Statistical Analysis of Simulation Output from Parallel Computing

CHEN ZHANG and NAN CHEN, National University of Singapore

This paper addresses statistical output analysis of transient simulations in the parallel computing environment with fixed computing time. Using parallel computing, most unbiased estimators commonly used based on the output sequence compromise. To rectify this issue, this paper proposes an estimation procedure in the Bayesian framework. The proposed procedure is particularly useful when the computing time depends on the output value in each simulation replication. The effectiveness of our method is demonstrated through studies on queuing simulation and control chart simulation.

Additional Key Words and Phrases: Bias reduction; Markov chain Monte Carlo; Output analysis; Parallel computing; Transient simulation

ACM Reference format:
Chen Zhang and Nan Chen. 2017. Statistical Analysis of Simulation Output from Parallel Computing. *ACM Trans. Model. Comput. Simul.* 0, 0, Article 0 (2017), 22 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Simulation is commonly used in many applications to evaluate the performance of complex systems, or to estimate crucial quantities without analytic formulas. Statistical output analysis is a crucial aspect of the success of simulation. In any case one can never make a stochastic simulation run for somewhat arbitrary time and then treat the resulting simulation estimates as the "true" model characteristics. This is because that simulation is naturally stochastic. It uses random samples from certain probability distributions to drive the model through time, and outputs some realizations of these random variables. These realizations may have large variance and, in certain situations, differ greatly from the corresponding true characteristics. Consequently, careless use of simulation outputs leads to possibilities of making erroneous statistical inferences. Because of this importance, so far plenty of literature has been focusing on analyzing simulation outputs to achieve accurate statistical properties (e.g. Kelton and Law [12], Law [13]).

One trend is, as systems become more complex, and their operation mechanisms become more intriguing, simulation models necessarily become increasingly complex to achieve desired accuracy and fidelity. As a result, the simulation models inevitably require longer computing time to execute, and parallel computing grows as a natural solution to reduce the total execution time

This research is partially supported by Singapore AcRF funding R-266-000-085-112, and Future Resilience System under CREATE R-266-000-087-112.
Author's addresses: C. Zhang, N. Chen, Industrial Systems Engineering & Management, National University of Singapore, Singapore, 117576; email: zhangchen@u.nus.edu; isecn@nus.edu.sg. N. Chen is the corresponding author.

XXXX-XXXX/2017/0-ART0 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

for a given number of simulation replications. For a class of transient simulations, embarrassingly parallel setting is often sufficient, where each computing unit simulates one or more replications independently and multiple computing units work simultaneously. Though parallel computing provides promising speedup, it is also accompanied with more statistical issues and challenges for simulation output analysis. One crucial problem for transient simulation is that the mixed output sequence from multiple computing units compromises most commonly used unbiased estimators.

We use an example to illustrate this point. As commonly acknowledged, simulation plays an important role in complex queuing system analysis. When evaluating a queue's average overflow time, for a simulation replication, the queue starts with its queue length equal to 0, and then operates until the queue length flows over its threshold. Clearly the computing time of one replication highly depends on its output value (the overflow time), as shown in Figure 9a. In the parallel setting as Figure 1 illustrates, assume m computing units are employed, with each computing time budget T, to evaluate the average overflow time. All units start running at the same time, and stop when they reach the time budget T. Denote X_{ij} as the amount of computing time required to run the j^{th} replication on the unit i with output Y_{ij}. A sequence of outputs $\{Y_{ij}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, N_i\}$ can be obtained, where N_i is the number of outputs obtained on the unit i. For general simulation scenarios with correct random number generators, we can expect that Y_{ij} and $Y_{i'j'}$ are independent when $i \neq i'$ or $j \neq j'$.

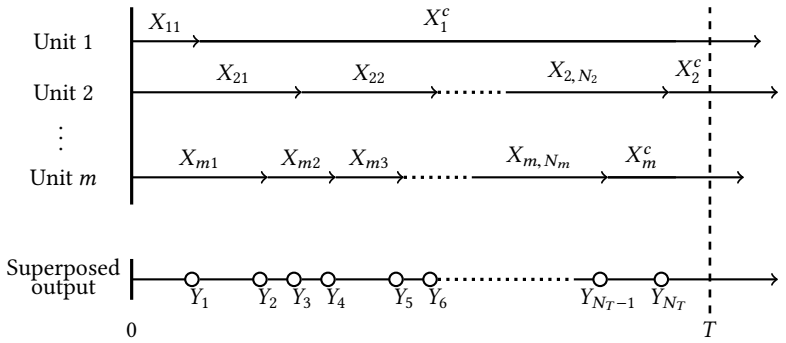


Fig. 1. Illustration of the parallel computing. X_i^c indicates the unfinished replication at T.

A natural estimator of the mean (i.e., $\mathrm{E}(Y)$) is

$$\hat{\mu}_0 = \frac{\sum_{i=1}^{m} \sum_{j=1}^{N_i} Y_{ij}}{\sum_{i=1}^{m} N_i}. \tag{1}$$

However, (1) is biased. Figure 2 shows its bias magnitude when Y follows an exponential distribution with mean 1 and $X_{ij} = Y_{ij}$. The bias is particularly severe for large m or small computing budget T. In fact, as mentioned in [9], when $m \to \infty$, most natural estimators used in sequential computing environments (i.e., $m = 1$), including (1), are guaranteed to converge to wrong quantities in parallel computing environments.

Despite that similar problems broadly exist in practical simulation studies, they receive little attention with (1) taken for granted. To our best knowledge the only two works focusing on this problem are [9] and [8]. They proposed effective methods to reduce the estimation bias under some limited conditions. In particular, both methods need to know exactly which computing unit generates each output Y_{ij}. In addition, these methods need to control the parallel logic to wait

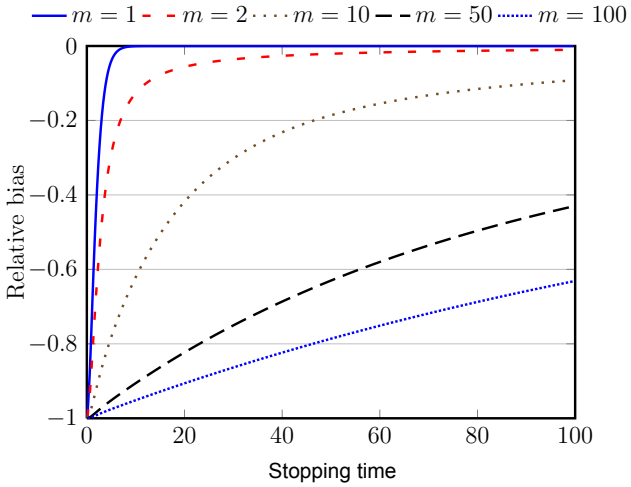


Fig. 2. Relative estimation bias for the mean of exponentially distributed Y_{ij} based on (1).

all or some of the unfinished replications at the stopping time T to finish. Unfortunately, both requirements might not be practical, especially when using commercial cloud computing services.

In fact, from a general point of view, accurate estimation of transient simulation in parallel computing with constrained time budget will not only provide good guidance on transient simulation output analysis, but also shed light upon analysis of other types of simulations. One example where this paradigm is possibly used is to rectify the initialization bias in steady-state simulation ([1]). As firstly proposed by [5], virtually any steady-state simulation can be represented in a regenerative way, and regenerative simulations can be regarded as one kind of transient simulations. Since the problem of initialization bias does not arise in regenerative (transient) simulations, algorithms for designing and analyzing transient simulations can be applied into steady-state simulations to guarantee correct statistical properties ([10, 11]). Another emerging area of increasing importance is about the ranking-and-selection problems. To select the best alternative (subsystem), the first task of the ranking-and-selection procedure is to construct estimators for every alternative with satisfactory accuracy. Otherwise bad estimators will mislead the selection decision. Usually transient measures or regenerative representations of steady-state measures are used in ranking-and-selection procedures. Mindful of the bias issue in time-constrained parallel transient simulations, most of current parallel ranking-and-selection algorithms circumvent this issue by considering fix-replication simulation, i.e., simulating every alternative for a fixed number of replications with random completion time ([17, 18]). However, its disadvantage is the unpredictable simulation time. Furthermore, for cases where the computing time and output value are correlated and a longer computing time indicates a better output, the price of suspending all the other units while waiting for a quite good output on one certain computing unit will be high. [15] addressed and alleviated this time issue by proposing an asymptotic fully sequential procedure. In this regard, if the bias problem in time-constrained parallel transient simulations can be solved, we may get another potential direction for considering time-constrained ranking-and-selection algorithms.

Motivated by the high demands for accurate output analysis of transient simulation in parallel computing and the infancy of literature stressing on it, the paper further explores this field from the following aspects: (1) we analyze the bias in a simple setting to obtain useful insights. We consider

exponentially distributed Y, and linear relation between X and Y. We give the exact bias expression for (1). (2) Inspired by this bias expression, we propose a new estimator in the Bayesian framework without rigid constraints on the parallel computing environment. Our key idea is to remap the superposed output sequence back to the m individual sequential output sequences. Based on this remapping, we can not only restore the superposed output sequence to *i.i.d* data, but also access the computing time that the unfinished replications have used. Then this information is to be used for better statistical analysis. Specifically, we treat the computing indices for $Y_n, n = 1, \ldots, N_T$ as missing data and formulate the estimation as a missing data problem. Then we use Markov chain Monte Carlo algorithm via Gibbs sampler to learn the distribution of the computing indices and the distribution parameters of Y, from which we can consequently obtain the mean value as well as other statistical properties of Y. Numerical results demonstrate the proposed procedure can achieve accurate estimation results. Our procedure is shown to be applicable to both parametric and nonparametric settings. Real case studies present satisfactory empirical evaluations of this procedure as well.

The remainder of the paper is organized as follows. We consider simulation with exponentially distributed outputs and analyze the causes and effects of the estimation bias in detail in Section 2. Then we elaborate our new estimator in Section 3. To evaluate the effectiveness, we compare our method with existing alternatives using some numerical studies in Section 4. We also demonstrate the applicability of the proposal using two examples from queuing simulation and control chart simulation in Section 5. Finally, we conclude the paper with remarks in Section 6. Some technical details are provided in the Appendix.

