Software Quality Journal (2020) 28:173–194
https://doi.org/10.1007/s11219-019-09458-1

A transient interval reliability analysis for software rejuvenation models with phase expansion

Junjun Zheng[1] · **Hiroyuki Okamura**[2] · **Tadashi Dohi**[2]

Published online: 19 July 2019

Abstract

The phenomenon of software aging refers to the continuing degradation of software system performance with the operation time and is usually caused by the aging-related bugs such as the memory leak and the accumulation of round-off errors. Software rejuvenation acts as one of the proactive fault management techniques against the software aging. In this paper, we evaluate the interval reliability for two basic stochastic models with periodic software rejuvenation by Garg et al. (1995) and Suzuki et al. (2002a, b). The interval reliability is one of the most generalized dependability measures that involve commonly used reliability function and steady-state availability as the special cases, and is helpful for the system design during a fixed mission period. From the mathematical point of view, the interval reliability for the software rejuvenation models leads to the transient analysis. We focus on the phase expansion approach for solving the transient solutions for the basic software rejuvenation models. The phase expansion is an approximate technique that replaces arbitrary probability distributions by the phase-type (PH) distributions. Benefiting from the phase expansion, we can numerically derive the transient interval reliability for two software rejuvenation models. In numerical examples, we discuss the accuracy of the phase expansion and also reveal quantitative properties of the interval reliability measures.

Keywords Software rejuvenation · Transient analysis · Interval reliability · Phase expansion · Markov regenerative process · Pointwise availability

This paper is an extension of work originally reported at The 8th International Workshop on Software Aging and Rejuvenation (WoSAR 2016) (Okamura and Dohi 2016b).

✉ Junjun Zheng
jzheng@asl.cs.ritsumei.ac.jp

Hiroyuki Okamura
okamu@hiroshima-u.ac.jp

Tadashi Dohi
dohi@hiroshima-u.ac.jp

[1] Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu 5258577, Japan

[2] Hiroshima University, 1-4-1 Kagamiyama, Higashihiroshima 7398527, Japan

1 Introduction

Aiming at enhancing the dependability and reducing outages of software systems, the studies on software aging and rejuvenation have been extensively taken over the last three decades. The software aging refers to the phenomenon that software system performance degrades continuously with time and system failure occurs suddenly. Although it is known that the software aging is usually caused by some aging-related software bugs, such as the memory leak and the accumulation of round-off errors, remaining in the system, it is difficult to remove such bugs completely in the software development and testing phases, due to their low reproducibility. The software rejuvenation is a proactive software control technique to prevent the system performance degradation and other associated failures related to software aging by refreshing the system condition occasionally or periodically before the system failure, e.g., a hardware reboot. The execution of rejuvenation typically bring an overhead, but it may prevent more severe failures from occurring.

The concept of software aging and rejuvenation was first introduced by Huang et al. (1995). They presented a simple continuous-time Markov chain (CTMC) model with four states (normal, failure-probable, failure, and rejuvenation states) of a software system. Since then, many authors have proposed various software rejuvenation policies for counteracting the software aging. For example, Garg et al. (1995) brought an idea of periodic rejuvenation (i.e., deterministic rejuvenation trigger interval) into Huang et al.'s model. In their assumption, the degraded system is renewed and the system state becomes normal immediately after the repair operation. However, there is the fact that, after the completion of the repair operation, some cleanup and the resume of process execution at the checkpointed state may be necessary before restarting the system; thus, another periodic rejuvenation model based on Garg et al.'s model was proposed by Suzuki et al. (2002a, b). As a result, Garg et al.'s model and Suzuki et al.'s model are regarded as two fundamental software rejuvenation models in software reliability engineering, since they have been widely studied and evaluated over years and can be extended to various complicated rejuvenation models.

To our knowledge, many researches focused on solving steady-state solutions and analyzing steady-state measures for software rejuvenation models such as the steady-state availability (Trivedi et al. 2000; Xie et al. 2005; Bao et al. 2005; Thein and Park 2009; Rezaei and Sharifi 2010; Dohi and Okamura 2016; Zheng et al. 2017). The steady-state measure, in general, is appropriate when the software systems have a very long lifetime. The steady-state availability is one of the typical steady-state measures, indicating the long-run probability when the system operation time tends to infinity. On the other hand, there is the case where one wishes to ensure the system availability for a specific time period. For example, it is important for a typical enterprise system with attendance management to be available during office-going hours, and it is not necessary to work for a whole day. In such a situation, transient measures are more appropriate than the steady-state measures. In fact, Hosford (1960) first proposed the concept of the interval availability and defined it as the fraction of time duration when a system is operational over a finite time horizon. Rubino and Sericola (1992) analyzed the interval availability distribution of repairable computer systems using Markov models.

This paper focuses on the interval reliability for software systems with periodic rejuvenation. The interval reliability is comprehensive and is known as one of the most generalized dependability measures, which involve both system reliability and system availability such as the steady-state availability and the pointwise availability. It assesses an ability of an operable software system over a specific time interval without failure and is helpful for the

system design during a fixed mission time period. However, from the perspective of software aging and rejuvenation, there are a few works concerning the interval reliability, since the formulation of the interval reliability leads to the transient analysis of the software rejuvenation models. The transient analysis is carried out to obtain the state probabilities of the system at an arbitrary time, so that it is a relatively much hard work compared with the steady-state analysis. Consequently, to date, the transient analysis is still a challenging issue when one focuses on the short period for system operation. Several studies by Dohi et al. (2010, 2012, 2018) tried to solve the transient solutions by the usage of Laplace-Stieltjes transform (LST) and its numerical inversion. However, the main problem is that the numerical inversion of LST is not stable, in other words, it is not easy to control the round-off and truncation errors in the numerical inversion of LST.

In this paper, we evaluate the interval reliability for software rejuvenation based on stochastic models through the transient analysis with the phase expansion approach. Concretely, the interval reliability is formulated via the transient analysis of two software rejuvenation models introduced by Garg et al. (1995) and Suzuki et al. (2002a, b). The transient analysis is easily conducted by using the phase expansion, which is a method to approximate an arbitrary general distribution with phase-type (PH) distribution (Okamura and Dohi 2016a). Since the PH distribution is defined by a CTMC with absorbing states, the approximate (phase-expanded) model reduces to the "equivalent" CTMC. It is well-known that the transient analysis of CTMC is computationally feasible via some numerical methods, e.g., uniformization. Based on the phase expansion, this paper also considers two special cases of interval reliability: the reliability function and the pointwise availability. The highlights of this paper are as follows:

- Coverage of the analytical transient solutions for both two fundamental software rejuvenation models in software reliability engineering;
- Formulation and evaluation of the transient behavior in terms of the interval reliability for two rejuvenation models using a phase expansion approach;
- Comparison of the interval reliabilities between two rejuvenation models.

The rest of this paper is organized as follows. In Section 2, we give an overview of the previous works on the analysis of software systems with rejuvenation. In Section 3, we introduce two fundamental software rejuvenation models using Markov regenerative models and their behavioral analysis with LST. Section 4 presents the transient analysis of software rejuvenation models with phase expansion. In Section 5, we formulate the interval reliability of software rejuvenation models in the context of phase-expanded CTMC models. Section 6 is devoted to numerical experiments. Finally, this paper is concluded with some remarks in Section 7.

2 Related works

Huang et al. (1995) first introduced the concept of software aging and rejuvenation in 1995. In Huang et al. (1995), a simple stochastic model for analyzing software rejuvenation in a continuously running telecommunication application was presented. They also expressed the expected downtime and expected cost due to the overhead of rejuvenation, aiming to handle transient software failures. Afterwards, Garg et al. (1995) proposed a periodic time-based rejuvenation policy upon Huang et al.'s model (1995). To deal with the deterministic rejuvenation trigger interval, the dynamics of system behavior were characterized by a

Markov regenerative stochastic Petri net (MRSPN) model. Both models in Huang et al. (1995) and Garg et al. (1995) were extensively analyzed by Dohi et al. (2000a, b, 2001) and Suzuki et al. (2002a). They showed that Huang et al.'s model (1995) and Garg et al.'s model (1995) could be extended to the well-known non-Markovian processes, i.e., the semi-Markov process (SMP) and the Markov regenerative process (MRGP). On the other hand, several papers (Okamura et al. 2001; Vaidyanathan and Trivedi 2005; Bruneo et al. 2013) discussed the workload-based rejuvenation policies where the rejuvenation timing is controlled by the number of processed transactions, namely, the rejuvenation is triggered when the number of processed transactions exceeds a threshold.

