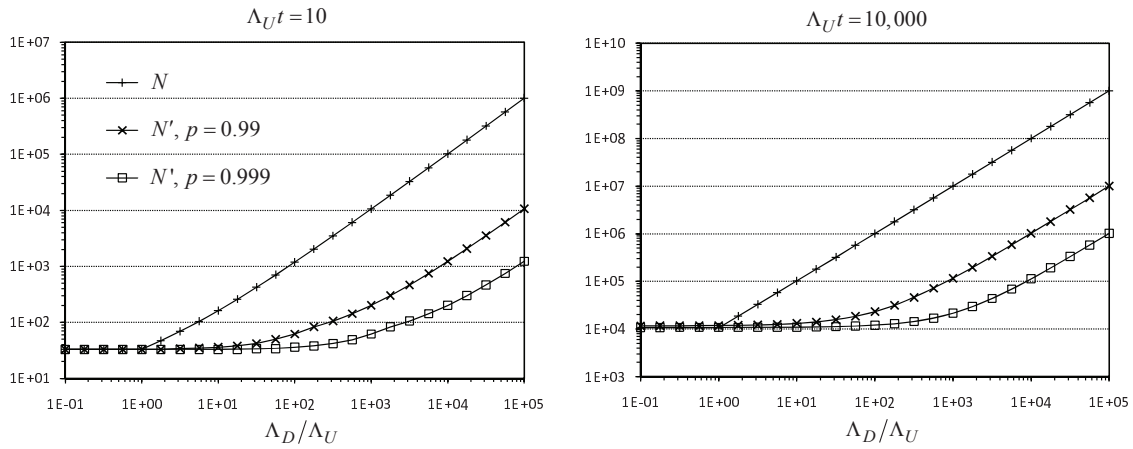

Figure 3: Truncation point N and new truncation point N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$.

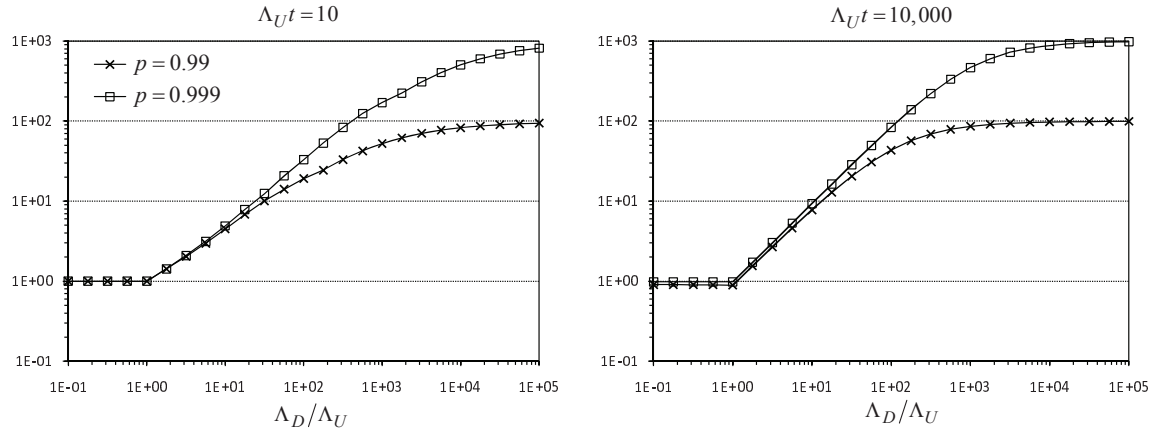

Figure 4: N/N' for the new truncation point N' as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$.

$$0 \le e'_{N',C'''}(t,p) = \sum_{n=C'''+2}^{N'} \sum_{k=1}^{n-C'''-1} \Omega^U_{n,k} F^U_{n,k}(t,p) + \sum_{n=C'''+2}^{N'} \sum_{k=1}^{n-C'''-1} \Omega^D_{n,k} F^D_{n,k}(t,p)\,.$$

Using $\Omega^U_{n,k}, \Omega^D_{n,k} \ge 0$, $C''' + 2 \le n \le N'$, $1 \le k \le n - C''' - 1$, $\sum_{k=1}^{n-C'''-1} \Omega^U_{n,k} + \sum_{k=1}^{n-C'''-1} \Omega^D_{n,k} \le 1$, $C''' + 2 \le n \le N'$, Theorem 2 and (6), we get

$$\begin{aligned}
e'_{N',C'''}(t,p) &\le \sum_{n=C'''+2}^{N'} \max\left\{ \max_{1\le k\le n-C'''-1} F^U_{n,k}(t,p), \max_{1\le k\le n-C'''-1} F^D_{n,k}(t,p) \right\} \\
&= \sum_{n=C'''+2}^{N'} \max_{1\le k\le n-C'''-1} \Lambda I_{n,k}(t,p) \\
&< \sum_{n=C'''+2}^{N'} \max_{1\le k\le n-C'''-1} \frac{\Lambda}{\Lambda_D} \frac{(\Lambda_U t)^{k-1}}{(k-1)!} e^{-p\Lambda_U t} \sum_{m=n+1-k}^{\infty} \frac{((1-p)\Lambda_D t)^m}{m!} e^{-(1-p)\Lambda_D t}\,.
\end{aligned}$$

The factor $(\Lambda_U t)^{k-1}/(k-1)!$ achieves its maximum at $k = k_0 = \lfloor \Lambda_U t \rfloor + 1$. Then, we have

$$\begin{aligned}
e'_{N',C'''}(t,p) &< \sum_{n=C'''+2}^{N'} \frac{\Lambda}{\Lambda_D} \frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!} e^{-p\Lambda_U t} \max_{1\le k\le n-C'''-1} \sum_{m=n+1-k}^{\infty} \frac{((1-p)\Lambda_D t)^m}{m!} e^{-(1-p)\Lambda_D t} \\
&= \sum_{n=C'''+2}^{N'} \frac{\Lambda}{\Lambda_D} \frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!} e^{-p\Lambda_U t} \sum_{m=C'''+2}^{\infty} \frac{((1-p)\Lambda_D t)^m}{m!} e^{-(1-p)\Lambda_D t} \\
&= \mathbf{1}_{C'''\le N'-1}(N' - C''' - 1) \frac{\Lambda}{\Lambda_D} \frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!} e^{-p\Lambda_U t} \\
&\quad \sum_{m=C'''+2}^{\infty} \frac{((1-p)\Lambda_D t)^m}{m!} e^{-(1-p)\Lambda_D t},
\end{aligned}$$

and C''' can be chosen as

$$\begin{aligned}
C''' = \min\Bigg\{ & c \ge \max\{\lfloor (1-p)\Lambda_D t \rfloor - 2, 0\} \,:\, \mathbf{1}_{c\le N'-1}(N' - c - 1) \frac{\Lambda}{\Lambda_D} \frac{(\Lambda_U t)^{k_0-1}}{(k_0-1)!} e^{-p\Lambda_U t} \\
& \sum_{m=c+2}^{\infty} \frac{((1-p)\Lambda_D t)^m}{m!} e^{-(1-p)\Lambda_D t} \le \frac{\varepsilon}{4} \Bigg\}\,. \qquad (11)
\end{aligned}$$

A comment similar to the one addressing the computation of the left-hand side of the inequality in (8) can be made concerning the computation of the left-hand side of the inequality in the expression for C'''.

The truncation point C''' is a function of ε, $\Lambda_D t$, Λ_U/Λ_D, and p. It can be compared with the truncation point C' used in Algorithm A. Figure 5 plots C' and C''' as a function of Λ_U/Λ_D for $\varepsilon = 10^{-8}$, $\Lambda_D t =$ 10, 10,000, and $p =$ 0.99, 0.999. Figure 6 plots C'/C''' as a function of Λ_U/Λ_D for $\varepsilon = 10^{-8}$, $\Lambda_D t =$ 10, 10,000, and $p =$ 0.99, 0.999. For large $\Lambda_D t$ but Λ_U/Λ_D around 1, C''' is slightly larger than C'. For Λ_U significantly larger than Λ_D, C''' is moderately smaller than C'.