2 BIAS ANALYSIS IN PARALLEL SIMULATION

In this section, we study the bias of (1) analytically. Following the simulation setting illustrated in Figure 1, we denote $S_{n_i} = X_{i1} + \cdots + X_{in_i}$ as the time taken for unit i to complete the first n_i replications. Given the simulation budget T, the number of completed replications from unit the i, N_i, is a counting process with $N_i = \sup\{n_i, 0 < S_{n_i} \leq T\}$. When m computing units run independently, N_i and N_j are independent for any $i \neq j$. As a result, the outputs from all the m units form a superposed renewal process ([14]) with $N_T = \sum_{i=1}^{m} N_i$ (It is to be noted that N_i is a function of T, and $N_i(T)$ may be more precise. However for notation simplicity, we omit T hereinafter, when no confusion will occur). The output sequence of this superposed process is mixed by outputs from each unit, and is ordered by the completion times of individual outputs. To make it general, we assume that we only observe the output values and synchronized computer times of the superposed output sequence, but we do not know which computing unit generates each output. Without loss of generality, we assume the speed of each computing unit is known, which can also be conveniently estimated through a pilot run.

Unfortunately, a general superposed renewal process is difficult to analyze. To obtain some insights on the cause of bias, we consider a special case. In particular, we assume Y_{ij} follows the exponential distribution with mean μ. The computing time is linear with the output value, i.e., $X_{ij} = Y_{ij}/\lambda_i$ where λ_i is the "speed" of unit i. As a result, $N_i(t)$ reduces to a Poisson process, and $N(T)$ is a superposed Poisson process. Following the properties of Poisson processes, conditioning on N_i, S_{N_i} has the density

$$f_{S_{N_i}}(s|N_i) = \frac{N_i s^{N_i-1}}{T^{N_i}}, \quad 0 \leq s \leq T,$$

and (1) can be represented as $\hat{\mu}_0 = \sum_{i=1}^{m} \lambda_i S_{N_i} / \sum_{i=1}^{m} N_i$.

Proposition 2.1. *The expectation of (1) can be computed as*

$$\mathrm{E}\,\hat{\mu}_0 = \frac{\mu\tau e^{-\tau}}{1-e^{-\tau}}\left[(1-m)\cdot\Gamma(1)+\sum_{i=1}^{m}(1-p_i)\cdot\Gamma(1-p_i)\right] + \frac{\mu}{1-e^{-\tau}}\left[m-e^{-\tau}\tau-\sum_{i=1}^{m}e^{-\tau p_i}\right],$$

where $\tau = \sum_{i=1}^{m}\lambda_i T/\mu$ *is the total "standardized" computing time,* $p_i = \lambda_i/\sum_{j=1}^{m}\lambda_j$ *is the proportion of the unit* i *in the overall computing time, and* $\Gamma(u)$ *is defined as the integral value* $\int_0^u (e^{\tau x}-1)/x\cdot \mathrm{d}x$.

From Proposition 2.1, it is easy to show that $\hat{\mu}_0$ is asymptotically unbiased as $T\to\infty$, but is clearly negatively biased for a fixed T. When $m = 1$, $\mathrm{E}\,\hat{\mu}_0$ only includes the second part in Proposition 2.1, and the corresponding bias reduces to $-e^{-\tau}\tau\mu/(1-e^{-\tau})$. It reveals that what we really estimate is the mean of the truncated distribution, i.e., $\mathrm{E}\,\hat{\mu}_0 = \mathrm{E}\,[Y|X\le T]\le \mathrm{E}\,[Y]$ ([7]).

When $m > 1$, in addition to the bias caused by this truncation, bias due to the mixed output sequence also comes into play. More specifically, since the simulation time is proportional to the output value, i.e., $X_{ij} = Y_{ij}/\lambda_i$, it is expected that the replications with smaller outputs complete earlier when all the replications start at the same time. In other words, the outputs no longer have *i.i.d.* values, but are "ordered" probabilistically according to their values. The situation is further complicated because each unit runs multiple replications sequentially. To better understand the difference between sequential computing with $m = 1$ and parallel computing with $m > 1$, ([15]) describes the simulation process as a queuing model. The simulation replications represent customers, which are waiting in the queue in a predetermined order. The m computing units represent m servers. For a single server system, the input and output sequences of the queue are always the same. However, for $m > 1$, the output sequence is stochastic, and different from the input sequence because the service time of different customers is random. See Figure 3 for a better illustration. In summary, since the estimator $\hat{\mu}_0$ does not consider the inherent "order" in the output sequence, $\hat{\mu}_0$ inevitably underestimates the true mean μ. In fact, it can be shown that for a fixed T, as $m\to\infty$, the bias converges to a value in the order of $O(1/T)$.

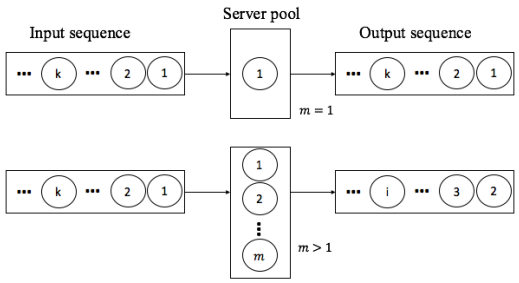


Fig. 3. Queue analogy of the simulation process.

The aforementioned analysis points out two important issues that need to be addressed to reduce the estimation bias. First, the unfinished replications at the stopping time T should be considered. They are similar to right censored data in survival analysis or reliability engineering. Neglecting right censored data typically leads to truncation bias. Second, it is important to recover the outputs from the mixed output sequence to *i.i.d* outputs so that the bias due to the "ordering"

can be avoided. In the next section, we propose a Bayesian approach to analyze the outputs to address both issues.

3 A BAYESIAN ESTIMATOR FOR BIAS REDUCTION

In this section, we are interested in estimating the distributional information of the random variable Y through simulation. We first assume Y is parametric with the probability density function (PDF) $f_y(Y;\Theta)$ and the survival function $R_y(Y;\Theta)$. To allow for more flexibility to incorporate prior information of Θ, here we propose to estimate Θ in the Bayesian framework. Then once we obtain the posterior distribution $\pi(\Theta|Y_{1:N_T})$, we can easily derive the statistical properties of Y, such as $\mu = \mathrm{E}(Y)$. For an easier exposition, we consider the relation between the computing time and the output value is known, i.e., $g_i(\cdot)$ is known in $X_{ij} = g_i(Y_{ij})$. In this paper we assume $g_i(\cdot)$ is a monotone increasing function, which is also continuous and derivable, and we denote $g_i^{-1}(\cdot)$ as the inverse function of $g_i(\cdot)$. According to the derivative of inverse functions, we have $dg_i(Y_{ij})/dY_{ij} = dX_{ij}/dg_i^{-1}(X_{ij})$. Then the PDF of X_{ij} can be easily derived as

$$f_{x_i}(X_{ij}) = f_y(g_i^{-1}(X_{ij});\Theta)\frac{dg_i^{-1}(X_{ij})}{dX_{ij}} = \frac{f_y(Y_{ij};\Theta)}{\frac{dX_{ij}}{dg_i^{-1}(X_{ij})}} = \frac{f_y(Y_{ij};\Theta)}{g_i'(Y_{ij})},$$

where $Y_{ij} = g_i^{-1}(X_{ij})$ and $g_i'(Y_{ij}) = \frac{dg_i(Y_{ij})}{dY_{ij}}$. Similarly, we can derive the survival function of X_{ij}, i.e., $R_{x_i}(X_{ij})$ as $R_y(g_i^{-1}(X_{ij});\Theta)$. Hereinafter, we simply denote $f_y(Y_{ij};\Theta)$ as $f(Y_{ij};\Theta)$ and $R_y(Y;\Theta)$ as $R(Y;\Theta)$ when no confusion is introduced. Correspondingly, we denote $f_{x_i}(X_{ij})$ as $f(Y_{ij};\Theta)/g_i'(Y_{ij})$, and $R_{x_i}(X_{ij})$ as $R(g_i^{-1}(X_{ij});\Theta)$. Later we generalize our proposed method to nonparametric cases where Y's distribution form, i.e., $f(\cdot)$, is unknown. In particular, we propose to approximate $f(Y;\Theta)$ by a phase-type distribution, and present the corresponding Gibbs sampler procedure to obtain the distribution of Y. The methodology in detail is discussed as follows.

3.1 Missing data formulation

From the analysis in Section 2, we can observe that to reduce the estimation bias, we need on one hand to restore the output sequence to *i.i.d* data, and on the other to get the censored computing time that the last unfinished replications have used before T. Both requirements can be fulfilled if we know exactly the index of computing unit from which any given output is generated. This information transforms the parallel output sequence into m sequential output sequences, which then can be easily analyzed using existing approaches.

When computing indices for $Y_n, n = 1, \cdots, N_T$ are unknown, in this paper, we treat these indices as missing data, and formulate the estimation as a missing data problem. More specifically, we define the missing unit index information $Z_n, n = 1, \cdots, N_T$, where $Z_n = i$ if the n^{th} output of the superposed output sequence comes from unit i. Clearly, $Z_n, n = 1, \cdots, N_T$, are sufficient to restore the parallel output sequence to m sequential output sequences. Denote $\mathbf{Y} = [Y_1, \cdots, Y_{N_T}]$ and $\mathbf{Z} = [Z_1, \cdots, Z_{N_T}]$. Then if $\mathbf{Z}$ is known, given the complete data $(\mathbf{Y}, \mathbf{Z})$, it is straightforward to compute the posterior distribution $\pi(\Theta|\mathbf{Y}, \mathbf{Z})$, according to the Bayes' Theorem,

$$\pi(\Theta|\mathbf{Y}, \mathbf{Z}) \propto p_0(\Theta) \cdot p(\mathbf{Y}|\Theta, \mathbf{Z}), \tag{2}$$

where $p_0(\Theta)$ is the prior distribution of the parameter Θ.