Since the frequent software rejuvenation typically involves much overheads, there is a tradeoff on the system performance between the execution frequency of rejuvenation and its associated overhead. In general, low system performance results in low system dependability. For example, the system hanging due to software aging extremely decreases the system dependability such as the system availability. Thus, it is important to determine the optimal rejuvenation policy based on the system performance indices, as well as the system dependability measures. In many papers, the system dependability measures or performance indices are derived by the means of the steady-state analysis of the stochastic models. Thein and Park (2009), Rezaei and Sharifi (2010), and Machida et al. (2013) derived the steady-state availability of a virtualized system with rejuvenation. Trivedi et al. (2000) discussed availability models with rejuvenation to evaluate the effectiveness of proactive fault management in operational software systems and determined the optimal times to perform rejuvenation under different scenarios. Xie et al. (2005) and Bao et al. (2005) described SMP models for degraded software systems with rejuvenation and determined the optimal rejuvenation schedule that maximizes the system availability. Recently, Dohi and Okamura (2016) considered an operational software system with multi-stage degradation levels due to software aging and derived the optimal dynamic software rejuvenation policy that maximizes the steady-state system availability via the semi-Markov decision process. Zheng et al. (2017) comprehensively evaluated six software rejuvenation policies for transaction systems in terms of steady-state performance measures, i.e., loss probability of transactions and the mean response time, under the environment where the arrival stream of system follows a Markovian arrival process.

On the contrary, only a few works focused on the transient measures for the systems considering software aging and rejuvenation, because it is well-known that the transient analysis of the software rejuvenation models is more difficult than the steady-state analysis and requires the state probabilities at an arbitrary time. The computation of state probabilities is not so easy even if the number of states in the model is small. One solution of solving the transient case is to derive the LSTs of state probabilities and take place the numerical inversion of the LSTs by Durbin's algorithm (Durbin 1974; Dubner and Abate 1968). For example, Dohi et al. (2010, 2012) formulated the pointwise availability and steady-state interval reliability of Huang et al.'s model (1995) and other variants by LSTs and their numerical inversion. Recently, Dohi et al. (2018) further examined numerically the transient behavior of the interval reliability of software rejuvenation models at an arbitrary operation time using the inversion of the LSTs. However, since Durbin's algorithm is based on a Fourier series expansion and requires some judicious choices of the parameters in his algorithm for rapid convergence, the biggest disadvantage of Durbin's method is the dependence of the discretization and truncation errors on the free parameters. In other words, by a suitable choice of the parameters, the discretization error becomes arbitrarily small, but the truncation error grows to infinity at the same time and vice versa. Therefore, it should be

pointed out that, the numerical inversion of LST is not stable, namely, it is not easy to control the round-off and truncation errors in the numerical inversion of LST. Alternatively, Okamura and Dohi (2016a) developed the fast PH fitting algorithm from the probability density function (p.d.f.) of general distribution. This enables us to perform the transient analysis more accurately. In fact, Okamura et al. (2014) demonstrated the transient analysis for the rejuvenation model of a virtualized system with phase-expanded CTMC models. Also, in Okamura and Dohi (2016b), they discussed the transient analysis of a basic software rejuvenation model introduced by Garg et al. (1995) and formulated the interval reliability for software rejuvenation.

This paper extends and improves and the work of Okamura and Dohi (2016b) to cover both two fundamental software rejuvenation models in software reliability engineering. Apart from Garg et al.'s model, this paper also formulates and evaluates the transient behavior of the interval reliability of another fundamental software rejuvenation model introduced by Suzuki et al. (2002a, b) using the phase expansion, and compare the reliability measures between two basic rejuvenation models.

3 Software rejuvenation models

3.1 Model description

3.1.1 Model-I

Consider the stochastic model with periodic software rejuvenation (Model-I, see Fig. 1) similar to Garg et al. (1995). We suppose three degradation levels: highly robust state, failure probable state, and degradation failure state. Note that the degradation failure is regarded as the system failure, e.g., a sudden crash of software application. All system states are defined as follows:

State 0: highly robust state (normal operation state)
State 1: failure probable state (degraded state)
State 2: software rejuvenation state from failure probable state
State 3: failure state
State 4: software rejuvenation state from highly robust state.

Without loss of generality, the software system is supposed to start at time $t = 0$ at the highly robust state (State 0: normal operation state). From State 0, the system performance degrades with the operation time. Let Z_0 be the random time to reach the failure probable state (State 1: degraded state) from State 0 with the cumulative distribution function (c.d.f.) $F_0(t) = P(Z_0 \leq t)$. In the degraded state, the system may fail with a positive probability. Let X be the time to failure from State 1 having the c.d.f. $F_f(t) = P(X \leq t)$. Whenever the system failure occurs, the system state enters State 3 where the system is forced to undergo the repair operation immediately. Let Y denote the time to complete the repair operation with the c.d.f. $F_a(t) = P(Y \leq t)$. After the completion of repair operation, the system state is assumed to be as good as new and returns back to State 0 from State 3.

From the aspect of prevention of the system failure, the rejuvenation of system may be performed at a random time interval measured from the start of the software in State 0. Thus, we define $F_r(t)$ as the c.d.f. of the time to execute the software rejuvenation and $F_c(t)$ as the c.d.f. of the time to complete the software rejuvenation. After the completion of the software rejuvenation, the software system also becomes as good as new and goes back to State 0

from State 2 (4). Generally speaking, the age of software system is renewed when the state becomes State 0 again due to either the repair operation or the rejuvenation, and the time clock of rejuvenation trigger is also reset when the system enters the highly robust state. We call the above state transition process as an MRGP (Kulkarni 1995) since the time clock of rejuvenation trigger does not reset when the system degrades to the failure probable state.

In the MRGP of Model-I illustrated in Fig. 1, the regeneration points and non-regeneration point are distinguished by circles (0, 2, 3, 4) and square (1), respectively. Model-I shows a very basic stochastic model with periodic software rejuvenation, which is essentially the same as that in Garg et al. (1995). Garg et al. (1995) numerically presented this model using an MRSPN.

3.1.2 Model-II

Consider a different model (Model-II, see Fig. 2) from Model-I introduced in Suzuki et al. (2002a, b). In Model-I, it is assumed that the system is renewed and the state moves to State 0 immediately after the completion of the repair operation. However, after the completion of the repair operation, some cleanup and the resume of process execution at the checkpointed state may be necessary before restarting the system; thus, in Model-II, an additional software rejuvenation after completing the repair operation is adopted to renew the system.

Figure 2 shows the state transition diagram of Model-II. The only difference between Model-I and Model-II is that the system is rejuvenated just after a system failure recovery in Model-II.

3.2 Behavioral analysis

For both Model-I and Model-II, Dohi et al. (2010) presented the behavioral analysis in terms of LST. Let $K_{ij}(t)$ be the probability that the system is in State j by time t when the state

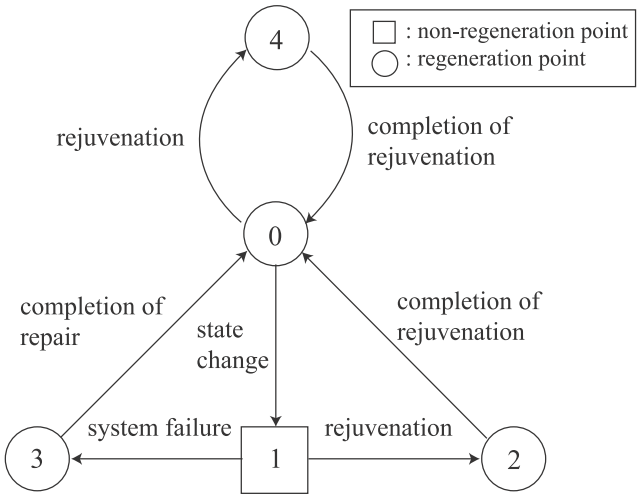


Fig. 1 State transition diagram (Model-I)

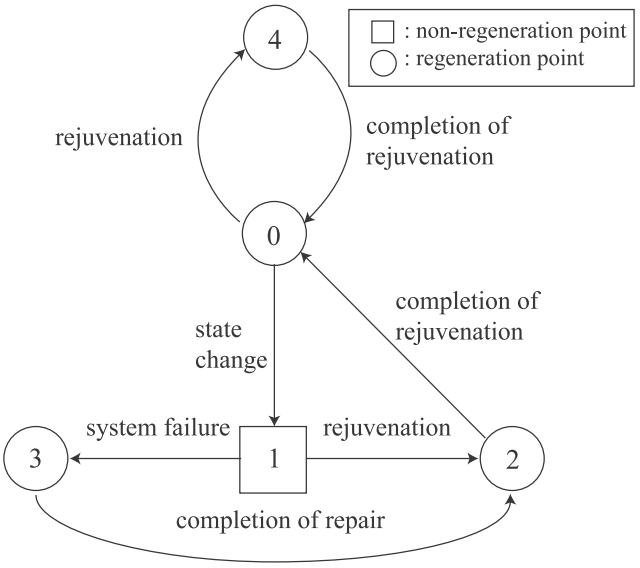


Fig. 2 State transition diagram (Model-II)

of system begins with regenerative State i at $t = 0$. Also let $q_{ij}(s)$ be the LST of $K_{ij}(t)$, i.e.,

$$q_{ij}(s) = \int_0^\infty e^{-st} dK_{ij}(t). \tag{1}$$

Further, let $K_{ij}^{(k)}(t)$ and $q_{ij}^{(k)}(t)$ be the probability that the system enters State j by time t when starting from State i via a non-regeneration State k and its LST, respectively.