The new method uses the formulation for IAVCD(t,p) given by (10), with N' and C''' selected using, respectively, (9) and (11), guaranteeing a nonnegative truncation error $\le \varepsilon/2$.

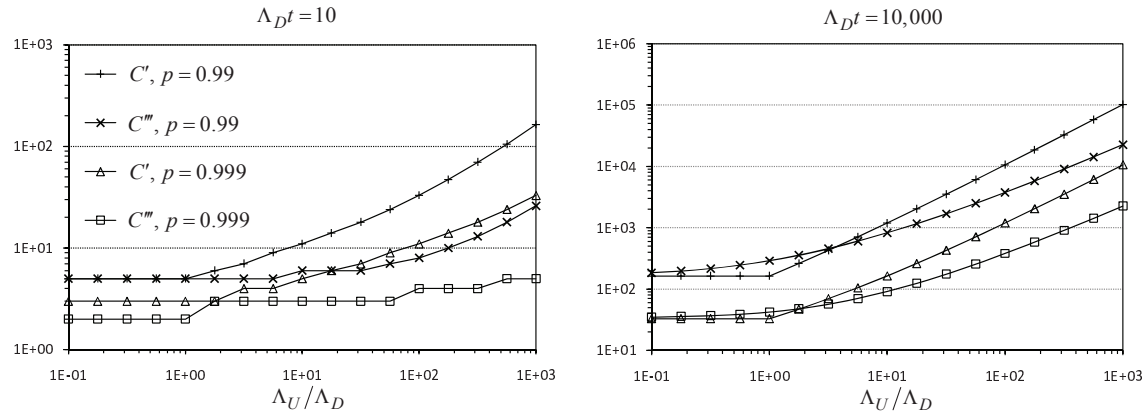

Figure 5: Truncation points C' and C''' as a function of Λ_U/Λ_D for $\varepsilon = 10^{-8}$.

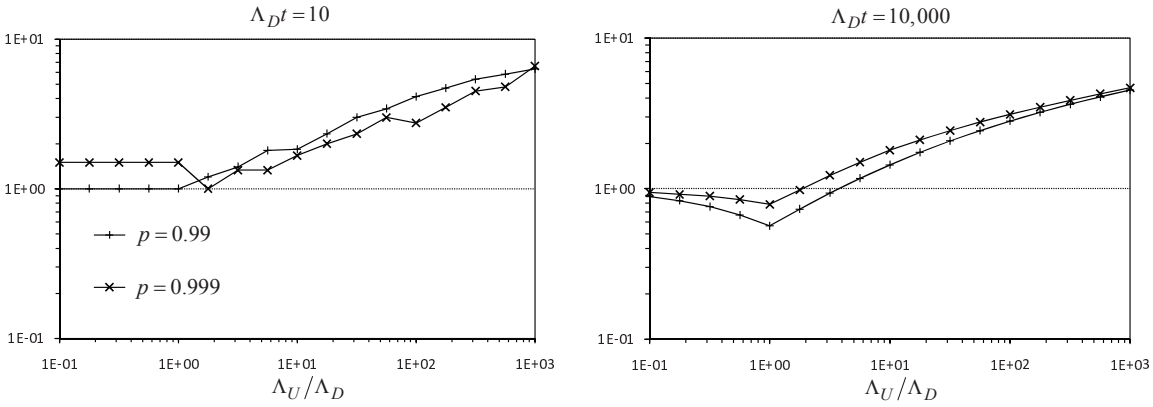

Figure 6: C'/C''' as a function of Λ_U/Λ_D for $\varepsilon = 10^{-8}$.

3.3 Computation of $\Omega^U_{n,k}$ and $\Omega^D_{n,k}$

We discuss next how the $\Omega^U_{n,k}$ and $\Omega^D_{n,k}$ involved in (10) can be computed. Let $\Omega^i_{n,k} = P[\#(\widehat{X}_{0:n} \in U) = k \wedge \widehat{X}_n = i]$ and let $\mathbf{\Omega}_{n,k}$ be the column vector $(\Omega^i_{n,k})_{i\in\Omega}$. Then, it is clear (3), (4) that $\Omega^U_{n,k} = \sum_{i\in U} \Omega^i_{n,k}$ and $\Omega^D_{n,k} = \sum_{i\in D} \Omega^i_{n,k}$. This translates the problem of computing the $\Omega^U_{n,k}$ and $\Omega^D_{n,k}$ involved in $\mathrm{IAVCD}'_{N',C'''}(t,p)$ to the problem of computing the vectors $\mathbf{\Omega}_{n,k}$ for the required (n,k) pairs: $0 \le n \le N'$, $\max\{1, n - C'''\} \le k \le n+1$. Let $\boldsymbol{\alpha}^U$ and $\boldsymbol{\alpha}^D$ denote the column subvectors of $\boldsymbol{\alpha}$ including the components with indices in, respectively, U and D, and let $\mathbf{\Omega}^U_{n,k}$ and $\mathbf{\Omega}^D_{n,k}$ denote the column subvectors of $\mathbf{\Omega}_{n,k}$ including the components with indices in, respectively, U and D. Then, the vectors $\mathbf{\Omega}_{n,k}$ for the required (n,k) pairs can be computed using for increasing n, $0 \le n \le N'$, and, for each n, for increasing k, $\max\{0, n - C'''\} \le k \le n+1$, the recurrences

$$(\mathbf{\Omega}^U_{n,k})^{\mathrm{T}} = \mathbf{\Omega}^{\mathrm{T}}_{n-1,k-1}\mathbf{P}_{\Omega,U}\,, \quad n \ge 1,\ 1 \le k \le n+1\,, \tag{12}$$

$$(\mathbf{\Omega}^D_{n,k})^{\mathrm{T}} = \mathbf{\Omega}^{\mathrm{T}}_{n-1,k}\mathbf{P}_{\Omega,D}\,, \quad n \ge 1,\ 0 \le k \le n\,, \tag{13}$$

$$\mathbf{\Omega}^U_{n,0} = \mathbf{0}\,, \quad n \ge 0\,, \tag{14}$$

$$\mathbf{\Omega}^D_{n,n+1} = \mathbf{0}\,, \quad n \ge 0 \tag{15}$$

with initial conditions

$$\mathbf{\Omega}^U_{0,1} = \boldsymbol{\alpha}^U\,, \tag{16}$$

$$\mathbf{\Omega}^D_{0,0} = \boldsymbol{\alpha}^D\,. \tag{17}$$

3.4 Computation of $I_{n,k}(t,p)$

It remains to discuss the computation of the integrals $I_{n,k}(t,p)$ appearing in (10): $I_{n,k}(t,p)$, $1 \le n \le N'$, $\max\{1, n - C'''\} \le k \le n$. Computation of these integrals can be tricky. It will turn out convenient to calculate instead

$$J_{n,k}(t,p) = \Lambda I_{n,k}(t,p) = \int_0^{(1-p)t} \Lambda\, \frac{(\Lambda_U(t-x))^{k-1}}{(k-1)!}\, e^{-\Lambda_U(t-x)}\, \frac{(\Lambda_D x)^{n-k}}{(n-k)!}\, e^{-\Lambda_D x}\, dx\,,$$

where $\Lambda = \max\{\Lambda_U, \Lambda_D\}$, and obtain $I_{n,k}(t,p)$ from $J_{n,k}(t,p)$ using $I_{n,k}(t,p) = J_{n,k}(t,p)/\Lambda$. The following theorem identifies recurrences that can be used to compute most of the $J_{n,k}(t,p)$.