In particular, from $\mathbf{Z}$, we may achieve the number of completed replications of unit i, i.e., $N_i = \sum_{n=1}^{N_T} \mathcal{I}(Z_n = i)$, and its corresponding censored simulation time $X_i^c = T - \sum_{n=1}^{N_T} \mathcal{I}(Z_n = i) g_i(Y_n)$. Here $\mathcal{I}(\cdot)$ is the indicator function which equals one when the condition is true and zero otherwise. Then similar to the reliability data analysis, the total likelihood $p(\mathbf{Y}|\Theta, \mathbf{Z})$ includes the PDFs of

the simulation times for the completed samples, i.e., $f_{x_{Z_n}}(X_n), n = 1, \ldots, N_T$, and the survival functions of the censored simulation times for the uncompleted samples, i.e., $R_{x_i}(X_i^c), i = 1, \ldots, m$. Consequently, we have

$$P(\mathbf{Y}|\Theta, \mathbf{Z}) = \prod_{n=1}^{N_T} f_{x_{Z_n}}(X_n) \prod_{i=1}^{m} R_{x_i}(X_i^c) = \prod_{n=1}^{N_T} \frac{f(Y_n;\Theta)}{g'_{Z_n}(Y_n)} \prod_{i=1}^{m} R(g_i^{-1}(X_i^c);\Theta).$$

As a result, the posterior distribution $p(\mathbf{Y}|\Theta, \mathbf{Z})$ becomes

$$p(\mathbf{Y}|\Theta, \mathbf{Z}) = \prod_{n=1}^{N_T} \frac{f(Y_n;\Theta)}{g'_{Z_n}(Y_n)} \prod_{i=1}^{m} R\left(g_i^{-1}(X_i^c);\Theta\right),$$

with $g_i^{-1}(\cdot)$ being the inverse function of $g_i(\cdot)$. It should be noted that to justify this inverse function exists, here $g^i(\cdot)$ is assumed to be a monotone function.

For commonly used distributions of Y, abundant literature has been developed to efficiently compute the posterior distribution of Θ in (2) with right censored data. Then (2) can easily lead to the desired estimation of any statistical property of Y, such as $\mathrm{E}(Y)$.

3.2 Gibbs sampler and proposal distribution

Unfortunately, $\mathbf{Z}$ is not known to make (2) directly usable. A common practice is to employ a Gibbs sampler to iteratively update the distributional information of both $\mathbf{Z}$ and Θ, i.e.,$\pi(\Theta, \mathbf{Z}|\mathbf{Y})$. In our problem, it is natural to consider two blocks of quantities, $\mathbf{Z}$ and Θ, and to divide their full conditional distribution as $\pi(\mathbf{Z}|\Theta, \mathbf{Y})$ and $\pi(\Theta|\mathbf{Z}, \mathbf{Y})$, respectively. In particular, the Gibbs sampler constructs a Markov chain with the limiting distribution being the target posterior distribution $\pi(\Theta, \mathbf{Z}|\mathbf{Y})$. Therefore, the posterior samples can be obtained by recursively sampling from these two conditional distributions, using the most recent values of the conditioning variables at each step. It can be summarized in the following iterative sampling scheme accordingly.

ALGORITHM 1: Gibbs Sampler

1: Set $k = 1$, generate $\Theta^{(0)}$ from $p_0(\Theta)$.
2: Generate $\mathbf{Z}^{(k)}$ from $\pi(\mathbf{Z}^{(k)}|\Theta^{(k-1)}, \mathbf{Y})$.
3: Generate $\Theta^{(k)}$ from $\pi(\Theta^{(k)}|\mathbf{Z}^{(k)}, \mathbf{Y})$.
4: Set $k = k + 1$; if $k \leq K + B$, go back to step 2; otherwise stop and return $\{\Theta^{(k)}, \mathbf{Z}^{(k)}\}, k = B + 1, \ldots, B + K$, where B is the number of burn-in samples.

The iterative steps in Algorithm 1 can generate K samples from the joint posterior distribution $\pi(\Theta, \mathbf{Z}|\mathbf{Y})$ as $\{\Theta^{(k)}, \mathbf{Z}^{(k)}\}, k = 1, ..., K$. For every sample $\Theta^{(k)}$, we can calculate the corresponding mean $\mu^{(k)}$ of the distribution parametrized by $\Theta^{(k)}$. These samples $\mu^{(1:K)}$ can be collectively used for point and interval estimation of μ. In a similar way, we can estimate other distribution properties of Y, such as its variance or quantiles based on $\Theta^{(1:K)}$.

While Step 3 can be solved in (2), obviously the most challenging and also the unique part in our problem is to take samples from $\pi(\mathbf{Z}|\Theta, \mathbf{Y})$. Using noninformative prior for $\mathbf{Z}$, i.e., $p_0(\mathbf{Z}) \propto 1$, the conditional distribution is proportional to $p(\mathbf{Y}|\Theta, \mathbf{Z})$. Equivalently,

$$\pi(\mathbf{Z}|\Theta, \mathbf{Y}) \propto \prod_{n=1}^{N_T} \frac{f(Y_n;\Theta)}{g'_{Z_n}(Y_n)} \prod_{i=1}^{m} R\left(g_i^{-1}(X_i^c);\Theta\right). \tag{3}$$

However, (3) is too complicated to warrant easy sampling. Alternatively, we adopt the Metropolis-Hastings (M-H) algorithm to take samples from (3). In detail, we construct a proposal distribution $q(\mathbf{Z}|\Theta, \mathbf{Y})$ which is easier to take samples from. At each step, we sample $\mathbf{Z}^*$ from $q(\mathbf{Z}|\Theta, \mathbf{Y})$. We

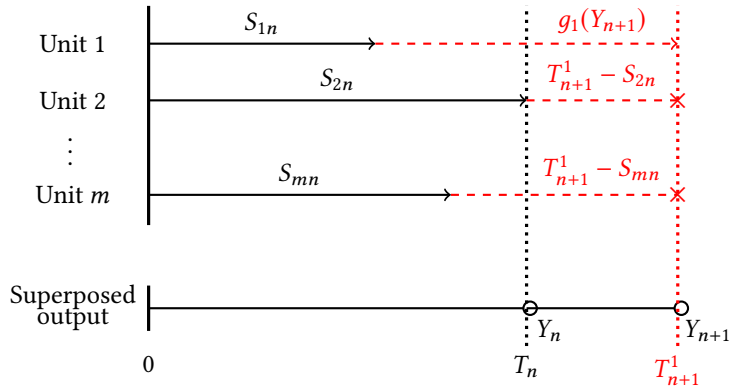


Fig. 4. Sequential construction of the proposal distribution $q(\mathbf{Z}|\Theta, \mathbf{Y})$.

accept $\mathbf{Z}^*$ as a sample from $\pi(\mathbf{Z}|\Theta, \mathbf{Y})$ with probability

$$\alpha(\mathbf{Z}, \mathbf{Z}^*|\mathbf{Y}) = \min\left\{\frac{p(\mathbf{Y}|\mathbf{Z}^*, \Theta)q(\mathbf{Z}|\Theta, \mathbf{Y})}{p(\mathbf{Y}|\mathbf{Z}, \Theta)q(\mathbf{Z}^*|\Theta, \mathbf{Y})}, 1\right\},$$

where $\mathbf{Z}$ is a valid sample in the previous step. Any discrete distribution with nonzero support in $\{1, \ldots, m\}$ can be the proposal distribution and guarantee the convergence of the M-H incorporated Gibbs sampler [16]. However, a good proposal should guarantee the acceptance probability $\alpha(\mathbf{Z}, \mathbf{Z}^*|\mathbf{Y})$ not very small, otherwise the chain will explore the parameter space too slowly. In general, an ideal proposal distribution among those in a parametric class is that best approximates the posterior distribution $\pi(\mathbf{Z}|\Theta, \mathbf{Y})$, according to some metrics, and yet easy to take samples from.

Considering $Y_n, n = 1, \cdots, N_T$ are generated sequentially in the output sequence, we construct the proposal distribution $q(\mathbf{Z}|\Theta, \mathbf{Y})$ sequentially as

$$q(Z_1^*|\Theta, Y_1) \cdot \prod_{n=1}^{N_T-1} q(Z_{n+1}^*|\Theta, Z_{1:n}^*, Y_{1:n+1}), \tag{4}$$

where $q(Z_1^*|\Theta, Y_1)$ is the initial sampling probability. We propose a multinomial distribution to define the probability $q(Z_{n+1}^*|\Theta, Z_{1:n}^*, Y_{1:n+1})$. In more details, given the previous assignments $Z_j^*, j = 1, \cdots, n$, we can calculate the running time of each computing unit since its last output. We define $S_{kn} = \sum_{j=1}^{n} \mathcal{I}(Z_j^* = k)g_k(Y_j)$ as the time when the most recent output from unit k is generated. Consequently, $T_n = \max\{S_{kn}, k = 1, \ldots, m\}$ is the time when the most recent output among all units, i.e., Y_n, is generated.

If the $(n+1)^{\text{th}}$ output Y_{n+1} is generated from unit i, the most recent output time becomes $T_{n+1}^i = S_{in} + g_i(Y_{n+1})$. At the same time, it also implies the remaining $m-1$ units have not generated any output between S_{kn} and T_{n+1}^i for all $k \neq i$. In other words, the current replication on any unit $k \neq i$ runs longer than $T_{n+1}^i - S_{kn}$, or equivalently the next output value from unit k is larger than $g_k^{-1}(T_{n+1}^i - S_{kn})$. See Figure 4 for a schematic illustration with $i = 1$. Clearly, we also need to ensure that $T_{n+1}^i > T_n$ because Y_{n+1} is generated later than T_n. Considering these properties, we can compute the probability $q(Z_{n+1}^* = i|\Theta, Z_{1:n}^*, Y_{1:n+1})$ by the instantaneous risk of generating Y_{n+1}

from unit i and the survival probability at T^i_{n+1} for the remaining $m-1$ units. In detail, we define

$$\xi_{n+1}(i) \equiv \mathcal{I}(T^i_{n+1} > T_n)\frac{f(Y_{n+1})}{g'_i(Y_{n+1})}\prod_{k\neq i} R\left(g_k^{-1}\left(T^i_{n+1} - S_{kn}\right);\Theta\right). \tag{5}$$

The sampling probability can then be obtained by normalizing the constants $\xi_{n+1}(i)$, i.e., $q(Z^*_{n+1} = i|\Theta, Z^*_{1:n}, Y_{1:n+1}) = \xi_{n+1}(i)/\sum_{j=1}^m \xi_{n+1}(j)$, which follows a multinomial distribution. The initial probability $q(Z^*_1|\Theta, Y_1)$ can be defined in a similar way by setting $S_{i0} = 0, i = 1, \cdots, m$. Actually, this proposal distribution of Z_n is motivated by the "competing risk" in reliability analysis [20], where failures of a system have m types. Given the previous failure times of these m types, we want to predict the next failure type. In particular, it assumes the probability that the next failure belongs to type i follows a multinomial distribution, and its mass probability distribution is proportional to the instantaneous risk of failure type i times the survival probabilities of the other $m-1$ failure types.