3.2.1 Model-I

For the Model-I, we obtain

$$q_{01}(s) = \int_0^\infty e^{-st} \overline{F}_r(t) dF_0(t), \tag{2}$$

$$q_{02}^{(1)}(s) = \int_0^\infty e^{-st} (F_0 * \overline{F}_f)(t) dF_r(t), \tag{3}$$

$$q_{03}^{(1)}(s) = \int_0^\infty e^{-st} \overline{F}_r(t) d(F_0 * F_f)(t), \tag{4}$$

$$q_{04}(s) = \int_0^\infty e^{-st} \overline{F}_0(t) dF_r(t), \tag{5}$$

$$q_{20}(s) = \int_0^\infty e^{-st} dF_c(t), \tag{6}$$

$$q_{30}(s) = \int_0^\infty e^{-st} dF_a(t), \tag{7}$$

$$q_{40}(s) = \int_0^\infty e^{-st} dF_c(t), \tag{8}$$

where $(F_0 * \overline{F}_f)(t)$ and $(F_0 * F_f)(t)$ are Stieltjes convolution functions which are given by

$$(F_0 * \overline{F}_f)(t) = \int_0^t \overline{F}_f(t-u)dF_0(u), \tag{9}$$

$$(F_0 * F_f)(t) = \int_0^t F_f(t-u)dF_0(u). \tag{10}$$

Defining $H_{00}(t)$ as the recurrence time distribution from State 0 to State 0, we then obtain its LST

$$\begin{aligned} h_{00}(s) &= \int_0^\infty e^{-st} dH_{00}(t) \\ &= q_{02}^{(1)}(s)q_{20}(s) + q_{03}^{(1)}(s)q_{30}(s) + q_{04}(s)q_{40}(s). \end{aligned} \tag{11}$$

The objective of our concern is to derive the transient probability of staying in State j at an arbitrary time, provided that the initial state at time $t = 0$ is State 0. Let $P_{0j}(t)$ denote the transient probability from State 0 to State j, then, its LST is given by $p_{0j}(s) = \int_0^\infty e^{-st} dP_{0j}(t)$. After some manipulations, we have

$$p_{00}(s) = \frac{1 - q_{01}(s) - q_{04}(s)}{1 - h_{00}(s)}, \tag{12}$$

$$p_{01}(s) = \frac{q_{01}(s) - q_{02}^{(1)}(s) - q_{03}^{(1)}(s)}{1 - h_{00}(s)}, \tag{13}$$

$$p_{02}(s) = \frac{q_{02}^{(1)}(s)(1 - q_{20}(s))}{1 - h_{00}(s)}, \tag{14}$$

$$p_{03}(s) = \frac{q_{03}^{(1)}(s)(1 - q_{30}(s))}{1 - h_{00}(s)}, \tag{15}$$

$$p_{04}(s) = \frac{q_{04}(s)(1 - q_{40}(s))}{1 - h_{00}(s)}. \tag{16}$$

3.2.2 Model-II

Similarly, for the Model-II, the LSTs of the probabilities that the system becomes State j by time t from State i are

$$q_{01}(s) = \int_0^\infty e^{-st} \overline{F}_r(t) dF_0(t), \tag{17}$$

$$q_{02}^{(1)}(s) = \int_0^\infty e^{-st} (F_0 * \overline{F}_f)(t) dF_r(t), \tag{18}$$

$$q_{03}^{(1)}(s) = \int_0^\infty e^{-st} \overline{F}_r(t) d(F_0 * F_f)(t), \tag{19}$$

$$q_{04}(s) = \int_0^\infty e^{-st} \overline{F}_0(t) dF_r(t), \tag{20}$$

$$q_{20}(s) = \int_0^\infty e^{-st} dF_c(t), \tag{21}$$

$$q_{32}(s) = \int_0^\infty e^{-st} dF_a(t), \tag{22}$$

$$q_{40}(s) = \int_0^\infty e^{-st} dF_c(t). \tag{23}$$

Then, we have the LST of the recurrence time distribution under Model-II

$$\begin{aligned} h_{00}(s) &= \int_0^\infty e^{-st} dH_{00}(t) \\ &= \left(q_{02}^{(1)}(s) + q_{03}^{(1)}(s) q_{32}(s)\right) q_{20}(s) + q_{04}(s) q_{40}(s), \end{aligned} \tag{24}$$

and the LSTs of the transient probabilities beginning from State 0

$$p_{00}(s) = \frac{1 - q_{01}(s) - q_{04}(s)}{1 - h_{00}(s)}, \tag{25}$$

$$p_{01}(s) = \frac{q_{01}(s) - q_{02}^{(1)}(s) - q_{03}^{(1)}(s)}{1 - h_{00}(s)}, \tag{26}$$

$$p_{02}(s) = \frac{q_{02}^{(1)}(s)(1 - q_{20}(s))}{1 - h_{00}(s)}, \tag{27}$$

$$p_{03}(s) = \frac{q_{03}^{(1)}(s)(1 - q_{32}(s) q_{20}(s))}{1 - h_{00}(s)}, \tag{28}$$

$$p_{04}(s) = \frac{q_{04}(s)(1 - q_{40}(s))}{1 - h_{00}(s)}. \tag{29}$$

Dohi et al. (2010, 2018) discussed the transient analysis based on the above LSTs by applying Durbin's algorithm (Durbin 1974; Dubner and Abate 1968) to solve the inversion of LSTs. But unfortunately, there are several problems about LST and its inversion in practice (Okamura and Dohi 2016b). At first, the LST is not always expressed in a closed form using elementary functions. For instance, the LST of Weibull distribution includes the special function such as gamma function. Therefore, even if we know the LSTs of transient probabilities, it is practically impossible to solve the inversion of LSTs analytically. On the other hand, it is known that the numerical methods for the inversion of LST or equivalently the inversion of Laplace transform are not stable, i.e., they are sensitive to round-off errors. For example, Durbin's algorithm needs some judicious choices of the parameters in the algorithm for rapid convergence, which may bring large truncation error easily.

4 Transient analysis with phase expansion

Here, we apply the phase expansion to seek the transient solutions of rejuvenation models. The phase expansion, also called PH approximation, is an approximation technique to replace general distributions (i.e., non-exponential distributions) by PH distributions (Okamura and Dohi 2016a). The PH distribution is defined by the probability distribution of the absorbing time in a CTMC with absorbing states. Since the PH distribution is dense for any probability distribution defined on positive domain (Asmussen and Koole 1993), the PH distribution can approximate any probability distribution with high precision. In this way, the original non-exponential model such as MRGP can be approximated by an phase-expanded CTMC. In other words, by solving the "equivalent" CTMC, we can obtain the solution of the original non-exponential model. There are, in general, several stable methods to compute

the transient solution of phase-expanded CTMC such as uniformization, moment match, least square method of the difference of p.d.f.s, and maximum likelihood estimation, so we can apply them to solve the approximation pattern. Thus, the transient solutions of MRGPs of software rejuvenation can be obtained through solving an "equivalent" CTMC.

Consider a phase-expanded CTMC. Without loss of generality, the infinitesimal generator $\boldsymbol{Q}$ of the CTMC is assumed to be partitioned as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{T} & \boldsymbol{\tau} \\ \mathbf{0} & 0 \end{pmatrix}, \tag{30}$$

where $\boldsymbol{T}$ is an infinitesimal generator of the underlying CTMC over transient states (non-absorbing states) and $\boldsymbol{\tau}$ is a column vector whose elements represent the transition rates from transient states to the absorbing ones. In general, the p.d.f. and c.d.f. of PH distribution are represented by

$$f_{PH}(t) = \boldsymbol{\alpha} \exp(\boldsymbol{T}t)\boldsymbol{\tau}, \quad F_{PH}(t) = 1 - \boldsymbol{\alpha} \exp(\boldsymbol{T}t)\mathbf{1}, \tag{31}$$

where $\boldsymbol{\alpha}$ is the probability (row) vector that determines the initial state of the underlying CTMC, and $\mathbf{1}$ is a column vector whose elements are 1. Note that $\boldsymbol{\xi} = -\boldsymbol{T}\mathbf{1}$ from the property of CTMC. The transient states of the underlying CTMC are called phases. The accuracy of approximation depends on the number of phases.