Theorem 4. *For $\Lambda_U > \Lambda_D$, assume that $J_{n,n}(t,p)$, $1 \le n \le N'$ and $J_{N',k}(t,p)$, $\max\{1, N' - C'''\} \le k \le N'-1$ are known with absolute errors $\le \delta$. Then, in exact arithmetic, $J_{n,k}(t,p)$, $2 \le n \le N'-1$, $\max\{1, n - C'''\} \le k \le n-1$ can be computed with absolute errors $\le \delta$ from $J_{n,n}(t,p)$, $1 \le n \le N'$ and $J_{N',k}(t,p)$, $\max\{1, N' - C'''\} \le k \le N'-1$ for decreasing n and, for each n, for decreasing k using the recurrence*

$$\begin{aligned} J_{n,k}(t,p) \;=\; & \frac{\Lambda_U - \Lambda_D}{\Lambda_U}\, J_{n+1,k+1}(t,p) + \frac{\Lambda_D}{\Lambda_U}\, J_{n,k+1}(t,p) \\ & - \frac{(p\Lambda_U t)^k}{k!}\, e^{-p\Lambda_U t}\, \frac{((1-p)\Lambda_D t)^{n-k}}{(n-k)!}\, e^{-(1-p)\Lambda_D t}\,. \end{aligned}$$

For $\Lambda_U \leq \Lambda_D$, assume that $J_{n,\max\{1,n-C'''\}}(t,p)$, $1 \leq n \leq N'$ and $J_{N',k}(t,p)$, $\max\{1,N'-C'''\}+1 \leq k \leq N'$ are known with absolute errors $\leq \delta$. Then, in exact arithmetic, $J_{n,k}(t,p)$, $2 \leq n \leq N'-1$, $\max\{1,n-C'''\}+1 \leq k \leq n$ can be computed with absolute errors $\leq \delta$ from $J_{n,\max\{1,n-C'''\}}(t,p)$, $1 \leq n \leq N'$ and $J_{N',k}(t,p)$, $\max\{1,N'-C'''\}+1 \leq k \leq N'$ for decreasing n and, for each n, for increasing k using the recurrence

$$\begin{aligned} J_{n,k}(t,p) &= \frac{\Lambda_D-\Lambda_U}{\Lambda_D}\, J_{n+1,k}(t,p) + \frac{\Lambda_U}{\Lambda_D}\, J_{n,k-1}(t,p) \\ &\quad + \frac{(p\Lambda_U t)^{k-1}}{(k-1)!}\, e^{-p\Lambda_U t}\, \frac{((1-p)\Lambda_D t)^{n+1-k}}{(n+1-k)!}\, e^{-(1-p)\Lambda_D t}\,. \end{aligned}$$

Proof. See the online supplement. □

Note that having absolute errors $\leq \delta$ in $J_{n,k}(t,p)$ implies having absolute errors $\leq \delta$ in $\Lambda_D I_{n,k}(t,p) = (\Lambda_D/\Lambda) J_{n,k}(t,p)$ and $\Lambda_U I_{n,k}(t,p) = (\Lambda_U/\Lambda) J_{n,k}(t,p)$, and, since $\Omega^U_{n,k}, \Omega^D_{n,k} \geq 0$, $1 \leq n \leq N'$, $\max\{1,n-C'''\} \leq k \leq n$ and $\sum_{k=\max\{1,n-C'''\}}^{n} \Omega^U_{n,k} + \sum_{k=\max\{1,n-C'''\}}^{n} \Omega^D_{n,k} \leq 1$, $1 \leq n \leq N'$, it implies having an absolute error $\leq N'\delta$ in $\mathrm{IAVCD}'_{N',C'''}(t,p)$. If we pick $\delta = \varepsilon/(2N')$, we can allow an absolute truncation error $\leq \delta$ in a numerical evaluation of the integrals $J_{n,k}(t,p)$ on which the recurrences are based with the result of introducing in $\mathrm{IAVCD}'_{N',C'''}(t,p)$ an absolute truncation error $\leq \varepsilon/2$, that added to the truncation error with which $\mathrm{IAVCD}'_{N',C'''}(t,p)$ gives $\mathrm{IAVCD}(t,p)$ results in the computation of $\mathrm{IAVCD}(t,p)$ with absolute truncation error $\leq \varepsilon$.

Allowing an absolute error $\leq \delta$ in the numerical evaluation of the integrals $J_{n,k}(t,p)$ assumed in the recurrences given by Theorem 4 makes it in many cases possible to restrict the integration domain to a much narrower interval than $[0,(1-p)t]$, reducing substantially the computational cost of the numerical evaluation. As numerical integration method we chose Romberg's method (see, for instance, Dahlquist and Björch 1974), which is efficient for smooth integrands. To truncate the integration interval we identify a reasonably tight interval for $\lambda \geq 0$ out of which the Poisson probability $P_k(\lambda) = (\lambda^k/k!)\, e^{-\lambda}$, $k \geq 0$ is smaller than a given small $\eta > 0$. To identify that integration interval, the basic idea is to consider that $P_k(\lambda)$, as a function of λ, is the probability density function of a $k+1$–Erlang random variable with parameter 1, which has mean $k+1$ and standard deviation $\sqrt{k+1}$. Furthermore, $P_k(\lambda)$ is increasing for $0 \leq \lambda \leq k$ and decreasing for $\lambda \geq k$. Then, for large k, $P_k(\lambda)$ should get tiny for values of λ several $\sqrt{k+1}$ apart from $k+1$. Let $T_R(k,\eta) = k+1+m_R(k,\eta)\sqrt{k+1}$ with $m_R(k,\eta) = \min\{m \geq 0 : P_k(k+1+m\sqrt{k+1}) \leq \eta\}$. Also, for $k=0$ or $k>0$ and $P_k(k+1-\lfloor\sqrt{k+1}\rfloor\sqrt{k+1}) > \eta$, define $T_L(k,\eta) = 0$ and, otherwise, define $T_L(k,\eta) = k+1-m_L(k,\eta)\sqrt{k+1}$ with $m_L(k,\eta) = \min\{m \geq 1 : P_k(k+1-m\sqrt{k+1}) \leq \eta\}$. We have $P_k(\lambda) \leq \eta$ for λ outside the interval $[T_L(k,\eta), T_R(k,\eta)]$. The following theorem defines the truncation of the integration interval $[0,(1-p)t]$ in terms of these truncation points for Poisson probabilities.

Theorem 5. *Let $\delta > 0$, $t > 0$, $0 < p < 1$, $k \geq 1$, $n \geq k$. Let $g_{n,k}(x)$ be the integrand in $J_{n,k}(t,p)$ and let $r_1 = \min\{n-k, \Lambda_D(1-p)t\}$, $r_2 = \max\{p\Lambda_U t, \min\{k-1, \Lambda_U t\}\}$,*

$$T_L^1 = T_L\left(k-1, \frac{\delta}{2(1-p)\Lambda t P_{n-k}(r_1)}\right),$$

$$T_R^1 = T_R\left(k-1, \frac{\delta}{2(1-p)\Lambda t P_{n-k}(r_1)}\right),$$
$$T_L^2 = T_L\left(n-k, \frac{\delta}{2(1-p)\Lambda t P_{k-1}(r_2)}\right),$$
$$T_R^2 = T_R\left(n-k, \frac{\delta}{2(1-p)\Lambda t P_{k-1}(r_2)}\right),$$

$L_1 = \max\{0, t - T_R^1/\Lambda_U\}$, $R_1 = \min\{(1-p)t, t - T_L^1/\Lambda_U\}$, $L_2 = T_L^2/\Lambda_D$, *and* $R_2 = \min\{(1-p)t, T_R^2/\Lambda_D\}$. *Then; for* $L_1 \geq R_1$ *or* $L_2 \geq R_2$ *or* $L_1 < R_1$, $L_2 < R_2$, $L_1 \leq L_2$, *and* $R_1 \leq L_2$, *or* $L_1 < R_1$, $L_2 < R_2$, $L_1 > L_2$, *and* $R_2 \leq L_1$, $J_{n,k}(t,p) = 0$ *with nonnegative truncation error* $\leq \delta$; *for* $L_1 < R_1$, $L_2 < R_2$, $L_1 > L_2$, $R_2 > L_1$, *and* $R_1 \leq R_2$, $J_{n,k}(t,p) = \int_{L_1}^{R_1} g_{n,k}(x)\,dx$ *with nonnegative truncation error* $\leq \delta$; *for* $L_1 < R_1$, $L_2 < R_2$, $L_1 > L_2$, $R_2 > L_1$, *and* $R_1 > R_2$, $J_{n,k}(t,p) = \int_{L_1}^{R_2} g_{n,k}(x)\,dx$ *with nonnegative truncation error* $\leq \delta$; *for* $L_1 < R_1$, $L_2 < R_2$, $L_1 \leq L_2$, $R_1 > L_2$, *and* $R_2 > R_1$, $J_{n,k}(t,p) = \int_{L_2}^{R_1} g_{n,k}(x)\,dx$ *with nonnegative truncation error* $\leq \delta$; *and, for* $L_1 < R_1$, $L_2 < R_2$, $L_1 \leq L_2$, $R_1 > L_2$, *and* $R_2 \leq R_1$, $J_{n,k}(t,p) = \int_{L_2}^{R_2} g_{n,k}(x)\,dx$ *with nonnegative truncation error* $\leq \delta$.