The proposed sequential construction of $q(\mathbf{Z}|\Theta, \mathbf{Y})$ imitates the evolution of the output sequence, and is expected to approximate $\pi(\mathbf{Z}|\Theta, \mathbf{Y})$ well. Its performance has also been well tested through numerical studies and real case examples, which will be discussed later.

Remark It should be noted when T or m is big, sampling $\mathbf{Z}^*$ according to the proposal distribution in Equations (4, 5) becomes time consuming. It is because in both cases N_T becomes big, and the searching space of $\mathbf{Z}^*$ increases. However, the feasible space of $\mathbf{Z}^*$ cannot increase as fast as the searching space. Consequently, the proportion of the feasible space in the searching space becomes smaller as N_T increases. The slow increase of the feasible space is mainly caused by the time order constraint of the superposed output sequence, i.e., Y_n should be completed after Y_{n-1} for $n = 2, \cdots, N_T$. In this way we can guarantee the virtual output sequence $\{Y^*_{1:N_T}\}$ formulated by $\{Z^*_{1:N_T}\}$ has the same time order as $\{Y_{1:N_T}\}$. However, actually this constraint does not influence the estimation of Θ too much, since (2) is only related to the censored observations $Y^c_{1:m}$. In this sense as long as $\{Z^*_{1:N_T}\}$ ensures a good sample of the censored data which has a big likelihood for (3), this $\{Z^*_{1:N_T}\}$ can give a good sample of Θ. When we release this constraint, we enlarge the feasible region of $\{Z^*_{1:N_T}\}$, and therefore enlarge the sample space of $\{Y^c_{1:m}\}$ and correspondingly Θ. It means that the Gibbs sampler needs more iterations to find the steady state of Θ and more burn-in samples, consequently leading to a slower convergence speed. However, the computing time for increasing the sampler length is much smaller compared with that for finding a feasible solution of $\{Z^*_{1:N_T}\}$ with unreleased constraints. From this point of view, we propose a substituted proposal distribution for $\{Z^*_{1:N_T}\}$ by removing $\mathcal{I}_{in}$ for large scale parallel computing with big m and T, i.e.,

$$\xi_{n+1}(i) = \frac{f(Y_{n+1})}{g'_i(Y_{n+1})}\prod_{k\neq i} R\left(g_k^{-1}\left(\left[T^i_{n+1} - S_{kn}\right]^+\right);\Theta\right).$$

This proposal demonstrates to be quite efficient in the aspects of speeding up the sampling time on one hand, and on the other delivering accurate estimations as shown in Section 4.

3.3 A nonparametric procedure by phase-type approximation

For nonparametric Y with $f(\cdot)$ unknown, we may still implement our procedure above by approximating Y using a phase-type distribution (PHD). This is because, for transient simulation, the simulation (termination) time can be formulated as the first hitting time of a Markov process, which can be conveniently modeled as a PHD. Furthermore, PHD is a commonly used nonparametric

distribution family. In fact, it can approximate any continuous probability distribution with nonnegative real values arbitrarily close ([19]). With these in mind, we present a particular Gibbs sampler algorithm for the PHD parameters.

3.3.1 Phase-type approximation. We first use two typical examples to illustrate the nature of transient simulation. The first is the average operation time before overflow in queuing system analysis. The queue starts with the initial queue length 0, and overflows when the queue length is bigger than a pre-specific value L. The overflow time Y is the total operation time of the queue before overflow. For a general $G/G/1$ queue, the distribution of Y has no analytical form, therefore $\mathrm{E}(Y)$ needs to be estimated via simulation. In one simulation replication, we start the queue with its terminating criteria as $Q > L$, and wait for its operating until terminating. As we know, in queuing system analysis, generally the queue length Q can be described by a Markov chain. If we set $Q > L$ as the absorbing state, then Y is the evolution time of the chain before entering the absorbing state. Another example arises in the control chart performance simulation. Usually we evaluate a chart in terms of average run length and use simulation to estimate it. In every simulation replication, the control chart runs until its charting statistic T goes out of the control limit L, and returns the run length Y of this replication. Actually, we can discretize the range of the charting statistic T within the control limit into several regions as transient states, and treat the region out of the limit as the absorbing state. Then we can treat the evolution of T as a Markov chain, and the run length Y is the first hitting time of chain on the absorbing state. This Markov chain approach has been vastly used as the most effective approximation method to study the run length characteristics of complex charts ([3, 21]).

As illustrated in the two examples above, for general transient simulations, there is a characteristic (or event) T denoting the current system state of the simulation model and a terminating area for T defined according to a pre-specific criterion. The simulation starts with an initialized T and runs until T reaches the terminating area. Then the simulation returns the output Y which is a measure of the evolving time of T. In this way T can be viewed as a continuous-state homogeneous Markov chain. Under very mild conditions, the chain can be accurately approximated using a finite-state homogeneous Markov chain by discretizing the range of T excluding the terminating area into p regions (transient states) and one region of the exact terminating area (absorbing state). Hence the output Y can be defined as a random variable describing the evolution time until absorption of the chain, i.e., a PHD. The formal definition of the PHD is introduced as bellows.

Definition 3.1. Continuous Phase-type Distribution (PHD) Consider any continuous-time Markov chain on a finite discrete state space with total $(p+1)$ states as $0, 1, \cdots, p$ where state 0 is the only absorbing state. Without loss of generality, the infinitesimal generator of the Markov chain can be expressed as:

$$\mathbf{Q} = \left[\begin{array}{c:c} 0 & \mathbf{0} \\ \hdashline \mathbf{s}^0 & \mathbf{S} \end{array}\right], \tag{6}$$

where $\mathbf{S} = (S_{ij})$ is the matrix of transition rates from non-absorbing state i to j for $i \neq j$, $i, j \in \{1, \cdots, p\}$. $\mathbf{s}^0$ is the vector of transition rates from state i to the absorbing state and $\mathbf{0}$ is a $p \times 1$ vector. A PHD with order p is defined to be the distribution of the time to enter the absorbing state of a continuous-time Markov chain with the generator as (6) and the initial state probabilities as $[1 - \boldsymbol{\pi}_0\mathbf{1}, \boldsymbol{\pi}_0]$. Here $\boldsymbol{\pi}_0$ is a $1 \times p$ vector with $0 < \sum_{i=1}^{p} \pi_{0i} \leq 1$. Then the CDF and PDF of the PHD variable Y are

$$F(Y; \mathbf{S}, \boldsymbol{\pi}_0) = 1 - \boldsymbol{\pi}_0 e^{\mathbf{S}Y}\mathbf{1}$$
$$f(Y; \mathbf{S}, \boldsymbol{\pi}_0) = \boldsymbol{\pi}_0 e^{\mathbf{S}Y}\mathbf{s}^0$$

Another motivation for using a PHD to approximate the output distribution is that the PHD is in the space of all continuous positive real axis and hence can approximate any continuous probability distribution arbitrarily close ([19]). As pointed by [4], since any distribution has a rational Laplace-Stieltjes transform, any distribution can be approximated by a set of exponentially distributed "stages" or "phases", and consequently represented by a phase-type distribution.

3.3.2 Gibbs sampling for the approximate phase-type distribution. Now we introduce the Gibbs sampler for the PHD estimation in detail. Suppose $Y \sim \mathrm{PH}_p(\cdot; \mathbf{S}, \boldsymbol{\pi}_0)$ with order p. We assume p is fixed in advance, and want to estimate $\Theta = \{\mathbf{S}, \boldsymbol{\pi}_0\}$ and the corresponding $\mathrm{E}(Y)$ by taking the censored data into account. In particular, sampling $\mathbf{Z}$ is the same as the general estimation procedure in Section 3.2. Here we only stress on how to sample Θ.

Since every Y_n only represents the time to absorption but does not tell the detailed Markov jump path such as where it starts, which of the states it visits and for how long, mere Y_n without the jump path information is still incomplete to estimate Θ. Consequently, further data augmentation for the path information is also needed in the Gibbs sampler. Specifically, we treat every Markov jump path $J_n = \{\mathbf{C}_n, \mathbf{U}_n\}$ $(n = 1, \cdots, N_T)$ as latent variables which include all its r_n visited states before absorption as $\mathbf{C}_n = \left[c_1, \cdots, c_{r_n}\right]$ (c_1 is the initial state) and the corresponding sojourn time for every state as $\mathbf{U}_n = \left[u_1, u_1, \cdots, u_{r_n}\right]$. For the censored observations $Y_i^c, i = 1, \cdots, m$, we also define $J_i^{'} = \{\mathbf{C}_i^{'}, \mathbf{U}_i^{'}\}$ representing the whole jump path of Y_i^c until absorption at an unknown time point bigger than Y_i^c.

Combining these two kinds of latent variables, the joint posterior distribution becomes $\pi(\Theta, J_{1:N_T}, J_{1:m}^{'} | Y_{1:N_T}, Y_{1:m}^c)$. We further divide the variables into three subgroups $\{J_{1:m}^{'}\}$, $\{J_{1:N_T}\}$, and $\{\mathbf{S}, \boldsymbol{\pi}_0\}$, and take samples from their full conditional distributions using the Gibbs sampler. We summarize the main results as bellows.

- **Sampling from** $\pi(J_{1:m}^{'} | \Theta, Y_{1:m}^c)$
  $\pi(J_i^{'} | \Theta, Y_i^c)$ for $i = 1, \cdots, m$ can be sampled via standard Markov chain simulation with rejective sampling as shown in Algorithm 2.

ALGORITHM 2: Simulate $\pi(J^{'} | \Theta, Y^c)$

1: Set $r = 1$, draw the starting state c_1 from $\boldsymbol{\pi}_0$. Initialize $\mathbf{C}^{'} = c_1$ and $\mathbf{U}^{'}$ to be empty.
2: Draw the sojourn time, u_r in the state c_r from the exponential distribution with rate $-S_{c_r c_r}$, and update $\mathbf{U}^{'} = \left[\mathbf{U}^{'}, u_r\right]$.
3: Draw the state move, l, from the multinomial distribution with PMF as

$$\mathrm{P}(l) = \frac{S_{c_r l}}{-S_{c_r c_r}}, l \neq c_k$$

4: If $l = 0$, then go to Step 5; Otherwise update $r = r + 1$, $c_r = l$ and $\mathbf{C}^{'} = \left[\mathbf{C}^{'}, c_r\right]$, then loop to Step 2.
5: if $\sum_{j=1}^{r} u_j > Y^c$, accept the current $J^{'} = \{\mathbf{C}^{'}, \mathbf{U}^{'}\}$, otherwise, reject the current $J^{'}$ and resample from Step 1.