For the stochastic models with periodic software rejuvenation (Model-I and Model-II) in Section 3, the probability distributions $F_0(t)$, $F_f(t)$, $F_a(t)$, $F_r(t)$, and $F_c(t)$ are approximated by the following PH distributions:

$$F_0(t) \approx 1 - \boldsymbol{\alpha}_0 \exp(\boldsymbol{T}_0 t)\mathbf{1}_0, \tag{32}$$
$$F_f(t) \approx 1 - \boldsymbol{\alpha}_f \exp(\boldsymbol{T}_f t)\mathbf{1}_f, \tag{33}$$
$$F_a(t) \approx 1 - \boldsymbol{\alpha}_a \exp(\boldsymbol{T}_a t)\mathbf{1}_a, \tag{34}$$
$$F_r(t) \approx 1 - \boldsymbol{\alpha}_r \exp(\boldsymbol{T}_r t)\mathbf{1}_r, \tag{35}$$
$$F_c(t) \approx 1 - \boldsymbol{\alpha}_c \exp(\boldsymbol{T}_c t)\mathbf{1}_c, \tag{36}$$

where $\mathbf{1}_0$, $\mathbf{1}_f$, $\mathbf{1}_a$, $\mathbf{1}_r$, and $\mathbf{1}_c$ are the 1's column vectors, whose sizes are determined by the numbers of phases of the respective PH distributions. Thus, we obtain the following approximate CTMCs for the rejuvenation models via using PH distributions.

4.1 Model-I

The Model-I is approximated by the CTMC with the following infinitesimal generator

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{T}_0 \oplus \boldsymbol{T}_r & (\boldsymbol{\tau}_0\boldsymbol{\alpha}_f) \otimes \boldsymbol{I}_{r,r} & \mathbf{1}_0 \otimes (\boldsymbol{\tau}_r\boldsymbol{\alpha}_c) & \\ & \boldsymbol{T}_f \oplus \boldsymbol{T}_r & \mathbf{1}_f \otimes (\boldsymbol{\tau}_r\boldsymbol{\alpha}_c) & (\boldsymbol{\tau}_f\boldsymbol{\alpha}_a) \otimes \mathbf{1}_r \\ \boldsymbol{\tau}_c(\boldsymbol{\alpha}_0 \otimes \boldsymbol{\alpha}_r) & & \boldsymbol{T}_c & \\ \boldsymbol{\tau}_a(\boldsymbol{\alpha}_0 \otimes \boldsymbol{\alpha}_r) & & & \boldsymbol{T}_a \end{pmatrix} \tag{37}$$

where

$$\boldsymbol{\tau}_0 = -\boldsymbol{T}_0\mathbf{1}_0, \quad \boldsymbol{\tau}_f = -\boldsymbol{T}_f\mathbf{1}_f, \quad \boldsymbol{\tau}_a = -\boldsymbol{T}_a\mathbf{1}_a, \tag{38}$$
$$\boldsymbol{\tau}_r = -\boldsymbol{T}_r\mathbf{1}_r, \quad \boldsymbol{\tau}_c = -\boldsymbol{T}_c\mathbf{1}_c. \tag{39}$$

Also, $\boldsymbol{I}_{r,r}$ is the identity matrix having the same dimension as $\boldsymbol{T}_r$. Moreover, $\otimes$ and $\oplus$ are operators of Kronecker product and sum, respectively. Each block of $\boldsymbol{Q}$ corresponds to State 0, State 1, State 2, or State 3. Note that State 4 is the same as State 2, since they both refer

to software rejuvenation state. When the system starts from State 0, the initial probability vector for the above CTMC is given by

$$\boldsymbol{\pi}_0 = \left(\boldsymbol{\alpha}_0 \otimes \boldsymbol{\alpha}_r \ \ \mathbf{0} \ \ \mathbf{0} \ \ \mathbf{0} \right). \tag{40}$$

From the argument of CTMC, the transient probability vector of the approximate CTMC at time t is computed by

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}_0 \exp(\boldsymbol{Q}t) \tag{41}$$

and then, the transient probabilities are approximately obtained by

$$\begin{aligned} &\left(P_{00}(t) \ \ P_{01}(t) \ \ P_{02}(t) \ \ P_{03}(t) \right) \\ &= \boldsymbol{\pi}(t) \begin{pmatrix} \mathbf{1}_0 \otimes \mathbf{1}_r & & & \\ & \mathbf{1}_f \otimes \mathbf{1}_r & & \\ & & \mathbf{1}_c & \\ & & & \mathbf{1}_a \end{pmatrix}. \end{aligned} \tag{42}$$

4.2 Model-II

For Model-II, the approximate CTMC is given with the following infinitesimal generator

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{T}_0 \oplus \boldsymbol{T}_r & (\boldsymbol{\tau}_0 \boldsymbol{\alpha}_f) \otimes \boldsymbol{I}_{r,r} & \mathbf{1}_0 \otimes (\boldsymbol{\tau}_r \boldsymbol{\alpha}_c) & \\ & \boldsymbol{T}_f \oplus \boldsymbol{T}_r & \mathbf{1}_f \otimes (\boldsymbol{\tau}_r \boldsymbol{\alpha}_c) & (\boldsymbol{\tau}_f \boldsymbol{\alpha}_a) \otimes \mathbf{1}_r \\ \boldsymbol{\tau}_c(\boldsymbol{\alpha}_0 \otimes \boldsymbol{\alpha}_r) & & \boldsymbol{T}_c & \\ & & \boldsymbol{\tau}_a \boldsymbol{\alpha}_c & \boldsymbol{T}_a \end{pmatrix}. \tag{43}$$

Similar to Model-I, the transient probabilities under Model-II can be obtained by using (40) through (42).

After obtaining the above approximate CTMCs for Model-I and Model-II, the most important problem is to determine the PH parameters so that the PH distribution can approximate the original distribution well. Okamura and Dohi (2016b) discussed two approaches for the estimation of PH parameters; moment-based (Osogami and Harchol-Balter 2006; Bobbio et al. 2005) and likelihood-based approaches (Okamura et al. 2011). In this paper, the PH parameters are estimated by the likelihood-based approach. Thus, the analytical transient solutions of the software rejuvenation models can be achieved through the analysis of the "equivalent" CTMCs.

5 Formulation of interval reliability

The interval reliability is, generally speaking, a generalized dependability measure derived from the transient analysis. More specifically, the interval reliability is formally defined as the probability that the system is operational throughout $[t, t+x]$, where t and x are called operation time and planning time, respectively. The interval reliability is comprehensive and embraces two other significant dependability measures as special cases, that is, it can be reduced to the pointwise availability at operation time t as $x \to 0$ and the (conditional) reliability at the planning time x given that the system has started in the working state at time $t = 0$. The interval reliability analysis was originally introduced by Barlow and Proschan (1965). In Dohi et al. (2018), a special case of the interval reliability (i.e., limiting interval reliability) for rejuvenation model was discussed.

Suppose that the software rejuvenation is not performed during the time interval $[t, t+x]$, even if the rejuvenation time has already come. Letting $M(t)$ be the expected number of

visits to State 0 during the time interval $(0, t]$, after the system state starts from State 0 at time $t = 0$, we have the LST of $M(t)$

$$m(s) = \frac{h_{00}(s)}{1 + h_{00}(s)}. \tag{44}$$

The interval reliability $R(t, x)$ is defined as the probability that at a specified operation time t, the software system is operating and will continue operating for an interval of $[t, t + x]$ (Barlow and Proschan 1965). The interval reliability is simply called reliability at time x when the operation time $t = 0$, and further becomes pointwise availability at time t as $x \to 0$. The interval reliability is given by

$$R(x, t) = F_{UP}(t) + \int_0^t F_{UP}(t - u) dM(u), \tag{45}$$

where $F_{UP}(t)$ gives the probability that the system continues operating during the time interval $(0, t + x]$ provided that the software rejuvenation is not performed before time t;

$$F_{UP}(t) = (1 - (F_0 * F_f)(t + x))(1 - F_r(t)). \tag{46}$$

In Dohi et al. (2018), the LST of (45) was derived. However, as mentioned before, the inversion of LST is not easy in practice.