Proof. See the online supplement. □

By extensive experimentation, we recognized that Romberg's method used with 1,024 integration subintervals was enough to perform the numerical integration with negligible relative error, and we use Romberg's method with that setting. The integration domain truncation procedure seems to be very efficient. When the number of (n,k) pairs for which $J_{n,k}(t,p)$ have to be computed directly is large, often, most of the integrals which have to be evaluated numerically are detected to be 0 with absolute error $\leq \delta$, and the number of integrals that are actually evaluated numerically is relatively small, reducing considerably the average computational cost of the evaluation of the integrals.

3.5 Description and computational cost

To clarify, we provide next a short description of the new method. After that, the numerical stability and the computational cost of the method will be discussed. We assume that $\mathrm{IAVCD}(t,p)$ has to be computed at a single (t,p) pair. First, we determine the truncation point N' using (9) and the truncation point C''' using (11). Next, we obtain the transition matrix $\mathbf{P} = \mathbf{I} + \mathbf{\Lambda}_{UD}^{-1}\mathbf{A}$ of the randomized DTMC $\widehat{X}$ considered in the method. After that, we obtain the vectors $\mathbf{\Omega}_{n,k}$, $0 \leq n \leq N'$, $\max\{0, n - C'''\} \leq k \leq n+1$ for increasing n and, for each n, for increasing k using the recurrences (12)–(17). As those vectors are obtained, we compute and store $\Omega_{n,k}^U = \sum_{i \in U} \Omega_{n,k}^i$, $0 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n+1$ and $\Omega_{n,k}^D = \sum_{i \in D} \Omega_{n,k}^i$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$. Next, if $\Lambda_U > \Lambda_D$, we compute, as described by Theorem 4, the integrals $J_{n,k}(t,p)$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$, with the integrals $J_{n,n}(t,p)$, $1 \leq n \leq N'$ and $J_{N',k}(t,p)$, $\max\{1, N' - C'''\} \leq k \leq N' - 1$ evaluated numerically with nonnegative truncation error $\leq \varepsilon/(2N')$ using Theorem 5. If $\Lambda_U \leq \Lambda_D$, we compute, as described by Theorem 4, the integrals $J_{n,k}(t,p)$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$, with the integrals $J_{n,\max\{1,n-C'''\}}(t,p)$, $1 \leq n \leq N'$ and $J_{N',k}(t,p)$, $\max\{1, N' - C'''\} + 1 \leq k \leq N'$ evaluated numerically with

nonnegative truncation error $\leq \varepsilon/(2N')$ using Theorem 5. As $J_{n,k}(t,p)$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$ are obtained, we compute $I_{n,k}(t,p)$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$ using $I_{n,k}(t,p) = J_{n,k}(t,p)/\Lambda$ and accumulate terms in the approximate value for $\mathrm{IAVCD}(t,p)$, $\mathrm{IAVCD}_{N',C'''}(t,p)$, given by (10).

The numerical stability of the method comes from the fact that, once the truncation parameters N' and C''' are known, no subtractions are involved in the computations leading to the approximate value for $\mathrm{IAVCD}(t,p)$ except in the computation of the diagonal elements of matrix $\mathbf{P}$, in the method we use for computing Poisson probabilities, in the application of Romberg's method to the numerical evaluation of some $J_{n,k}(t,p)$, and in the recurrences of Theorem 4. The diagonal elements of $\mathbf{P}$ either are very small or are computed with numerical stability. The method we use (Knüsel 1986, pp. 1028–1029) for computing Poisson probabilities is numerically stable. Romberg's method applied to nonnegative smooth integrands is also numerically stable, and the recurrence for $J_{n,k}(t,p)$ will not introduce significant relative round-off errors for the case $\Lambda_U \leq \Lambda_D$ and is very unlikely to do so for the case $\Lambda_U > \Lambda_D$, because, in that case, significant cancellations can only happen when the $J_{n,k}(t,p)$ computed in the recurrence is very small compared to either $J_{n+1,k+1}(t,p)$ or $J_{n,k+1}(t,p)$, and the integrals $J_{n,k}(t,p)$ can be expected to be smooth functions of both n and k for large n and k. In summary, the new method will be numerically stable. This will be confirmed in §4.1 through thorough experimentation.

We next compare the computational costs of the new method and Algorithm A, and start by discussing the CPU time. The average cost of computing the integrals $J_{n,k}(t,p)$ in the new method does not seem to be much larger than the average cost of computing the factors $((\Lambda t)^n/n!))\, e^{-\Lambda t}\binom{n}{k}p^k(1-p)^{n-k}$ in Algorithm A and the cost of computing the truncation parameters in both methods will often be similar. Then, the numbers of points in the domain of (n,k) pairs for which, in the new method, $\mathbf{\Omega}_{n,k}$ has to be obtained and, in Algorithm A, $\mathbf{Y}_{n,k}$ has to be obtained, are reasonable relative estimates of the costs in terms of CPU times of both methods for large models. Rough estimates for those numbers of points are $P' = (N'+1)(C''+2)$ for the new method and $P = (N+1)(C'+1)$ for Algorithm A. Then, a reasonable, approximate measure of the speedup of the new method with respect to Algorithm A for large models is $S = P/P'$. Figure 7 plots S as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$, $\Lambda t = 100$, 100,000, and $p =$ 0,2, 0.5, 0.9, 0.95, 0.99, 0.999, 0.9999. For p close to 1, there are small regions around $\Lambda_D/\Lambda_U = 1$ in which $S < 1$; further apart from $\Lambda_D/\Lambda_U = 1$, S becomes > 1. The speedup measure S is significantly larger than 1 for $\Lambda_D > \Lambda_U$ and moderately for $\Lambda_D < \Lambda_U$ and increases with Λt. This suggests that the new method can be much faster than Algorithm A for $\Lambda_D > \Lambda_U$ and moderately faster for $\Lambda_D < \Lambda_U$, and that the speedup will increase with Λt. In §4.2 we will numerically corroborate this approximate analysis and will illustrate that S is a reasonable approximate measure of the speedup for large models. The truncation points N, C', N' and C''' can be computed a priori. From them, we can compute S and decide to use the new method if $S > 1$ and to use Algorithm A otherwise. Because, for large models, the computational cost associated with the determination of N, C', N', and C''' will be relatively small, the switch between the new method and Algorithm A will be relatively inexpensive in that case. Regarding memory cost, assuming matrix $\mathbf{P}$ rewrites matrix $\mathbf{A}$, Algorithm A has a cost $O((C'+3)|\Omega|)$ be-