- **Sampling from** $\pi(J_{1:N_T} | \Theta, Y_{1:N_T})$
  To simulate $\pi(J_n | \Theta, Y_n)$ for $n = 1, \cdots, N_T$, we resort to the M-H algorithm by considering another Markov path J_n^* with absorption time beyond Y_n as the proposal distribution. The detailed procedure is shown in Algorithm 3. After K steps of this M-H algorithm, the final process J_n^* can be regarded as a sample from the target Markov chain. Its detailed derivation can be refereed to [2].

ALGORITHM 3: Simulate $\pi(J|\Theta, Y)$

1: Set $k = 1$, generate $J^{'} = \{\mathbf{C}^{'}, \mathbf{U}^{'}\}$ from $\pi(J^{'}|\Theta, Y)$. Obtain $J = \{\mathbf{C}, \mathbf{U}\}$ by truncating $J^{'}$ at $r = \max_i(\sum_{j=1}^{i} u_j < Y)$ and only reserving the first r components, i.e., resetting $\mathbf{C} = \left[c_1^{'}, \cdots, c_r^{'}\right]$, and $\mathbf{U} = \left[u_1^{'}, \cdots, u_{r-1}^{'}, Y - \sum_{j=1}^{r-1} u_j^{'}\right]$.
2: Obtain another J^* by firstly generating $J^{*^{'}}$ from $\pi(J^{*^{'}}|\Theta, Y)$ and then truncating $J^{*^{'}}$ at r^* in the same way as Step 1.
3: Draw $v \sim \text{Uniform}(0, 1)$.
4: If $v \leq \min(1, \frac{s^0_{c_r}}{s^0_{c^*_{r^*}}})$, where $s^0_{c_r}$ and $s^0_{c^*_{r^*}}$ are the absorbing rates at the last state c_r for J and $c^*_{r^*}$ for J^* as defined earlier, then replace J with J^*.
5: Set $k = k + 1$; If $k < K$, go back to step 2; Otherwise terminate and return J.

- **Sampling from** $\pi(\Theta|J_{1:N_T}, J^{'}_{1:m})$
  The target density of $\{\mathbf{S}, \boldsymbol{\pi}_0\}$ based on the augmented complete observations can be written as

$$\pi(\mathbf{S}, \boldsymbol{\pi}_0|J_{1:N_T}, J^{'}_{1:m}) \propto p_0(\mathbf{S}, \boldsymbol{\pi}_0)p(J_{1:N_T}, J^{'}_{1:m}|\mathbf{S}, \boldsymbol{\pi}_0)$$

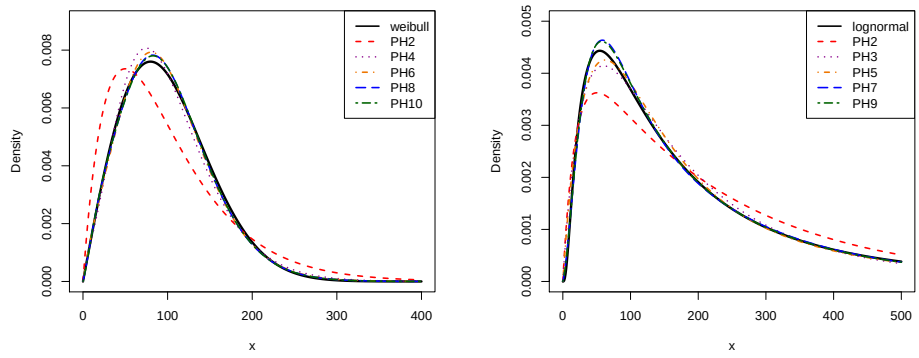
  where $p_0(\mathbf{S}, \boldsymbol{\pi}_0)$ is the prior distribution. Here instead of sampling the matrix $\mathbf{S}$ directly, we choose to sample its off-diagonal items $\{S_{kl}; k, l = 1, \cdots, p, k \neq l\}$ and the absorbing rates $\{s^0_k, k = 1, \cdots, p\}$ with the relation $s^0_k = -\sum_{l=1}^{p} S_{kl}$ in mind. By setting proper independent conjugate prior distributions, their marginal posterior distributions have close forms, i.e.,

$$\left.\begin{array}{l} p_0(s^0_k) \sim \text{Gamma}(\alpha_{k0}, \beta_k) \\ p_0(S_{kl}) \sim \text{Gamma}(\alpha_{kl}, \beta_k) \\ p_0(\boldsymbol{\pi}_0) \sim \text{Dirichlet}(\theta_1, \cdots, \theta_p) \end{array}\right\} \longrightarrow \begin{cases} \pi(s^0_k|J_{1:N_T}, J^{'}_{1:m}) \sim \text{Gamma}(\alpha_{k0} + N_{k0}, \beta_k + r_k) \\ \pi(S_{kl}|J_{1:N_T}, J^{'}_{1:m}) \sim \text{Gamma}(\alpha_{kl} + N_{kl}, \beta_k + r_k) \\ \pi(\boldsymbol{\pi}_0|J_{1:N_T}, J^{'}_{1:m}) \sim \text{Dirichlet}(\theta_1 + b_1, \cdots, \theta_p + b_p) \end{cases} \quad (7)$$

  where
  - b_k: the total number of jump paths with initial state k;
  - r_k: the total time spent in the state k across all the paths $\{J_{1:N_T}, J^{'}_{1:m}\}$;
  - N_{kl}: the total number of transition moves from k to l across all the paths.
  - N_{k0}: the total number of absorbing moves from state k to state 0 across all the paths.

The detailed derivation of (7) is shown in the Appendix. With these three steps above, we can get a sample $\Theta^{(k)}$ and $\mu^{(k)}$ via the relation $\mu = -\boldsymbol{\pi}_0\mathbf{S}^{-1}\mathbf{1}$. Combining with sampling step of $\mathbf{Z}$ in the Section 3.2, we can get the full Gibbs sampler for the PHD approximation.

Remark One main concern with the PHD approximation is how to choose the number of phases, i.e., the order p for the PHD approximation. We illustrate this point using two examples, fitting Weibull and log-normal distributions by PHD with different orders. For every distribution 1000 complete samples are generated, based on which we use the Gibbs sampler for $\pi(\Theta, J_{1:1000}|Y_{1:1000})$ to estimate the PHD parameters. Figure 5 draws the density function of the true distribution and the fitted PHD. We can see that when p increases firstly, the approximate PHD approaches the true density with smaller difference. Later though p goes on increasing, the approximation performance no longer improves significantly. This reveals that there is a minimum value of p required for a good approximation, and yet p bigger than this minimum value will no longer provide significant performance improvement. Moreover, since the complexity of the approximate PHD representation depends on the size of the state space of the underlying Markov chain, a smaller p is more desirable to keep the number of states low and consequently ensures a more stable numerical result. In our



(a) Weibull distribution (b) log-normal distribution

Fig. 5. PHD approximation for (a) Weibull (α, β) distribution with scale $\alpha = 2$ and shape $\beta = 113$; (b) Log-normal (μ, σ) distribution with mean $\mu = 5$ and standard deviation $\sigma = 1$.

cases, p can be decided in advance by comparing different p's fitting performance for the complete observations $\{Y_{1:N_T}\}$. Furthermore, as shown in the numerical studies, the proposed Gibbs sampler for the PHD estimation is very robust to the choice of p and can remain accurate even in the cases where the chosen p is a bit bigger than the true p. Furthermore, given a chosen p, it is also important to select the prior distribution for (7). In reality, with the observations $\{Y_{1:N_T}\}$, we can first use the maximum likelihood estimation in [6] to get a point estimation of $\boldsymbol{\pi}_0$, $\mathbf{S}$ and $\mathbf{s}^0$, denoted as $\boldsymbol{\pi}_0^\star$, $\mathbf{S}^\star$ and $\mathbf{s}^{0\star}$. According to them, we further set the hyper-parameters for (7). In particular, we set $\boldsymbol{\theta} = \boldsymbol{\pi}_0^\star$ for the Dirichlet distribution. We let all the Gamma distributions have the same scale parameter as $\beta_k = \beta_0$ for $k = 1, \ldots, p$, and set the shape parameters of these Gamma distributions as $\alpha_{kl} = S_{kl}^\star \beta_0$ and $\alpha_k^0 = s_k^{0\star} \beta_0$. In this way we ensure the mean of these Gamma distributions equals $S_{kl}^\star$ or $s_k^{0\star}$. In the numerical studies of Section 4, we set $\beta_0 = 0.1$. Other values of β_0 and α_{kl} can also be used, and our empirical studies show that the Gibbs sampler is very robust to the hyper-parameters. This is because, in the Bayesian analysis, if the sample size N_T is big enough, the influence of the hyper-parameters can be negligible.

4 NUMERICAL STUDIES

In this section we use some numerical studies to demonstrate the feasibility and accuracy of the proposed procedure. In addition to the naive estimator $\hat{\mu}_0$ in (1), we also compare our method with another two estimators, the estimator in [9] denoted as $\hat{\mu}_1$, and the one in [8] denoted as $\hat{\mu}_2$. It is noted that both $\hat{\mu}_1$ and $\hat{\mu}_2$ assume the computing indices $\mathbf{Z}$ of all simulation outputs $\mathbf{Y}$ are known, which might be inapplicable in real examples.