Next we consider the interval reliability under the phase expansion. According to the definition of interval reliability, we define the following matrices:

$$\boldsymbol{Q}_{UP} = \begin{pmatrix} \boldsymbol{T}_0 & \boldsymbol{\tau}_0 \boldsymbol{\alpha}_f & & \\ & \boldsymbol{T}_f & & \\ & & \boldsymbol{O}_{c,c} & \\ & & & \boldsymbol{O}_{a,a} \end{pmatrix}, \tag{47}$$

where $\boldsymbol{O}$ means a zero matrix. The above matrix is the infinitesimal generator without rejuvenation. Then, the interval reliability is written by

$$R(x, t) = \boldsymbol{\pi}_0 \exp(\boldsymbol{Q}t) \boldsymbol{U} \exp(\boldsymbol{Q}_{UP} x) \mathbf{1}_{UP}, \tag{48}$$

where

$$\mathbf{1}_{UP} = \begin{pmatrix} \mathbf{1}_0 \\ \mathbf{1}_f \\ \mathbf{0}_c \\ \mathbf{0}_a \end{pmatrix}, \tag{49}$$

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{I}_{0,0} \otimes \mathbf{1}_r & & & \\ & \boldsymbol{I}_{f,f} \otimes \mathbf{1}_r & & \\ & & \boldsymbol{O}_{c,c} & \\ & & & \boldsymbol{O}_{a,a} \end{pmatrix}. \tag{50}$$

As special cases, $R(x, 0)$ and $R(0, t)$ are reduced to the reliability function and the pointwise availability, respectively, that is

$$R(x, 0) = \boldsymbol{\pi}_0 \boldsymbol{U} \exp(\boldsymbol{Q}_{UP} x) \mathbf{1}_{UP}, \tag{51}$$

$$R(0, t) = \boldsymbol{\pi}_0 \exp(\boldsymbol{Q}t) \boldsymbol{U} \mathbf{1}_{UP}. \tag{52}$$

Since the approximate CTMC is ergodic, the limiting interval reliability $R(x, \infty) = \lim_{t \to \infty} R(x, t)$ is simply given by

$$R(x, \infty) = \boldsymbol{\pi}_s \boldsymbol{U} \exp(\boldsymbol{Q}_{UP} x) \mathbf{1}_{UP}, \tag{53}$$

where $\boldsymbol{\pi}_s$ is the steady-state probability vector satisfying

$$\boldsymbol{\pi}_s \boldsymbol{Q} = \mathbf{0}, \quad \boldsymbol{\pi}_s \mathbf{1} = 1. \tag{54}$$

In this paper, we take account into the above three special cases of the interval reliability, covering different desired system dependability properties under specific operation conditions. For example, when one wishes to ensure the system dependability for a specific time period, the interval reliability and reliability function are good candidates to be considered. On the contrary, for systems with long operational lifetimes, the limiting interval reliability is more preferred.

6 Numerical illustration

This section consists of two illustrative experiments. One is conducted to examine the accuracy of the phase expansion from the viewpoints of distribution curves and the interval reliabilities, including the approximate pointwise availabilities, reliability functions, and limiting interval reliabilities, of Model-I. Another is to compare the interval reliabilities between two software rejuvenation models. All model parameters are set empirically in our current experiments, aiming mainly to examine the accuracy of the phase expansion and execute comparison between two rejuvenation models.

6.1 Accuracy of phase expansion

We present an example to analyze quantitatively the transient measures of the software rejuvenation model, Model-I, with the phase expansion. To examine the accuracy of phase expansion, we assume that the exact steady-state availability, reliability function, and limiting interval reliability function are known. In such a case, the probability distributions of state transitions in Fig. 1 are given in Table 1. From this table, we see that the distributions are assumed to be Weibull and lognormal distributions, that means, the transient analysis is difficult by using common methods, e.g., the LST and its numerical inversion approach. The columns of Mean and CV represent the mean time and the coefficient of variation, respectively.

Each distribution is approximated by a PH distribution using the likelihood-based approach (Okamura and Dohi 2015; Okamura et al. 2011). We consider three cases on the numbers of phases; PH1 (1 phase), PH10 (10 phases), and PH100 (100 phases) to examine the accuracy of PH approximation. It should be noted that when the number of phases is 1, the corresponding PH distribution becomes the exponential distribution as a special case. Figure 3 shows the p.d.f.s of approximate PH distributions for the distributions $F_0(t)$, $F_f(t)$, $F_r(t)$, $F_c(t)$, and $F_a(t)$. In the figure, the original p.d.f.'s are drawn as dotted lines.

Table 1 Probability distributions of state transitions

Notation	Transition	Distribution	Mean (hrs.)	CV
$F_0(t)$	State 0 to State 1	Weibull	240	0.5
$F_f(t)$	State 1 to State 3	Weibull	2160	0.5
$F_r(t)$	State 0 or State 1 to State 2	Lognormal	300	0.1
$F_c(t)$	State 2 (State 4) to State 0	Lognormal	1/6	0.2
$F_a(t)$	State 3 to State 0	Lognormal	1/3	0.2

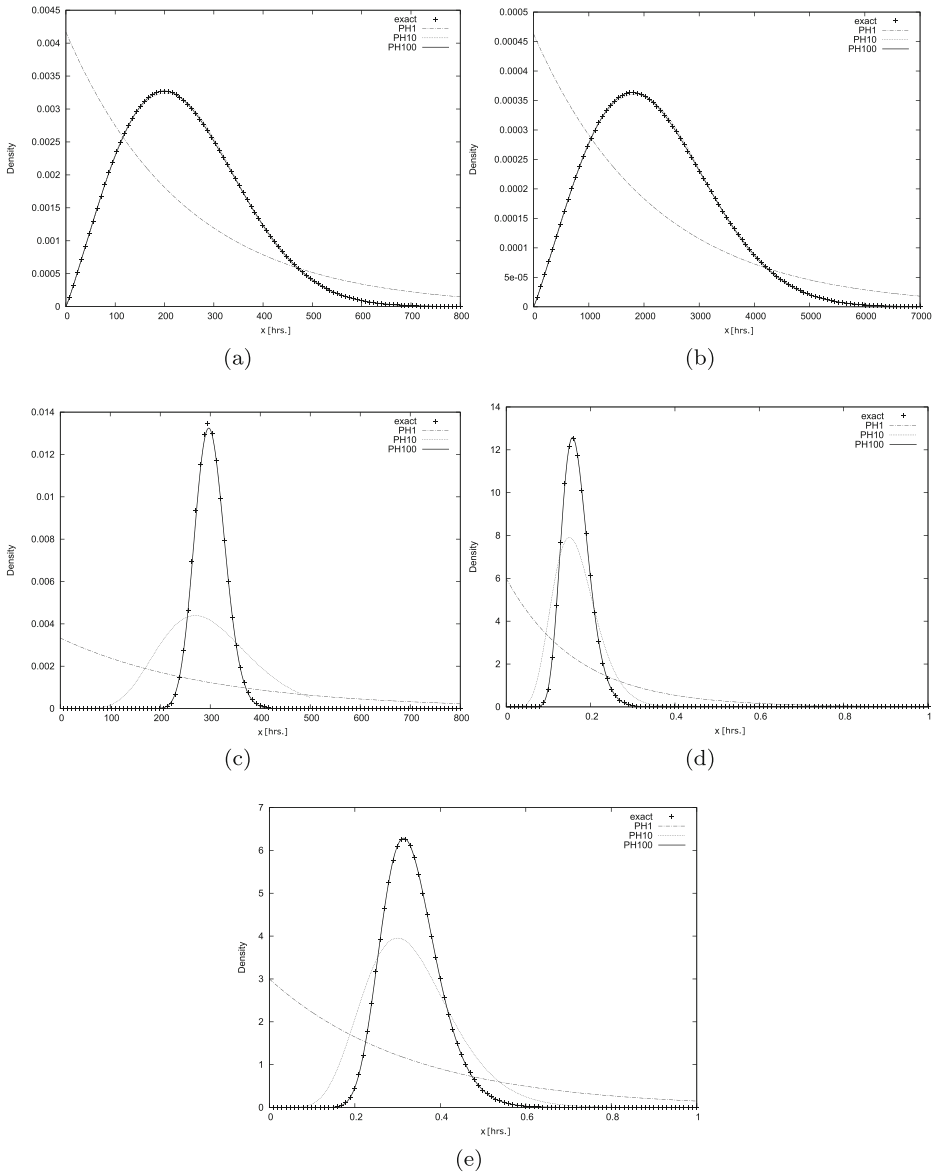


Fig. 3 Approximate PH distributions. **a** $F_0(t)$. **b** $F_f(t)$. **c** $F_r(t)$. **d** $F_c(t)$. **e** $F_a(t)$

From these figures, we can find that the PH distribution gets close to the original distribution when the number of phases increases. Also, in the case where CV is small, the large number of phases is needed to approximate the original distribution accurately. For example, the PH distributions with PH10 are accurate enough to approximate the distributions $F_0(t)$ and $F_f(t)$ with CV=0.5 (see (a) and (b) in Fig. 3), whereas the PH distributions with PH100 are needed for the distributions $F_r(t)$ with CV=0.1, $F_c(t)$ and $F_a(t)$ with CV=0.2 (see (c), (d), and (e) in Fig. 3).