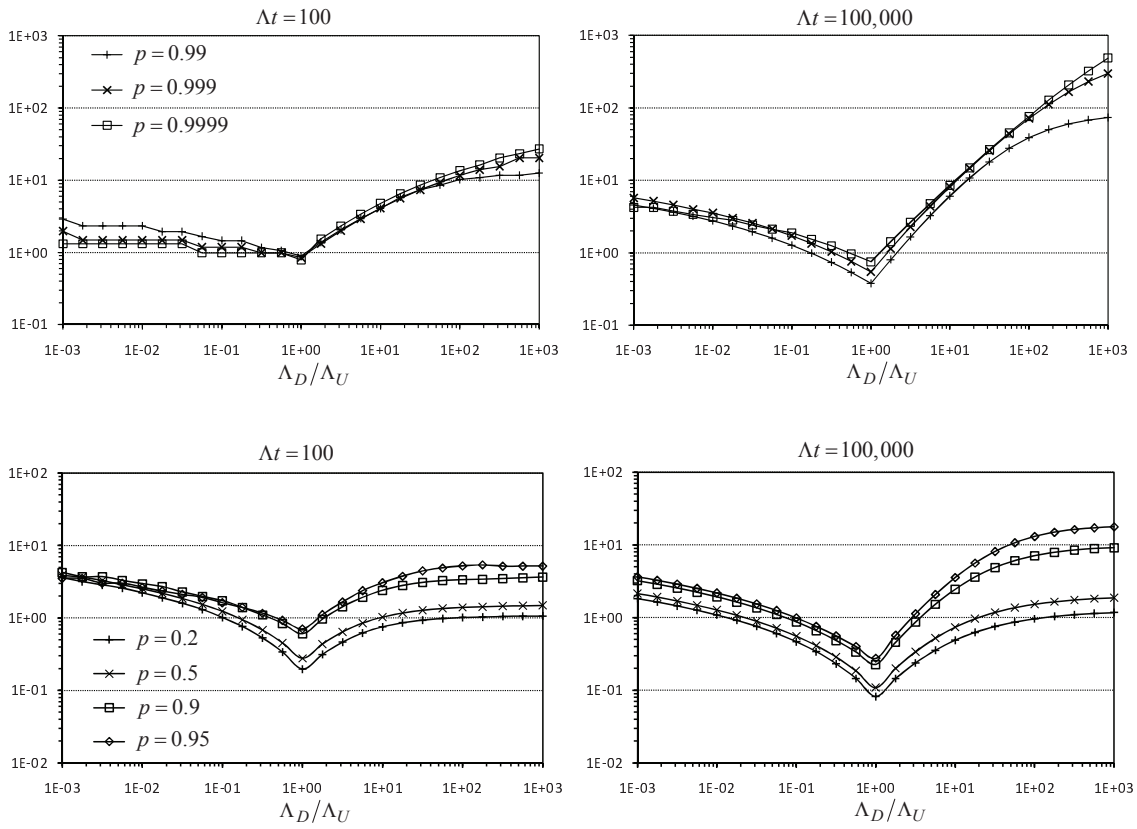

Figure 7: S as a function of Λ_D/Λ_U for $\varepsilon = 10^{-8}$.

cause of the $C' + 3$ vectors of size $|\Omega|$ needed to hold $\mathbf{Y}_{n,k}$, taking into account the ordering in which the $\mathbf{Y}_{n,k}$ are obtained. Assuming matrix $\mathbf{P}$ rewrites matrix $\mathbf{A}$, the new method has memory cost $O((C''' + 3)|\Omega| + 2P' + C''' + 3) = O((C''' + 3)|\Omega| + 2N'C''')$, because of the $C''' + 3$ vectors of size $|\Omega|$ needed to hold $\mathbf{\Omega}_{n,k}$, taking into account the ordering in which the $\mathbf{\Omega}_{n,k}$ are obtained, the storage required to hold $\Omega^U_{n,k}$, $0 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n + 1$ and $\Omega^D_{n,k}$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$, and the storage required to hold $J_{n,k}(t, p)$, $1 \leq n \leq N'$, $\max\{1, n - C'''\} \leq k \leq n$, taking into account the ordering in which those integrals are computed.

To end, if $\mathrm{IAVCD}(t, p)$ had to be computed at several (t, p) pairs, a plausible, simple approach would be to compute the required N' and C''' for each (t, p) pair, obtain the vectors $\mathbf{\Omega}_{n,k}$ and $\Omega^U_{n,k}$, $\Omega^D_{n,k}$ for the (n, k) domains obtained by taking the maximum N' and the maximum C''', but when using (10) for each (t, p) pair, take the N' and C''' corresponding to that pair.

4 Numerical Analysis

This section analyzes the new method for the computation of the interval availability distribution developed in §3. The section includes two subsections. In §4.1, we analyze the numerical stability of the method using a CTMC example with closed-form solution. Section 4.2 illustrates, using two

large CTMC models, that the new method can be significantly less costly in terms of CPU time than Algorithm A, corroborating numerically the approximate analysis regarding the relative costs of both methods performed at the end of §3. Another goal of §4.2 is to validate the switch between the new method and Algorithm A described in §3, which is based on the approximate speedup measure S. All floating-point computations were performed in an environment conforming to the standard IEEE 754 for floating-point arithmetic (IEEE 1985), using the double format and the default rounding mode Round to Nearest. In that environment, $\mathrm{EPS} = 2.2204 \times 10^{-16}$, where EPS is the machine epsilon of the computer (difference between the smallest exactly representable number greater than 1 and 1 (Higham 2002)), and the absolute relative round-off error introduced when performing a basic arithmetic operation resulting in a normalized number (this can be expected to always be the case) is bounded from above by $\mathrm{EPS}/2$.

4.1 Test of numerical stability using an example

We will use the CTMC model with the state diagram of Figure 8 (left-hand side), subset of up states $\{1_1, \ldots, 1_{50}, 2_1, \ldots, 2_{50}\}$ and initial state 1_1. In that CTMC model, there is a transition rate with value $\rho/50$ from every state 1_i to every state 2_j; a transition rate with value $\rho/50$ from every state 2_i to every state 1_j; a transition rate with value λ from every state 1_i and every state 2_i to state 3; and a transition rate with value $\mu/100$ from state 3 to every state 1_i and every state 2_i. That CTMC is ordinarily lumpable with respect to the partition of the state space $\{\{1_1, \ldots, 1_{50}, 2_1, \ldots, 2_{50}\}, \{3\}\}$ (see, for instance, Buchholz 1994) and the lumped CTMC has the state diagram of Figure 8 (right-hand side) and initial state 1. That lumped two-state CTMC has with subset of up states $\{1\}$ the same interval availability distribution as the CTMC model. We take $\lambda = 5 \times 10^{-4}$, $\mu = 1$, and values of ρ varying between 5×10^{-4} and $10^3 - 5 \times 10^{-4}$, so that $\Lambda_D/\Lambda_U = \mu/(\lambda + \rho)$ varies between 10^{-3} and 1,000. For the CTMC model, $\Lambda = \max\{\lambda + \rho, \mu\}$ and, for each value of ρ, we consider two values of t: the first one chosen so that $\Lambda t = 10$ and the second one chosen so that $\Lambda t = 100{,}000$. We consider three values for p: 0.99, 0.999 and 0.9999. We ran the method with a tiny truncation error target $\varepsilon = 10^{-26}$ to isolate the impact of round-off errors, and, using a known closed-form solution (see Takács (1957)) for the interval availability distribution of the lumped two-state CTMC model, computed the absolute relative error in the numerical solution given by the method. That closed-form solution is

$$\mathrm{IAVCD}(t,p) = e^{-\lambda p\, t}\left[1 + \sqrt{\lambda\mu p\, t}\int_0^{(1-p)t} \frac{e^{-\mu y}}{\sqrt{y}} I_1(2\sqrt{\lambda\mu p\, t y})\, dy\right],$$

where $I_1(x)$ is the modified Bessel function of first kind and order 1. For $\Lambda t = 10$, the exact value of $\mathrm{IAVCD}(t,p)$ ranged from 0.995017948786 to 0.999995050507. For $\Lambda t = 100{,}000$, the exact value of $\mathrm{IAVCD}(t,p)$ ranged from 0.000000024579 to 1,000000000000. Figure 9 gives the absolute relative errors for $\Lambda t = 10$ (left) and $\Lambda t = 100{,}000$ (right) against N'. The reason is that we can expect round-off errors in $\Omega^U_{n,k}$ and $\Omega^D_{n,k}$ to increase with N' because the longest dependency chain in the recurrences (12)–(15) has length N'. We note that the absolute relative errors are very small in all cases and depend on N' approximately linearly. For the CTMC model considered and the quite representative cases considered, we get in the worst case approximately 10 digits of accuracy.