4.1 Simulation validation

We first consider the parametric scenario, where Y follows the Weibull distribution with scale $a = 225.6758$ and shape $b = 2$. We are interested in estimating its mean $\mathrm{E}(Y) = 200$. We set $X_{ij} = Y_{ij}/\lambda_i$, with $\lambda_i = 1, 2/3, 1/2, 2/5, 1/3$ for 20% of the m units, respectively. We compare the performance when $m = 10$ and $m = 100$. In addition, we consider stopping times T ranging from 800

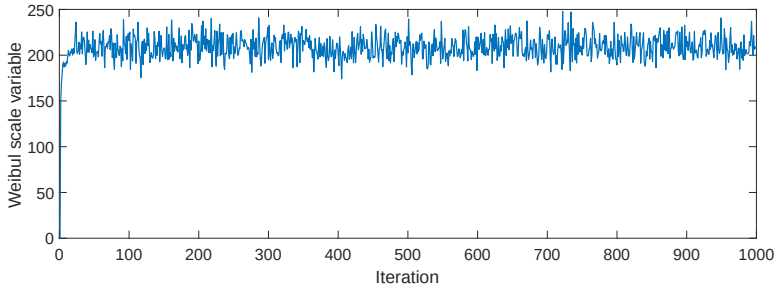


Fig. 6. The estimated Weibull scale parameter a (true value equals 225.6758) for different iterations in a Markov Chain in the simulation with $m = 10$ and $T = 800$.

to 2000[1]. For every T, we simulate 200 different macro replications of $\{Y_{1:N_T}\}$. For every replication, we assume knowing the distribution form of Y and run the Gibbs sampler particularly designed for the Weibull distribution. We generate a Markov chain with 1000 observations. We empirically evaluate its convergence rate and the required burn-in samples. For example, for $m = 10$ and $T = 800$, Figure 6 shows the estimated Weibull scale parameter in one Markov chain. We can see that the chain converges to the stationary state after the first 20 observations. The convergence rate will further increase as the sample size N_T increases. As such, for our simulations, a burn-in sample size 50 is large enough to discard the non-stationary observations. We use the remaining 950 steady-state samples to approximate the empirical distribution of $\mu = \mathrm{E}(Y)$. We use the mode of this empirical distribution, noted as $\hat{\mu}_G$, as the point estimate of μ for this macro replication. Then the estimation bias of $\hat{\mu}_G$ is calculated as the mean of the $\hat{\mu}_G$ from these 200 macro replications minus the true value. We also construct the estimators $\hat{\mu}_0$, $\hat{\mu}_1$, and $\hat{\mu}_2$ for every macro replication and calculate their corresponding bias.

The results are summarized in Figure 7a and 7b. They clearly show that $\hat{\mu}_0$ always has negative bias, and the bias is large when T is small or m is big. In contrast, the other three estimators have negligible bias throughout. However, both estimators $\hat{\mu}_1$ and $\hat{\mu}_2$ require more information on the typically unknown indices $\mathbf{Z}$, and take much longer simulation time than the budget T to obtain the outputs from the unfinished replications at T. It is recorded that for $T = 800$, the average finishing time of μ_1 is $T_1 = 1500$ for $m = 10$, and $T_1 = 2200$ for $m = 100$. This time extension $T_1 - T$ will continue increasing as T increases due to the so-called paradox ([9]). Conversely, our proposed method can reduce the bias significantly at no expense of either extra computing time or knowing $\mathbf{Z}$.

Furthermore, as shown in Figures 7a and 7b, for the same T, when m is larger, $\hat{\mu}_G$ performs better in terms of a smaller estimation variance. This is because that a larger m leads to a larger N_T, and therefore a lower estimation variance. We have done extra simulations to validate this point. In particular, we fix $T = 1000$, and increase m from 10 to 100. For every value of m, we do simulation in the same way as above by assuming Y_n follows Weibull(225.676, 2). Then we calculate

[1]It should be noted that for all the simulations in this section, they do not require a real parallel computing environment, since the simulation of every replication output does not necessarily take real time. In other words, assume the simulation output follows a certain distribution, such as the Weibull distribution. We "simulate" the outputs sequentially by randomly generating samples Y_{ij}, $j = 1, 2, \ldots$ from the Weibull distribution sequentially, for every "virtual" computing unit i. The random sample generation is so fast that the simulation does not take any time. However, to mimic simulations in reality, we assume the "real" computation time for every sample of every unit is $X_{ij} = Y_{ij}/\lambda_{ij}$. Since the "real" computation time is not real, there is no real time unit here.

the estimation bias of $\hat{\mu}_G$ together with its standard deviation as shown in Table 1. We can see that the bias almost equals 0 for all m consistently, demonstrating the efficiency of the proposed bias correction algorithm. Furthermore, as m increases, the estimation variance decreases, as discussed above.

Table 1. PHD approximation results with $T = 1000$ and various m.

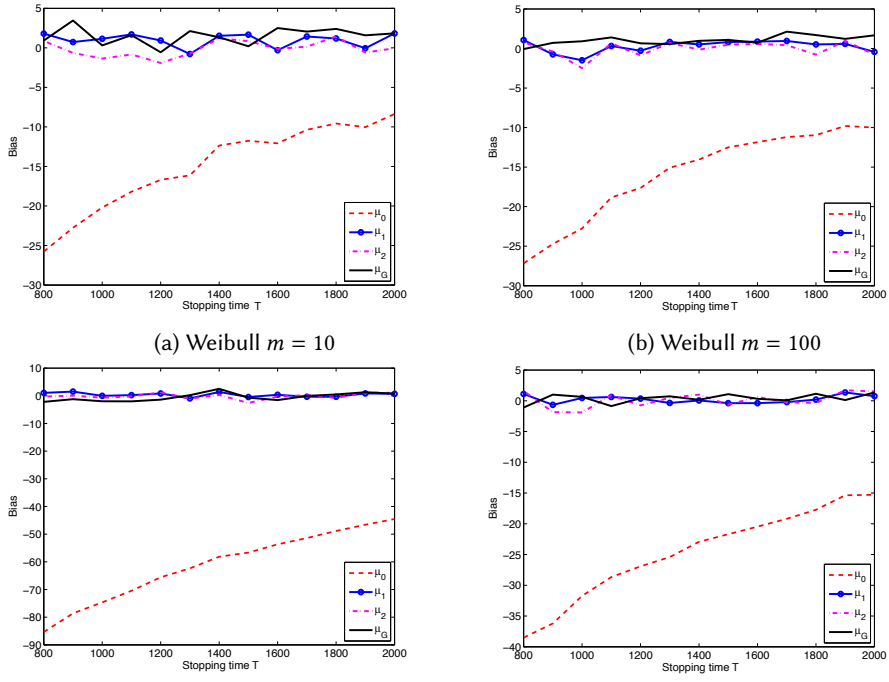
m	10	25	40	55	70	85	100
bias	1.11	-0.98	-0.41	0.49	0.52	0.66	0.25
Std	20.3	13.5	9.89	8.92	8.07	7.42	6.31

We also consider the nonparametric scenario with PHD approximation. We use two new distributions of Y for demonstration. One is the third order hyperexponential distribution with probability parameters $\mathbf{p} = (1/4, 1/4, 1/2)$ and rate parameters $\boldsymbol{\lambda} = (1/150, 1/150, 1/250)$. The other is the Erlang distribution with parameters $(3, 0.015)$. Both distributions have mean 200. We assume the distribution form of Y is unknown, and apply the nonparametric estimation procedure. In particular, for both distributions, we first use the Gibbs sampler for $\pi(\Theta, J_{1:N_T}|Y_{1:N_T})$ to decide the PHD order p, which is rightly 3. Then we test the proposed estimation procedure following the same steps as the last paragraph. From the results in Figure 7c and 7d, we can see the performance is similar as that for Weibull distribution. which again demonstrates the satisfactory performance of our method.

Now we further show that even though in some cases the moment matching algorithm may not provide an accurate (or optimal) choice of p, as long as p does not deviate from the true (or optimal) p too much, this incorrect p will not affect the performance of the proposed procedure too much. In other words, when the chosen p is a bit larger the true one, our algorithm is still numerical feasible and stable. When the chosen p is a bit smaller than the true one, our algorithm can still achieve a reasonable result. We demonstrate this point by fitting the previous HypExp $(\mathbf{a}, \mathbf{b})$ distribution with 2-order and 4-order PHD and compare their bias and rooted mean square error (RMSE) in Table 2. We still simulate 200 different macro replications of $\{Y_{1:N_T}\}$. The bias is calculated in the same way as previously, and the RMSE of $\hat{\mu}_G$ is calculated as the sample standard deviation of the 200 $\hat{\mu}_G$. We can see that with no surprise the 3-order PHD has the smallest bias and RMSE. While for the 2-order PHD, though its bias is slightly bigger than the other two due to its oversimplified model assumption, the result is still reasonable and acceptable. On the contrary, the overparameterized 4-order PHD can achieve a comparable small bias as the 3-order one, but the former has a slightly bigger RMSE than the later. To further illustrate the quality of a PHD approximation, we also fit the previous Erlang distribution with a 2-order PHD, a 3-order PHD and a 4-order PHD, and compare their empirical probability densities with the true one as Figure 8 shows. Every empirical density is plotted based on totally 950×200 PHD observations. In particular, for a macro replication, for every $\Theta^{(k)}$ drawn from the Gibbs sampler after the burn-in period, We draw one PHD observation from its PHD$(\cdot|\Theta^{(k)})$. In this way we totally have 950 samples for one macro replication, and we combine all the samples from the 200 macro replications together. We can see that consistently with Table 2, the 2-order PHD fails to describe the true density to some degree, while the 3-order PHD and the 4-order PHD approximate the true density quite well.

5 EMPIRICAL EVALUATION

In this section, we use some empirical studies from queuing system and control chart performance simulation to test the implementability of the proposed procedure.



(a) Weibull $m = 10$ (b) Weibull $m = 100$

(c) HypExp $m = 100$ (d) Erlang $m = 100$

Fig. 7. Estimation bias of the mean of Weibull(225.676, 2) with (a) $m = 10$ and (b) $m = 100$, (c) HypExp($\mathbf{p}, \boldsymbol{\lambda}$) with $m = 100$, and (d) Erlang(3, 0.015) with $m = 100$.