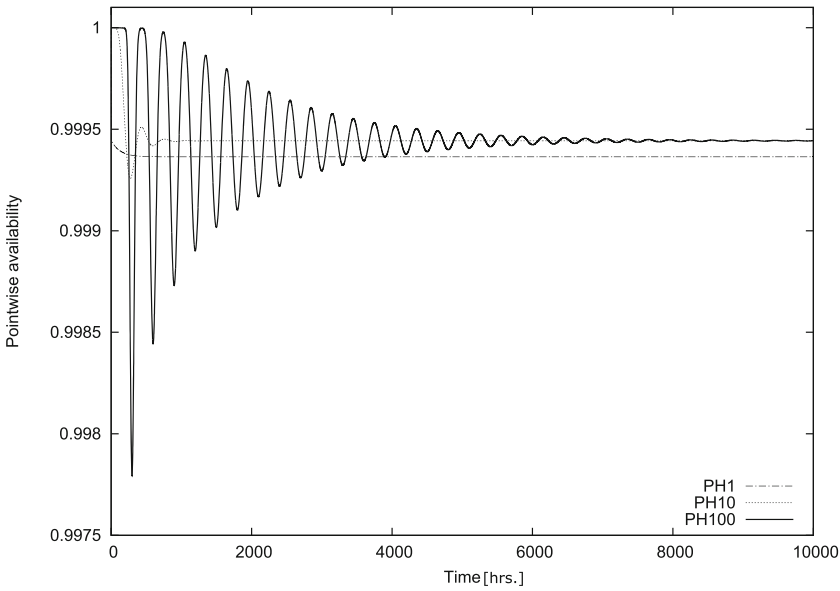


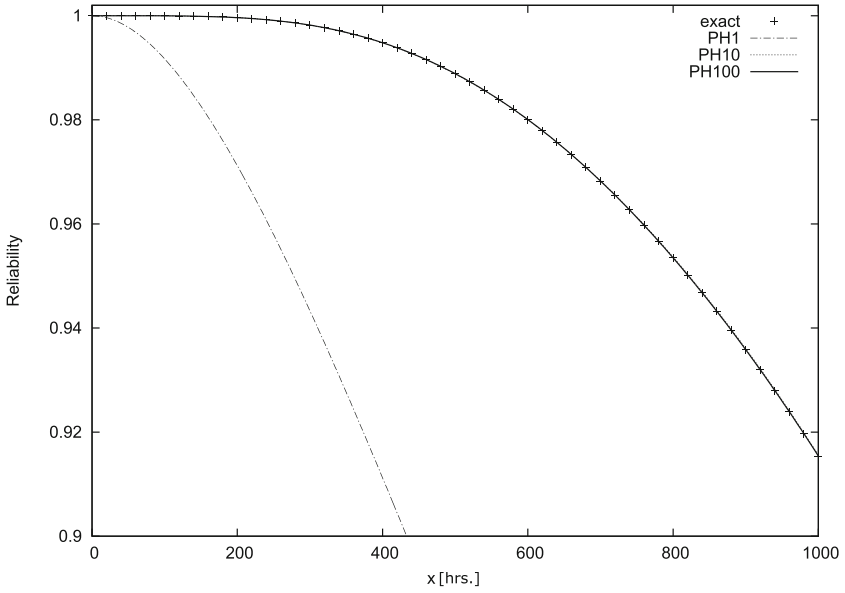
Fig. 4 Approximate pointwise availabilities of Model-I

We evaluate the approximate pointwise availabilities of Model-I. Figure 4 illustrates the approximate pointwise availabilities of Model-I with the phase expansion. The behavior of these availabilities depends on the number of phases. PH1 and PH10 are not enough to approximate accurately the behavior of pointwise availability. In particular, PH1 quickly converges to the steady state, compared with the others. Also, Table 2 shows the exact steady-state availability and those with PH1, PH10, and PH100 under Model-I. Dissimilar to the pointwise availability, the results of PH1 and PH10 get close to the exact steady-state availability, though they are not good to approximate the pointwise availability. Consequently, PH100 is the most accurate to capture the behavior of pointwise availability, as well as the steady-state availability. However, when focusing on only the steady state, PH10 is a more good choice due to lower computational cost, compared with PH100. Moreover, from Fig. 4, in the case of PH100, the pointwise availability reaches its minimum at $t = 297.01$ hours with $R(0, 297.01) = 0.99779$. Also, oscillations can be observed in this figure. A possible explanation is that at the beginning, no rejuvenation actions are performed and the availability first decreases due to failures. A first rejuvenation action takes place at a time with mean 300 h, which brings an impact on pointwise availability. That is, the availability increases roughly from time 297.01 up to time 436.52 h. After a while, this first rejuvenation action has no more impact and the availability starts again to decrease. However, the

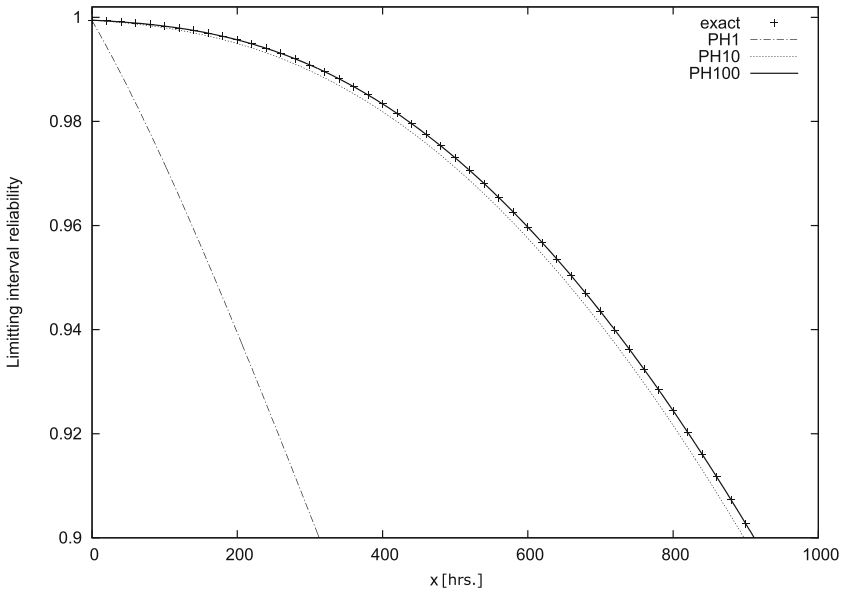
Table 2 Steady-state availabilities of Model-I with phase expansion

Method	Steady-state availability
Exact	0.99944
PH1	0.99936
PH10	0.99944
PH100	0.99944

time to the second rejuvenation action becomes shorter than for the first one, because the first one is not perfect, so that the availability does not decrease much than before. This leads to damped oscillations that can be seen in Fig. 4. Note that the pointwise availability



(a)



(b)

Fig. 5 Approximate reliability functions of Model-I. **a** Approximate reliability functions. **b** Approximate limiting interval reliabilities

Table 3 Probability distributions of state transitions

Notation	Transition	Distribution	Mean (hrs.)	CV
$F_0(t)$	State 0 to State 1	Weibull	600	0.5
$F_f(t)$	State 1 to State 3	Weibull	2160	0.5
$F_r(t)$	State 0 or State 1 to State 2	Lognormal	720	0.1
$F_c(t)$	State 2 (State 4) to State 0	Lognormal	5	0.2
$F_a(t)$	State 3 to State 0	Lognormal	10	0.2

eventually stabilizes and tends to 0.99944 as $t \to \infty$, which is regarded as steady-state availability as shown in Table 2.

Next, we investigate the interval reliability. To compare with the exact value, we consider the reliability function $R(x, 0)$ and the limiting interval reliability $R(x, \infty)$. Figure 5a and b show the reliability function $R(x, 0)$ and the limiting interval reliability function $R(x, \infty)$ with respect to x of Model-I, respectively. Note that the exact results are drawn as dotted lines in both figures. From these figures, it is found that except for PH1, the PH approximation (i.e., PH10 or PH100) works well to approximate the reliabilities. In particular, PH10 becomes more accurate, compared with the result of pointwise availability. This is caused by the fact that $F_0(t)$ and $F_f(t)$ are accurately approximated by PH10. The behavior of the reliability is less sensitive to the rejuvenation trigger time distribution $F_r(t)$, because the rejuvenation is not performed in $[t, t + x]$. Hence, the approximation with PH10 was improved.