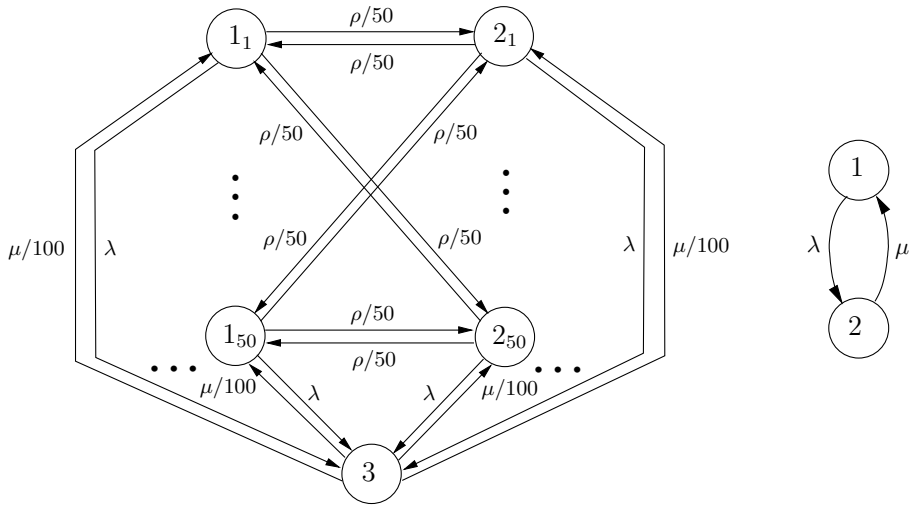

Figure 8: State diagram of CTMC model (left) and state diagram of lumped two-state CTMC model (right).

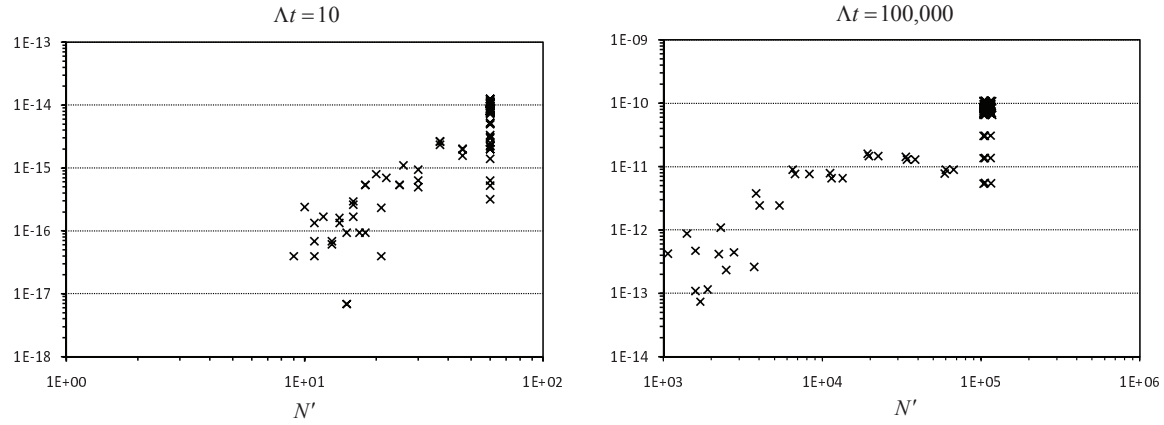

Figure 9: Absolute relative errors in the new method against N'.

In the CTMC model considered, the number of non-zero elements in every column of $\mathbf{P}_{\Omega,U}$ is 51 and the number of non-zero elements in the single column of $\mathbf{P}_{\Omega,D}$ is 100. Certainly, the errors should get larger as the numbers of non-zero elements in the columns of $\mathbf{P}_{\Omega,U}$ and the numbers of non-zero elements in the columns of $\mathbf{P}_{\Omega,D}$ increase, but the CTMC model considered looks like a hard enough test to support the numerical stability of the method.

4.2 Analysis of computational cost using two examples

The first large CTMC model corresponds to a software system with progressive software upgrades. The system includes three software subsystems. Each software subsystem is subject to ten upgrades. The mean time between consecutive upgrades is exponentially distributed with parameter $\rho = 1/720\mathrm{h}^{-1}$, yielding an average time between successive upgrades of about one month. Software upgrades reduce the failure rate of the software subsystem. The failure rate of the first software sub-

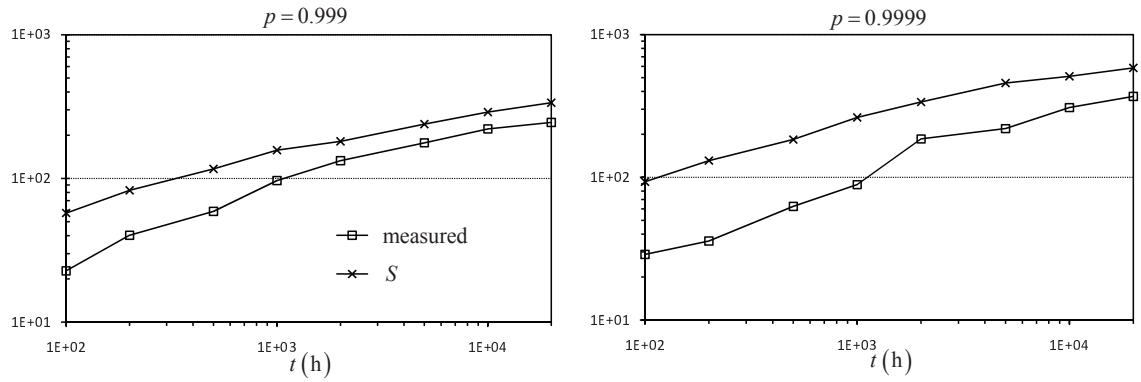

Figure 10: Measured speedup of the new method over Algorithm A and S for the CTMC model of the software system.

system after the ith upgrade is $\lambda_{1,i} = 10^{-4} + (10-i)(4\times10^{-5})\,\mathrm{h}^{-1}$. The failure rate of the second software subsystem after the ith upgrade is $\lambda_{2,i} = 5\times10^{-5} + (10-i)(2\times10^{-5})\mathrm{h}^{-1}$. The failure rate of the third software subsystem after the ith upgrade is $\lambda_{3,i} = 2\times10^{-5} + (10-i)(8\times10^{-6})\mathrm{h}^{-1}$. The three software subsystems have to be operational for the system to be up. Software subsystems can fail in two modes. The first mode occurs with probability 0.8 and is recovered by a restart operation whose duration has an exponential distribution with parameter $\mu = 6\,\mathrm{h}^{-1}$. The second mode occurs with probability 0.2 and is recovered by a manual repair operation whose duration is exponentially distributed with parameter $\xi = 0.5\,\mathrm{h}^{-1}$. When the system is down, software subsystems do not fail and software upgrades are suspended. The initial state of the CTMC model is the state in which the three software subsystems are operational and without upgrades. The CTMC model has 9,317 states, 19,602 transitions, $\Lambda_U = 5.0167\times10^{-3}\,\mathrm{h}^{-1}$, and $\Lambda_D = 6\,\mathrm{h}^{-1}$, yielding $\Lambda_D/\Lambda_U = 1{,}196$. Thus, according to the analysis performed in §3.5, we can expect the new method to be significantly faster than Algorithm A for p close to 1. Table 1 gives $\mathrm{IAVCD}(t,p)$ for $p = 0.999, 0.9999$ and several values of t varying from 100 to 20,000 h. The table also gives the CPU times in seconds of the new method and Algorithm A for each (t,p) pair. The methods were run with a truncation error target $\varepsilon = 10^{-8}$ and CPU times were measured on a multiprocessor with 16 Xeon X7350 2.93 GHz cores, with the method running on a single core and without any other significant process running. Figure 10 compares measured speedups of the new method over Algorithm A with the speedup measure S. We can note that, as expected, the new method is much faster than Algorithm A. Measured speedups differ somehow from S in some cases. Those differences must be attributed to the different costs of the computation of the truncation parameters in both methods and to the difference between the average cost of the computation of the integrals $J_{n,k}(t,p)$ in the new method and the average cost of the computation of the factors $((\Lambda t)^n/n!)e^{-\Lambda t}\binom{n}{k}p^k(1-p)^{n-k}$ in Algorithm A. For larger CTMC models, S will be a more accurate speedup measure.