Table 2. PHD approximation with different orders for the hyperexponential distribution

	PHD_2		PHD_3		PHD_4	
t	Bias	RMSE	Bias	RMSE	Bias	RMSE
800	3.93	5.19	-1.10	6.62	-1.67	7.28
1200	3.46	5.10	0.41	6.01	-0.45	6.70
1600	3.16	4.85	0.35	4.57	0.44	5.36
2000	3.04	4.25	0.30	4.27	0.39	5.24

5.1 $M/M/1$ queue overflow simulation

Simulation plays an important role in queuing system analysis. Here we consider a simple case, analyzing the overflow time Y of a $M/M/1$ queuing system, for illustration. In particular, the $M/M/1$ queue has an arrival rate 1, and a departure rate 1.1. Defining the overflow as $Q > 20$ where Q is the queue length with initial value 0, we use simulation to estimate the average time to overflow, i.e., E(Y). In particular, we use the Dell Precision Tower T7910 as the parallel computing environment. The workstation has dual Intel Xeon E5-2670 processors. Every processor has 16 cores, with totally 32 cores available. Among them $m = 30$ cores are used as different computing

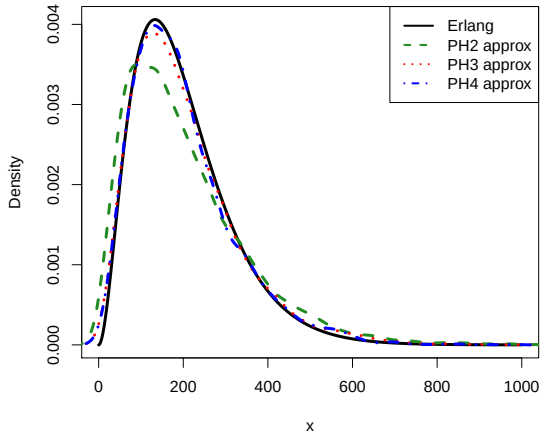
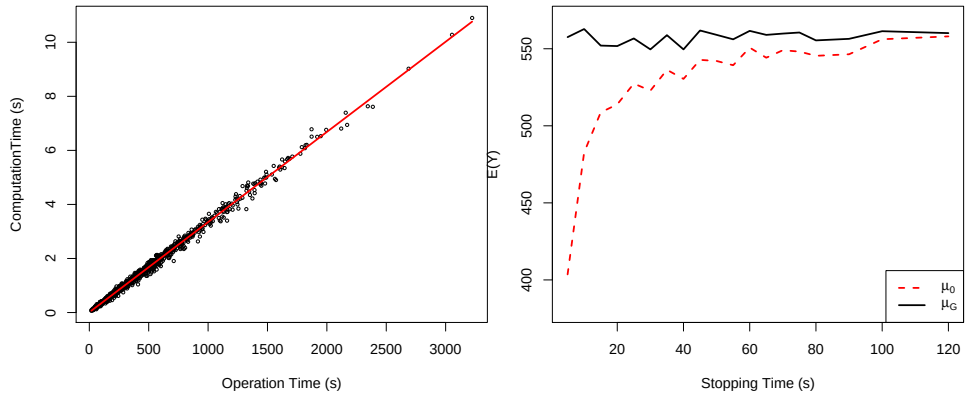


Fig. 8. PHD approximation with different orders for Erlang (3,0.015).

units to run the simulation in parallel[2]. Pilot runs show that all units have the same speed, with the fitted relation $X_{ij} = 0.0033Y_{ij}$ for all computing units as Figure 9a shows. Estimation using the completed Y suggests a 2-order PHD approximation. Then we calculate $\hat{\mu}_G$ according to the proposed nonparametric method in Section 3.2 and 3.3.2, and compare it with the naive estimator $\hat{\mu}_0$. Figure 9b shows their results at different stopping times up to $T = 120$. We can see that $\hat{\mu}_0$ fails to provide a reliable estimate when T is small, while $\hat{\mu}_G$ has negligible bias regardless of T. It is to be noted that since the index information $\mathbf{Z}$ is not known in this and the next example, $\hat{\mu}_1$ and $\hat{\mu}_2$ are not applicable.



(a) $X \sim Y$ (b) $\mathrm{E}(Y)$ estimation

Fig. 9. (a) The relation between overflow time Y and computing time X; (b) Average overflow time, $\mathrm{E}(Y)$, estimation using different methods.

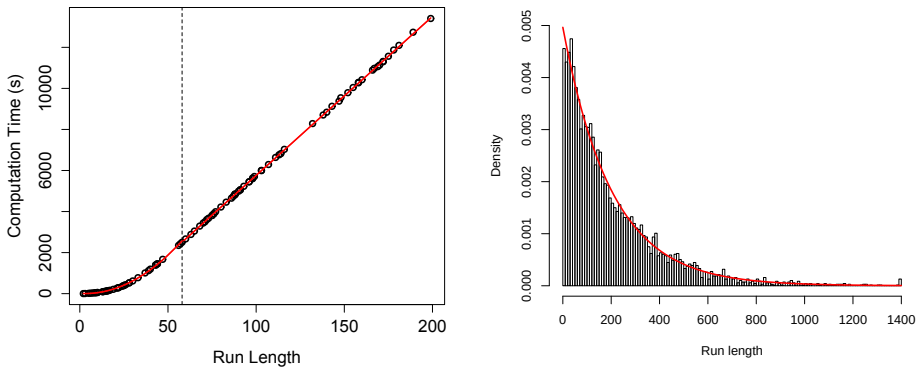
[2]It should be noted in this parallel computing environment, we can actually fetch the index information for every simulation output. However, in our simulation context, we assume this information unknown.

5.2 Control chart simulation

We also apply our method to estimate the average run length (ARL) of the control chart in [22] through simulation. In this example, we use the high-performance computing (HPC) resource of National University of Singapore as the parallel computing environment. Every user account can submit limited simulation tasks to the HPC and the HPC randomly allocates idle computation cores (units) for the user. Due to limited authorities and confidential information, users don't know the exact information of every computation core. We use $m = 172$ computing units with $T = 315$ hours. All run length outputs Y are recorded in a single file in the order of their completion instants. Figure 10a shows that for $Y \leq 58$, X is quadratic in Y, while for $Y > 58$, X is linear in Y. This piecewise relation is consistent with its original logic in [22]. Therefore, we use the following model for such relation

$$X_{ij} = \begin{cases} a_{i1}Y_{ij}^2 + a_{i2}Y_{ij} + a_{i3}, & \text{for } Y_{ij} \leq 58 \\ b_{i1}Y_{ij} + b_{i2}, & \text{for } Y_{ij} > 58 \end{cases}, \tag{8}$$

where the model parameters are unit dependent. The fitting results (red line in Figure 10a) show that (8) is satisfactory.



(a) $X \sim Y$ (b) Histgram of Y

Fig. 10. (a) The relation between run length Y and computing time X; (b) The histogram of Y.

We calculate the ARL using $\hat{\mu}_G$ at different T up to 315 hours, and compare $\hat{\mu}_G$ with $\hat{\mu}_0$ in Figure 11. In addition, since the output Y is exponentially distributed as Figure 10b shows, we can approximate the relation between X and Y to be a linear one and use the analytical formula of Proposition (2.1) to obtain a bias-adjusted estimator, denoted by $\hat{\mu}_A$. To be more specific, in Proposition 2.1 p_i can be derived from the estimated linear relation for every unit. τ is the expected number of finished replications up to T and can be approximated by N_T. Then we can calculate the bias expansion for every T based on Proposition 2.1 and adjust the estimator correspondingly. Figure 11 demonstrates that both methods can reduce the bias significantly, and are much superior than the naive estimator.

6 CONCLUDING REMARKS

For transient simulation using parallel computing with a time budget, when the simulation time depends on the output value of the replication, most popular unbiased estimators compromise and

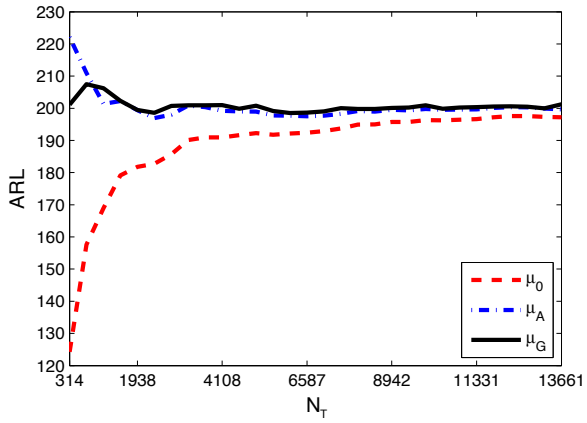


Fig. 11. ARL, $\mathrm{E}(Y)$, estimation using different methods.

lose their statistical accuracy. This paper proposes a new estimator in the Bayesian framework to reduce the bias. Numerical studies together with two case examples illustrate that the proposed procedure can achieve satisfactory estimation results under mild restrictions.

Along this research direction, there are still some valuable extensions. One is to extend the current framework into steady-state simulation to rectify its initial bias. Another is to combine the current framework with ranking-and-selection procedures. Since the Bayesian analysis not only provides the point estimation, but also the posterior distribution of $\mathrm{E}(Y)$, this extra information is expected to provide a better evaluation of the subsystems in the ranking-and-selection problems. Furthermore, we may also consider developing a nonparametric Gibbs sampler to estimate $\mathrm{E}(Y)$ directly using some nonparametric Bayesian methods in survival analysis without impressing any distribution assumption on Y. For example, we may assume the hazard rate of Y as a martingale jump process and use it to express the PDF and the survival function of Y.

APPENDICES

A-I Proof of Proposition 2.1

As previously shown,

$$\begin{aligned}
\mathrm{E}\,\hat{\mu}_0 &= \sum_{k_1=0}^{\infty}\cdots\sum_{k_m=0}^{\infty}\left(\int_0^T\cdots\int_0^T \frac{\lambda_1 x_1+\cdots+\lambda_m x_m}{k_1+\cdots+k_m}\prod_{i=1}^{m}\frac{k_i x_i^{k_i-1}}{t^{k_i}}\mathrm{d}x_1\cdots\mathrm{d}x_m\right)\cdot\prod_{i=1}^{m}\mathrm{P}(N_i=k_i)\\
&= \sum_{k_1=0}^{\infty}\cdots\sum_{k_m=0}^{\infty}\left(\frac{\sum_{i=1}^m \lambda_i k_i/(k_i+1)}{\sum_{i=1}^m k_i}T\right)\cdot\prod_{i=1}^{m}\frac{\exp(-\frac{\lambda_i}{\mu}T)(\frac{\lambda_i}{\mu}T)^{k_i}}{k_i!}\\
&= \mathrm{E}_{N_T}\frac{T}{N_T}\,\mathrm{E}_{N_1,\cdots,N_m|N_T}\sum_{i=1}^{m}\frac{1}{\theta_i}\frac{N_i}{N_i+1}, \qquad N_T\geq 1 \qquad \text{(A.1)}
\end{aligned}$$

According to the properties of the Poisson distribution, N_T follows the Poisson distribution with rate $\tau=\sum_{i=1}^m\frac{\lambda_i}{\mu}T$. In addition, the conditional distribution of N_i given N_T follows multinomial distribution with parameters $(N_T, p_i, i=1,\cdots,m)$, where $p_i=\lambda_i/\sum_{j=1}^m\lambda_j$. To compute (A.1), we first provide the following useful lemma.