6.2 Comparison between Model-I and Model-II

We evaluate the interval reliabilities of a Web server under both two software rejuvenation models described in Section 3 by using the phase expansion and compare them between

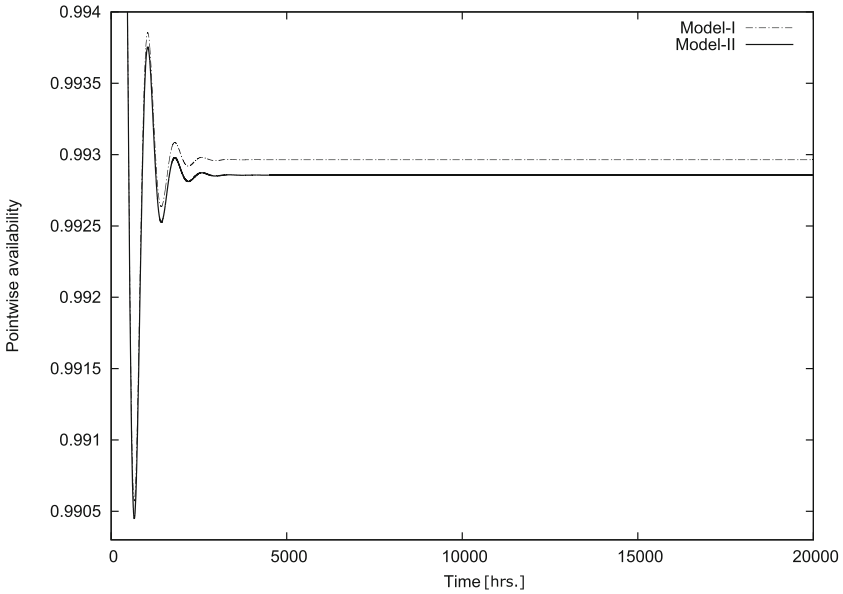


Fig. 6 Approximate pointwise availabilities of Model-I and Model-II (PH10)

Table 4 Steady-state availabilities of Model-I and Model-II

	Steady-state availability
Model-I	0.99305
Model-II	0.99297

two models. Table 3 gives the probability distributions of state transitions. The number of phase is fixed as 10. The reasons are given as follows: (i) The PH distributions with PH10 are accurate enough to approximate the distributions $F_0(t)$ and $F_f(t)$ with CV=0.5 and (ii) although the PH distributions with PH10 are not accurate enough to approximate the distributions $F_r(t)$, $F_c(t)$, and $F_a(t)$ with small CV, but the behavior of the reliability is less sensitive to the distributions $F_r(t)$, $F_c(t)$, and $F_a(t)$, which are not performed in $[t, t + x]$.

We consider the approximate pointwise availabilities of Model-I and Model-II with PH10, which are illustrated in Fig. 6. From this figure, the availability of either Model-I or Model-II quickly converges to the steady state. The approximate pointwise availability of Model-II is slightly smaller than that of Model-I. This is because under Model-II, a software rejuvenation after the completion of the repair operation will be needed to renew the system, which causes much downtime, compared with Model-I. Also, Model-I has a larger steady-state availability, compared with Model-II (see Table 4).

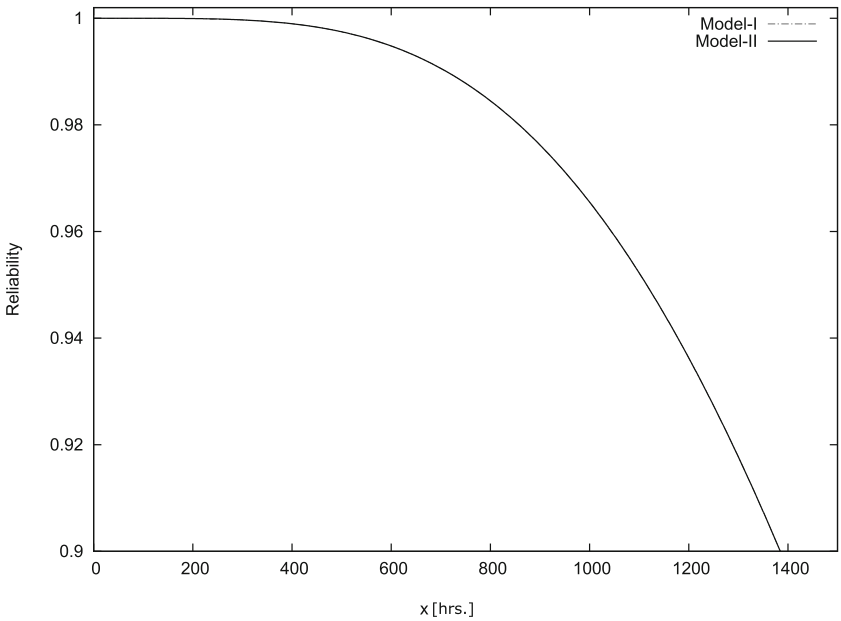
Next, we investigate the interval reliability of two software rejuvenation models. Figure 7 a and b present the approximate reliability functions and the approximate limiting interval reliabilities of Model-I and Model-II with PH10, respectively. It can be observed that all the reliability functions of Model-I and Model-II show the same results. This is due to the fact that when considering the reliability, we focus on only the transitions from State 0 to State 3 or from State 0 to State 2 that are the same in both models (see Figs. 1 and 2). In other words, the repair and rejuvenation operations cannot affect the system reliability significantly.

7 Conclusion and future work

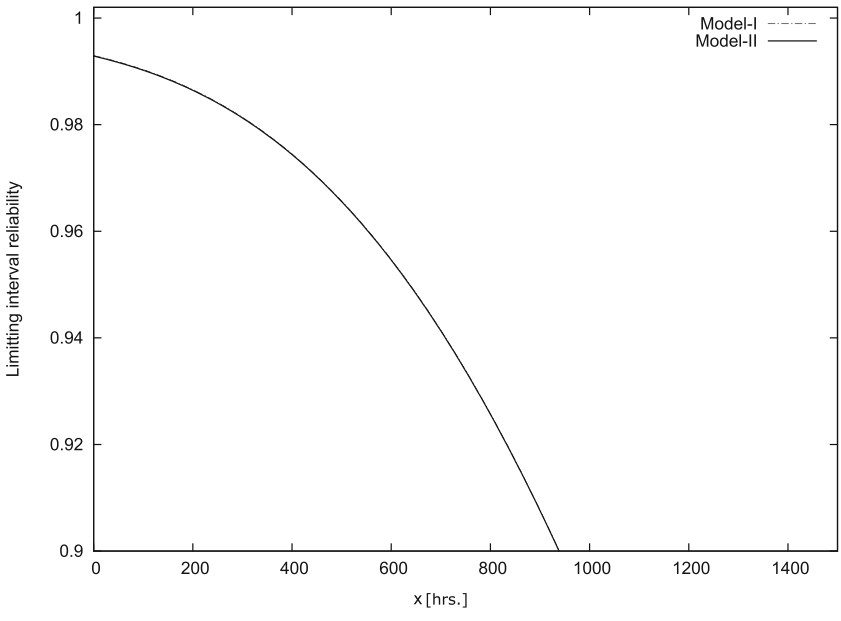
More and more systems focus on software reliability. Software reliability engineering is an important subset of the larger domain of software engineering. In software engineering, software rejuvenation is a proactive software control technique to prevent system performance degradation and other associated failures related to software aging due to aging-related bugs. In this paper, we have focused on analyzing the interval reliability for two fundamental software rejuvenation models in software reliability engineering. It is well-known that the transient analysis of stochastic model is relatively difficult compared with the steady-state analysis. Even if we know the LST representation, the inversion of LST is not efficient from the viewpoint of numerical analysis. Therefore, the key significance of this paper lies on three aspects:

- This paper covers the analytical transient solutions for both two fundamental software rejuvenation models;
- The interval reliability has been formulated through the transient analysis of two software rejuvenation models with phase expansion approximately;
- A comprehensive comparison has been made between two software rejuvenation models based on three interval reliability measures.

In the numerical examples, we have derived the approximate PH distributions from the original distributions and discussed the accuracy of the PH approximation. As a result, the phase



(a)



(b)

Fig. 7 Approximate reliability functions of Model-I and Model-II (PH10). **a** Approximate reliability functions. **b** Approximate limiting interval reliabilities

expansion with the moderate or large number of phases is effective to analyze the transient behavior of rejuvenation model. However, the PH approximation requires the large number of phases when CV of the original distribution is small. Also, according to the results from the accuracy analysis of phase expansion, we have investigated the dependability measures with the interval reliability of a web server under two different software rejuvenation models by using the phase expansion. The lessons learned from the experiments are twofold: 1) the approximate pointwise availability of the model with rejuvenation after the completion of repair operation seems to be slightly smaller than that without rejuvenation after the repair operation and 2) two rejuvenation models have the same reliability functions because the rejuvenation operation cannot affect the system reliability.

In the future, we will combine the discrete PH distribution with the present approach. The discrete PH distribution is effective to approximate the distribution with a small CV such as a constant distribution. Thus, we will try to make a new framework with both continuous and discrete PH distributions. Also, we will consider a practical method to determine the number of phases from the viewpoint of tolerance error.

References

Asmussen, S., & Koole, G. (1993). Marked point processes as limits of Markovian arrival streams. *J. Appl. Prob.*, *30*, 365–372.