The second large CTMC model corresponds to a fault-tolerant control system including six control sites. Each site includes two hardware modules working in dual configuration. The failure rates of the hardware modules are $\lambda_1 = 5\times10^{-4}\,\mathrm{h}^{-1}$ for modules in site 1, $\lambda_2 = 4.5\times10^{-4}\,\mathrm{h}^{-1}$ for modules in site 2, $\lambda_3 = 4\times10^{-4}\,\mathrm{h}^{-1}$ for modules in site 3, $\lambda_4 = 3.5\times10^{-4}\,\mathrm{h}^{-1}$ for modules in

Table 1: $\mathrm{IAVCD}(t,p)$ and CPU times in seconds of the new method (N) and Algorithm A (A) for the CTMC model of the software system.

t (h)	p	$\mathrm{IAVCD}(t,p)$	CPU time (N)	CPU time (A)
100	0.999	0.94806210	0.09201	2.092
100	0.9999	0.92265401	0.04400	1.268
200	0.999	0.93025187	0.1240	4.988
200	0.9999	0.85846616	0.06400	2.288
500	0.999	0.91603409	0.2720	16.06
500	0.9999	0.72102120	0.1120	7.004
1,000	0.999	0.89734409	0.4720	45.57
1,000	0.9999	0.59391149	0.1920	17.05
2,000	0.999	0.87869758	0.9921	131.9
2,000	0.9999	0.48085218	0.2360	43.83
5,000	0.999	0.91580678	3.308	585.5
5,000	0.9999	0.39794869	0.6840	150.1
10,000	0.999	0.97648531	8.705	1,924
10,000	0.9999	0.43142930	1.408	433.9
20,000	0.999	0.99860736	26.42	6,486
20,000	0.9999	0.47579569	3.452	1,274

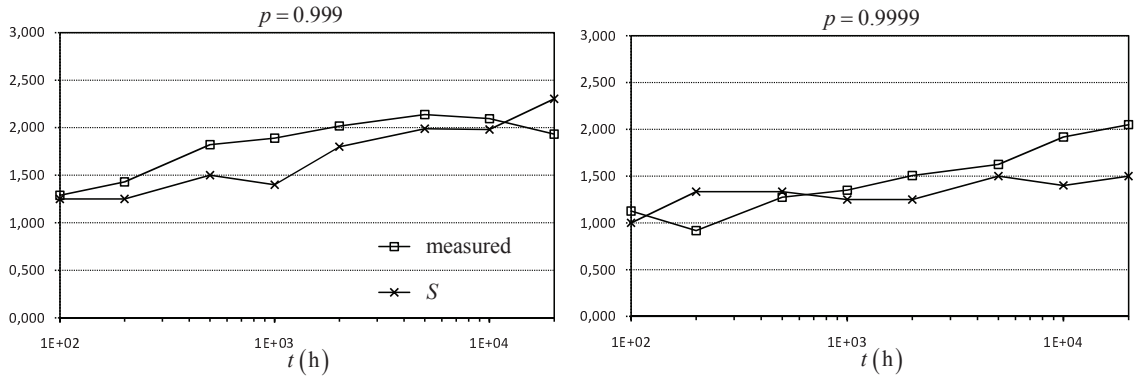

Figure 11: Measured speedup of the new method over Algorithm A and S for the CTMC model of the fault-tolerant control system.

site 4, $\lambda_5 = 3 \times 10^{-4}\,\mathrm{h}^{-1}$ for modules in site 5, and $\lambda_6 = 2.5 \times 10^{-4}\,\mathrm{h}^{-1}$ for modules in site 6. The system is up if all sites are operational. A site is operational if it has no failed module or one module in covered fault. The coverage to faults of hardware modules is $C = 0.98$. Modules in covered fault are repaired at rate $\mu_{\mathrm{C}} = 6\,\mathrm{h}^{-1}$. Modules in uncovered fault or failed modules in sites with both modules failed are repaired at rate $\mu_{\mathrm{U}} = 0.2\,\mathrm{h}^{-1}$. When both modules of a site are failed and one is repaired, the other one is considered to become in covered fault. The much higher repair rate of modules in covered fault is explained by the fact that the repair of those modules only involves the replacement of the module, while, otherwise, the repair of the module, in addition to its replacement, requires a lengthy diagnosis process. Hardware modules continue to fail when the system is down. There is a single repairman who gives preemptive priority to modules in uncovered fault and who is shared by all failed modules with same repair priority. The initial state of the CTMC model is the state in which all sites are operational with no failed module. The CTMC model has 4,096 states, 37,056 transitions, $\Lambda_U = 6.00425\,\mathrm{h}^{-1}$, and $\Lambda_D = 0.20425\,\mathrm{h}^{-1}$, yielding $\Lambda_D/\Lambda_U = 0.03402$. Then, according to the analysis performed in §3.5, we can expect the new method to be moderately faster than Algorithm A for p close to 1. Table 2 gives $\mathrm{IAVCD}(t, p)$ for $p = 0.999$, 0.9999 and several values of t varying from 100 to 20,000 h. The table also gives the CPU times in seconds of the new method and Algorithm A for each (t, p) pair. The methods were run with a truncation error target $\varepsilon = 10^{-8}$ and CPU times were measured as for the first large CTMC model. Figure 11 compares measured speedups of the new method over Algorithm A with the speedup measure S. We can note that the new method is, in most cases, moderately faster than Algorithm A. The differences between measured speedups and S can be attributed to the same causes as for the first large CTMC model. Again, for larger CTMC models, S will be a more accurate speedup measure.

5 Conclusions

We have developed a new randomization-based general-purpose method for the computation of the interval availability distribution of systems modeled by CTMCs. The basic idea of the new method

Table 2: $\mathrm{IAVCD}(t, p)$ and CPU times in seconds of the new method (N) and Algorithm A (A) for the CTMC model of the fault-tolerant control system.

t (h)	p	$\mathrm{IAVCD}(t, p)$	CPU time (N)	CPU time (A)
100	0.999	0.99119876	2.260	2.944
100	0.9999	0.99103160	1.432	1.664
200	0.999	0.98281885	4.932	7.152
200	0.9999	0.98217893	3.396	3.104
500	0.999	0.96001411	13.26	24.05
500	0.9999	0.95629808	7.772	9.857
1,000	0.999	0.92876245	35.99	68.06
1,000	0.9999	0.91531894	18.34	24.81
2,000	0.999	0.88544726	97.85	197.2
2,000	0.9999	0.84074059	42.47	63.94
5,000	0.999	0.83650699	408.6	880.4
5,000	0.9999	0.66442446	137.0	222.5
10,000	0.999	0.83797806	1,389	2,914
10,000	0.9999	0.47477429	336.9	646.3
20,000	0.999	0.87505490	5,125	9,849
20,000	0.9999	0.28008887	929.2	1,905

is the use of a randomization construct with different randomization rates for the up and down states. The new method is numerically stable and computes the interval availability distribution with well-controlled truncation error. In addition, for large CTMC models, when the maximum output rates from up and down states are significantly different, and when the interval availability has to be guaranteed to have a level close to one, the new method is significantly or moderately less costly in terms of CPU time than a previous randomization-based state-of-the-art method, depending on whether the maximum output rate from down states is larger than the maximum output rate from up states, or vice versa. The new method can be more costly, but a relatively inexpensive switch for large models of reasonably quality can be easily implemented to choose the fastest method. Along the way, we have shown the correctness of a generalized randomization construct, in which arbitrarily different randomization rates can be associated with different states, for both finite CTMCs with infinitesimal generator and uniformizable CTMCs with denumerable state space. A direction in which this work could be continued is the development of another randomization-based general-purpose method for the computation of the interval availability distribution of systems models by CTMCs that for large CTMC models id less costly interms of CPU time than the previous randomization-based state-of-the-art method.

6 Electronic companion

An electronic companion to this paper is available as part of the online version at `http://dx.doi.org/10.1287/ijoc.1120.0539`.

Acknowledgments

This research work has been supported by the Ministry of Science and Technology of Spain and FEDER ("Fondo Europeo de Desarrollo Regional") under Grant DPI2004-05077, and by the Ministry of Science and Innovation of Spain and FEDER under Grant TIN2008-06735-C02-02.

References

[1] Abramowitz, M., I.A. Stegun. 1970. *Handbook of Mathematical Functions*. Dover, New York.

[2] Barlow, R.E., F. Proschan. 1981. *Statistical Theory of Reliability and Life Testing. Probability Models*. McArdle Press, Silver Spring, MD.

[3] Buchholz, P. 1994. Exact and ordinary lumpability in finite Markov chains. *J. of Appl. Probab.* **31**(1) 59–75.

[4] Carrasco, J.A. 2003. Computation of bounds for transient measures of large rewarded Markov models using regenerative randomization. *Computers & Oper. Res.* **30**(7) 1005–1035.

[5] Carrasco, J.A. 2004a. Solving large interval availability models using a model transformation approach. *Computers & Oper. Res.* **31**(6) 807–861.

[6] Carrasco, J.A. 2004b. Transient analysis of some rewarded Markov models using randomization with quasistationarity detection. *IEEE Trans. on Computers* **53**(9) 1106–1120.

[7] Carrasco, J.A. 2005. Transient analysis of large Markov models with absorbing states using regenerative randomization. *Comm. in Statist.—Simulation and Comput.* **34**(4) 1027–1052.

[8] Carrasco, J.A. 2006. Two methods for computing bounds for the distribution of cumulative reward for large Markov models. *Performance Evaluation* **63**(12) 1165–1195.

[9] Carrasco, J.A. 2011. An efficient and numerically stable method for computing bounds for the interval availability distribution. *INFORMS J. on Computing* **23**(2) 268–283.

[10] Dahlquist, G., Å. Björch. 1974. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, USA.

[11] de Souza e Silva, E., H.R. Gail. 1986. Calculating cumulative operational time distributions of repairable computer systems. *IEEE Trans. on Computers* **35**(4) 322–332.

[12] de Souza e Silva, E., H.R. Gail. 1989. Calculating availability and performability measures of repairable computer systems using randomization. *J. of the ACM* **36**(1) 171–193.

[13] Donatiello, L., V. Grassi. 1991. On evaluating the cumulative performance distribution of fault-tolerant computer systems. *IEEE Trans. on Computers* **40**(11) 1301–1307.

[14] Fox, B.L., P.W. Glynn. 1988. Computing Poisson probabilities. *Comm. of the ACM* **31**(4) 440–445.

[15] Goyal, A., A.N. Tantawi. 1988. A measure of guaranteed availability and its numerical evaluation. *IEEE Trans. on Computers* **37**(1) 25–32.

[16] Grassmann W. 1987. Means and variances of time averages in Markovian environments. *European J. of Oper. Res.* **31**(1) 132–139.

[17] Grassmann, W.K. 1977a. Transient solutions in Markovian queuing systems. *Computers & Oper. Res.* **4**(1) 47–53.

[18] Grassmann, W.K. 1977b. Transient solutions in Markovian queues. An algorithm for finding them and determining their waiting-time distributions. *European J. of Oper. Res.* **1**(6) 396–402.

[19] Grassmann W.K. 1993. Rounding errors in certain algorithms involving Markov chains. *ACM Trans. on Math. Software* **19**(4) 496–508.

[20] Gross, D., D.R. Miller. 1984. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Oper. Res.* **32**(2) 343–361.

[21] Higham, N.J. 2002. *Accuracy and stability of numerical algorithms*. SIAM, Philadelphia.

[22] IEEE. 1985. *IEEE Standard for Binary Floating-Point Arithmetic*. IEEE Computer Society Press, New York{, NY, USA}.

[23] Islam, S.M.R., H.H. Ammar. 1989. Performability of the hypercube. *IEEE Trans. on Reliability* **38**(5) 518–526.

[24] Kijima, M. 1997. *Markov Processes for Stochastic Modeling*. Chapman & Hall, London.

[25] Knüsel, L. 1986. Computation of the chi-square and Poisson distribution. *SIAM J. on Sci. and Statist. Comput.* **7**(3) 1022–1036.

[26] Melamed, B., M. Yadin. 1984. Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes. *Oper. Res.* **32**(4) 926–944.

[27] Nabli, H., B. Sericola. 1996. Performability analysis: a new algorithm. *IEEE Trans. on Computers* **45**(4) 491–494.

[28] Pattipati K.R., Y. Li, H.A. Blom. 1993. A unified framework for the performability evaluation of fault-tolerant computer systems. *IEEE Trans. on Computers* **42**(2) 312–326.

[29] Qureshi, M.A., W.H. Sanders. 1996. A new methodology for calculating distributions of reward accumulated during a finite interval. *IEEE Annual Symposium on Fault-Tolerant Computing, Sendai, Japan, June 1996*. IEEE Computer Society Press, Los Alamitos, CA. 116–125.

[30] Rácz, S., A. Tari, M. Telek. 2002. MRMSolve: distribution estimation of large Markov reward models. *Lecture Notes in Computer Science* **2324** 141–158.

[31] Reibman, A., K. Trivedi. 1988. Numerical transient analysis of Markov models. *Computers & Oper. Res.* **15**(1) 19–36.

[32] Reibman, A., K. Trivedi. 1989. Transient analysis of cumulative measures of Markov model behavior. *Comm. in Statist.—Stochastic Models* **5**(4) 683–710.

[33] Ross, S.M. 1983. *Stochastic Processes*. John Wiley & Sons, New York{, NY, USA}.

[34] Rubino, G., B. Sericola. 1992. Interval availability analysis using operational periods. *Performance Evaluation* **14**(3-4) 257–272.

[35] Rubino, G., B. Sericola. 1993. Interval availability distribution computation. *IEEE International Symposium on Fault-Tolerant Computing, Toulouse, France, June 1993*. IEEE Computer Society Press, Los Alamitos, CA. 48–55.

[36] Rubino, G., B. Sericola. 1995. Interval availability analysis using denumerable Markov processes: application to multiprocessor subject to breakdowns and repair. *IEEE Trans. on Computers* **44**(2) 286–291.

[37] Sericola, B. 1990. Closed form solution for the distribution of the total time spent in a subset of a homogeneous Markov process during a finite observation period. *J. of Appl. Probab.* **27**(2) 713–719.

[38] Sericola, B. 1999. Availability analysis of repairable computer systems and stationarity detection. *IEEE Trans. on Computers* **48**(11) 1166-1172.

[39] Smith, R., K.S. Trivedi, A.V. Ramesh. 1988. Performability analysis: measures, an algorithm, and a case study. *IEEE Trans. on Computers* **37**(4) 406–417.

[40] Suñé, V., J.A. Carrasco. 2005. Efficient implementations of the randomization method with control of the relative error. *Computers & Oper. Res.* **32**(5) 1089–1114.

[41] Suñé, V., J.A. Carrasco, H. Nabli, B. Sericola. 2010. Comment on "Performability analysis: a new algorithm". *IEEE Trans. on Computers* **59**(1) 137–138.

[42] Takács, L. 1957. On certain sojourn time problems in the theory of stochastic processes. *Acta Mathematica Hungarica* **8**(1-2) 169–191.

[43] van Moorsel, A.P.A., W.H. Sanders. 1994. Adaptive uniformization. *Comm. in Statist.—Stochastic Models* **10**(3) 619–647.

[44] van Moorsel, A.P.A., W.H. Sanders. 1997. Transient solution of Markov models by combining adaptive & standard uniformization. *IEEE Trans. on Reliability* **46**(3) 430–440.