Bao, Y., Sun, X., Trivedi, K.S. (2005). A workload-based analysis of software aging, and rejuvenation. *IEEE Trans. Rel.*, *54*(3), 541–548.

Barlow, R.E., & Proschan, F. (1965). *Mathematical theory of reliability*. New York: Wiley.

Bobbio, A., Horváth, A., Telek, M. (2005). Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, *21*(2-3), 303–326.

Bruneo, D., Distefano, S., Longo, F., Puliafito, A. (2013). Workload-based software rejuvenation in cloud systems. *IEEE Trans. Computer*, *62*(6), 1072–1085.

Dohi, T., Goševa-Popstojanova, K., Trivedi, K.S. (2000a). Analysis of software cost models with rejuvenation. In: Proceedings of the 5th International Symposium on High Assurance Systems Engineering (HASE '00), pp. 25-34.

Dohi, T., Goševa-Popstojanova, K., Trivedi, K.S. (2000b). Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule. In: Proceedings of IEEE 2000 pacific rim international symposium on dependable computing (PRDC '00), pp. 77-84.

Dohi, T., Goševa-Popstojanova, K., Trivedi, K.S. (2001). Estimating software rejuvenation schedule in high assurance systems. *Computer J.*, *47*, 473–485.

Dohi, T., & Okamura, H. (2016). Dynamic software availability model with rejuvenation. *Journal of The Operations Research Society of Japan*, *59*(4), 270–290.

Dohi, T., Okamura, H., Osaki, S. (2010). Transient analysis of software availability models with rejuvenation. In: Proceedings of the 4th Asia-pacific international symposium on advanced reliability and maintenance modeling (APARM '10), pp. 169-176.

Dohi, T., Okamura, H., Trivedi, K.S. (2012). Optimizing software rejuvenation policies under interval reliability criteria. In: Proceedings of the 9th IEEE international conference on autonomic and trusted computing (ATC '12), pp. 478-485.

Dohi, T., Zheng, J., Okamura, H., Trivedi, K.S. (2018). Optimal periodic software rejuvenation policies based on interval reliability criteria. *Rel. Eng. Syst. Saf.*, *180*, 463–475.

Dubner, R., & Abate, J. (1968). Numerical inversion of Laplace transform by relating them to the finite Fourier cosine transform. *Journal of the ACM*, *15*, 115–123.

Durbin, F. (1974). Numerical inversion of Laplace transform: an efficient improvement to Dubner and Abate's method. *Computer J.*, *17*(4), 371–376.

Garg, S., Telek, M., Puliafito, A., Trivedi, K.S. (1995). Analysis of software rejuvenation using Markov regenerative stochastic Petri net. In: Proceedings of the 6th international symposium on software reliability engineering (ISSRE '95), pp. 24-27.

Hosford, J.E. (1960). Measures of dependability. *Operations Research*, *8*(1), 53–64.
Huang, Y., Kintala, C., Kolettis, N., Fulton, N.D. (1995). Software rejuvenation: analysis, module and applications. In: Proceedings of the 25th international symposium on fault tolerant computing (FTC '95), pp. 381-390.
Kulkarni, V.G. (1995). *Modeling and analysis of stochastic systems*. New York: Chapman and Hall.
Machida, F., Kim, D.S., Trivedi, K.S. (2013). Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration. *Performance Evaluation*, *70*(3), 212–230.
Okamura, H., & Dohi, T. (2015). Mapfit: An R-based tool for PH/MAP parameter estimation. In Campos, J., & Haverkort, B.R. (Eds.) *Proceedings of the 12th international conference on quantitative evaluation of systems (QEST '15), LNCS 9295* (pp. 105–112): Springer-Verlag.
Okamura, H., & Dohi, T. (2016a). Fitting phase-type distributions and Markovian arrival processes: Algorithms and tools. In Fiondella, L., & Puliafito, A. (Eds.) *Performance and reliability modeling and evaluation* (pp. 49–75): Springer.
Okamura, H., & Dohi, T. (2016b). A phase expansion approach for transient analysis of software rejuvenation model. In: Proceedings of the 8th international workshop on software aging and rejuvenation (WoSAR '16), pp. 98–103.
Okamura, H., Miyahara, S., Dohi, T., Osaki, S. (2001). Performance evaluation of workload-based software rejuvenation scheme. IEICE Trans. *Info. Syst (D)*, *E84-D*(10), 1368–1375.
Okamura, H., Dohi, T., Trivedi, K.S. (2011). A refined EM algorithm for PH distributions. *Performance Evaluation*, *68*(10), 938–954.
Okamura, H., Yamamoto, K., Dohi, T. (2014). Transient analysis of software rejuvenation policies in virtualized system: Phase-type expansion approach. *Quality Technology and Quantitative Management*, *11*(3), 335–352.
Osogami, T., & Harchol-Balter, M. (2006). Closed form solutions for mapping general distributions to minimal PH distributions. *Performance Evaluation*, *63*(6), 524–552.
Rezaei, A., & Sharifi, M. (2010). Rejuvenation high available virtualized systems. In: Proceedings of the 5th international conference on availability, reliability and security (ARES '10), pp. 289–294.
Rubino, G., & Sericola, B. (1992). Interval availability analysis using operational periods. *Performance Evaluation*, *14*, 257–272.
Suzuki, H., Dohi, T., Goševa-Popstojanova, K., Trivedi, K.S. (2002a). Analysis of multistep failure models with periodic software rejuvenation. In Artalejo, J.R., & Krishnamoorthy, A. (Eds.) *Advanced in stochastic modelling. Notable Publications, Inc.* (pp. 85–108).
Suzuki, H., Dohi, T., Okamura, H. (2002b). Cost-effective analysis of software systems with periodic rejuvenation. IEICE Trans. Fundamentals of Electronics. *Communications and Computer Sciences (A)*, *E85-A*(12), 2923–2932.
Thein, T., & Park, J. (2009). Availability analysis of application server using software rejuvenation and virtualization. *J. Computer Science and Technology*, *24*(2), 339–346.
Trivedi, K.S., Vaidyanathan, K., Goševa-Popstojanova, K. (2000). Modeling and analysis of software aging and rejuvenation. In: Proceedings of the 33rd annual on simulation symposium, pp. 270–279.
Vaidyanathan, K., & Trivedi, K.S. (2005). A comprehensive model for software rejuvenation. *IEEE Trans. Dependable and Secure Comput.*, *2*(2), 124–137.
Xie, W., Hong, Y., Trivedi, K.S. (2005). Analysis of a two-level software rejuvenation policy. *Rel. Eng. Syst. Saf.*, *87*(1), 13–22.
Zheng, J., Okamura, H., Li, L., Dohi, T. (2017). A comprehensive evaluation of Software rejuvenation policies for transaction systems with Markovian arrivals. *IEEE Trans. Rel.*, *66*(4), 1157–1177.

Junjun Zheng received the B.S.E. degree in engineering from Fujian Normal University, Fuzhou, China, in 2010, and the M.S. and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 2013 and 2016, respectively.

In 2016 and 2017, he was a Visiting Researcher with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. Since 2018, he has been an Assistant Professor with the Department of Computer Science, Ritsumeikan University, Japan. His research interests include performance evaluation and dependable computing.

Dr. Zheng is a member of the Operations Research Society of Japan, the Reliability Engineering Association of Japan, the Institute of Electrical, Information and Communication Engineers, and the Institute of Electrical and Electronics Engineers.

Hiroyuki Okamura received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 1995, 1997, and 2001, respectively.

In 1998, he joined Hiroshima University as an Assistant Professor, where he has been an Associate Professor with the Department of Information Engineering, Graduate School of Engineering, since 2003, and a Processor with the Department of Information Engineering, Graduate School of Engineering, since 2018. His research interests include performance evaluation, dependable computing, and applied statistics.

Dr. Okamura is a member of the Operations Research Society of Japan, the Institute of Electrical, Information and Communication Engineers, the Japan Society for Industrial and Applied Mathematics, the Information Processing Society of Japan, the Association for Computing Machinery, and the Institute of Electrical and Electronics Engineers.

Tadashi Dohi received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 1989, 1991, and 1995, respectively.

In 1992 and 2000, he was a Visiting Researcher with the Faculty of Commerce and Business Administration, University of British Columbia, Canada, and the Hudson School of Engineering, Duke University, USA, respectively, on the leave from Hiroshima University. Since 2002, he has been a Professor with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. His research interests include reliability engineering, software reliability, and dependable computing.

Dr. Dohi is a member of the Operations Research Society of Japan, the Institute of Electrical, Information and Communication Engineers, the Information Processing Society of Japan, the Reliability Engineering Association of Japan, and the Institute of Electrical and Electronics Engineers. He is also a member of the Editorial Board of the IEEE TRANSACTIONS ON RELIABILITY.