Lemma 6.1. *If X follows Binomial distribution with parameter (n,p), then*

$$\mathrm{E}\,\frac{X}{X+1} = 1 - \frac{1-(1-p)^{n+1}}{(n+1)p} \xrightarrow[np\to\lambda]{n\to\infty} 1 - \frac{1-\exp(-\lambda)}{\lambda}$$

Proof of Lemma 6.1. Since $X \sim \text{Binomial}(n,p)$, it is easy to note that

$$\begin{aligned}
\mathrm{E}\,\frac{X}{X+1} &= \sum_{k=0}^{n} \frac{k}{k+1}\cdot\frac{n!}{k!(n-k)!}p^k(1-p)^{n-k} && \text{(A.2)}\\
&= \frac{1}{(n+1)p}\sum_{k=0}^{n} k\cdot\frac{(n+1)!}{(k+1)!(n-k)!}p^{k+1}(1-p)^{n-k} && \text{(A.3)}\\
&= \frac{1}{(n+1)p}\sum_{j=1}^{n+1}(j-1)\cdot\frac{(n+1)!}{j!(n+1-j)!}p^{j}(1-p)^{n+1-j} && \text{(A.4)}\\
&= \frac{1}{(n+1)p}\left[\mathrm{E}(Y-1)+(1-p)^{n+1}\right], \quad \text{where } Y\sim\text{Binomial}(n+1,p) && \text{(A.5)}\\
&= 1-\frac{1-(1-p)^{n+1}}{(n+1)p}. && \text{(A.6)}
\end{aligned}$$

In addition, when $n\to\infty, p\to 0$, and $np\to\lambda$, we have

$$\begin{aligned}
\lim_{n\to\infty,np\to\lambda}\mathrm{E}\,\frac{X}{X+1} &= \lim_{n\to\infty,np\to\lambda}\left[1-\frac{1-(1-p)^{n+1}}{(n+1)p}\right] && \text{(A.7)}\\
&= \lim_{n\to\infty,np\to\lambda} 1-\frac{1}{\lambda}\left[1-\left(1-\frac{(n+1)p}{(n+1)}\right)^{n+1}\right] && \text{(A.8)}\\
&= 1-\frac{1-\exp(-\lambda)}{\lambda}. \qquad \square && \text{(A.9)}
\end{aligned}$$

$\square$

According to Lemma 6.1, $\mathrm{E}\,\hat{\mu}_0$ essentially reduces to

$$\begin{aligned}
\mathrm{E}\,\hat{\mu}_0 &= \mathrm{E}_{N_T}\frac{T}{N_T}\sum_{i=1}^{m}\lambda_i\left[1-\frac{1-(1-p_i)^{N_T+1}}{(N_T+1)p_i}\right]\\
&= \sum_{i=1}^{m}\lambda_i T\,\mathrm{E}_{N_T}\left[\frac{1}{N_T}-\frac{1-(1-p_i)^{N_T+1}}{N_T(N_T+1)p_i}\right], \quad N_T\geq 1\\
&= \mu\tau\left[\mathrm{E}_{N_T}\frac{1}{N_T}-\sum_{i=1}^{m}\mathrm{E}_{N_T}\frac{1-(1-p_i)^{N_T+1}}{N_T(N_T+1)}\right]\\
&= \mu\tau\left[\mathrm{E}_{N_T}\frac{1-m+\sum_{i=1}^{m}(1-p_i)^{N_T+1}}{N_T}+\mathrm{E}_{N_T}\frac{m-\sum_{i=1}^{m}(1-p_i)^{N_T+1}}{N_T+1}\right] \qquad \text{(A.10)}
\end{aligned}$$

(A.10) can be further simplified using the following lemma.

Lemma 6.2. *If N follows the Poisson distribution with mean parameter λ, then*

$$\mathrm{E}(\frac{1}{N}|N\geq 1)=\frac{\exp(-\lambda)}{1-\exp(-\lambda)}\int_0^1\frac{e^{\lambda x}-1}{x}\mathrm{d}x, \quad \mathrm{E}(\frac{p^N}{N}|N\geq 1)=\frac{\exp(-\lambda)}{1-\exp(-\lambda)}\int_0^1\frac{e^{p\lambda x}-1}{x}\mathrm{d}x.$$

Proof of Lemma 6.2. Since N follows Poisson distribution, we have

$$\mathrm{E}\frac{p^N}{N} = \sum_{n=1}^{\infty}\frac{e^{-\lambda}\lambda^n}{(1-e^{-\lambda})n!}\cdot\frac{p^n}{n} = \frac{e^{-\lambda}}{1-e^{-\lambda}}\sum_{n=1}^{\infty}\frac{(p\lambda)^n}{n!\cdot n} \tag{A.11}$$

$$= \frac{e^{-\lambda}}{1-e^{-\lambda}}\sum_{n=1}^{\infty}\int_0^{p\lambda}\frac{u^{n-1}}{n!}\mathrm{d}u = \frac{e^{-\lambda}}{1-e^{-\lambda}}\int_0^{p\lambda}\sum_{n=1}^{\infty}\frac{u^{n-1}}{n!}\mathrm{d}u \tag{A.12}$$

$$= \frac{e^{-\lambda}}{1-e^{-\lambda}}\int_0^{p\lambda}\frac{1}{u}\left(\sum_{n=1}^{\infty}\frac{u^n}{n!}\right)\mathrm{d}u = \frac{e^{-\lambda}}{1-e^{-\lambda}}\int_0^{p\lambda}\frac{e^u-1}{u}\mathrm{d}u \tag{A.13}$$

$$= \frac{e^{-\lambda}}{1-e^{-\lambda}}\int_0^{1}\frac{e^{p\lambda u}-1}{u}\mathrm{d}u. \tag{A.14}$$

The sequence of infinite summation and integral can be interchanged in the second line, because the series $\sum_{n=1}^{\infty}u^n/n!$ is uniformly convergent for finite u. The last equality can be obtained by changes of variables in the integral. When $p = 1$, the result degenerates to the special case of inverse moment of Poisson distribution. Even though the integral is not analytically computable, it can be easily calculated using numerical integrations. □ □

Therefore finally we have

$$\mathrm{E}\,\hat{\mu}_0 = \mu\left\{\tau\frac{e^{-\tau}}{1-e^{-\tau}}\left[(1-m)\int_0^1\frac{e^{\tau x}-1}{x}\mathrm{d}x + \sum_{i=1}^{m}(1-p_i)\int_0^1\frac{e^{\tau(1-p_i)x}-1}{x}\mathrm{d}x\right]\right.$$
$$\left. + \frac{1}{1-e^{-\tau}}\left[m - e^{-\tau}\tau - \sum_{i=1}^{m}e^{-\tau p_i}\right]\right\}, \tag{A.15}$$

which is the exact form of Proposition 2.1.

A-II Sampling $\pi(\Theta|J_{1:N_t}, J'_{1:m})$

The full likelihood function for $J_{1:N_t}$ and $J'_{1:m}$ is

$$p(J_{1:N}, J'_{1:m}|\mathrm{S}, \boldsymbol{\pi}_0) = \prod_{k=1}^{p}\pi_{k0}^{b_k}\prod_{k=1}^{p}\exp(S_{kk}r_k)\prod_{k=1}^{p}\prod_{l=0,l\neq k}^{p}S_{kl}^{N_{kl}} \tag{A.16}$$

where $S_{k0} = s_k^0$, b_k is the number of jump processes with starting state equal to k, r_k is the total time spent in state k, N_{kl} is the total number of state moves from k to l, and N_{k0} is the total number of absorbing moves from state k to state 0 across all the paths $\{J_{1:N_t}, J'_{1:m}\}$. According to $S_{kk} = -\sum_{l\neq k}S_{kl} - s_k^0$, we can rewrite (A.16) as

$$p(J_{1:N_t}, J'_{1:m}|\mathrm{S}, \boldsymbol{\pi}_0) = \prod_{k=1}^{p}\pi_{k0}^{b_k}\prod_{k=1}^{p}\exp(-s_k^0 r_k)\prod_{k=1}^{p}\prod_{l=0,l\neq k}^{p}S_{kl}^{N_{kl}}\exp(-S_{kl}r_l). \tag{A.17}$$

This forms a product of p^2 kernels of Gamma distributions and a kernel of Dirichlet distribution. Consequently, if we choose independent conjugate priors for each parameter, we will get (7).

REFERENCES

[1] Argon, N. T., Andradóttir, S., Alexopoulos, C., and Goldsman, D. (2013), "Steady-State Simulation with Replication-Dependent Initial Transients: Analysis and Examples," *INFORMS Journal on Computing*, 25, 177–191.

[2] Bladt, M., Gonzalez, A., and Lauritzen, S. L. (2003), "The estimation of Phase-type related functionals using Markov chain Monte Carlo methods," *Scandinavian Actuarial Journal*, 2003, 280–300.

[3] Brook, D. and Evans, D. (1972), "An approach to the probability distribution of CUSUM run length," *Biometrika*, 59, 539–549.
[4] Bux, W. and Herzog, U. (1977), "The phase concept: Approximation of measured data and performance analysis," *Computer Performance*, 23–38.
[5] Crane, M. A. and Iglehart, D. L. (1975), "Simulating stable stochastic systems: III. Regenerative processes and discrete-event simulations," *Operations Research*, 23, 33–45.
[6] Esparza, L. J. R., Nielsen, B. F., and Bladt, M. (2010), "Maximum likelihood estimation of phase-type distributions," Ph.D. thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Applied Mathematics and Computer ScienceInstitut for Matematik og Computer Science.
[7] Glynn, P. W. and Heidelberger, P. (1990), "Bias properties of budget constrained simulations," *Operations Research*, 38, 801–814.
[8] — (1991), "Analysis of parallel replicated simulations under a completion time constraint," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 1, 3–23.
[9] Heidelberger, P. (1988), "Discrete event simulations and parallel processing: statistical properties," *SIAM Journal on Scientific and Statistical Computing*, 9, 1114–1132.
[10] Henderson, S. G. and Glynn, P. W. (1999), "Can the regenerative method be applied to discrete-event simulation?" in *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, ACM, pp. 367–373.
[11] — (2001), "Regenerative steady-state simulation of discrete-event systems," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 11, 313–345.
[12] Kelton, W. D. and Law, A. M. (2006), *Simulation modeling and analysis*, McGraw Hill Boston.
[13] Law, A. M. (2007), "Statistical analysis of simulation output data: the practical state of the art," in *Simulation Conference, 2007 Winter*, IEEE, pp. 77–83.
[14] Lawrence, A. (1973), "Dependency of intervals between events in superposition processes," *Journal of the Royal Statistical Society. Series B (Methodological)*, 306–315.
[15] Luo, J., Hong, L. J., Nelson, B. L., and Wu, Y. (2015), "Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments," *Operations Research*, 63, 1177–1194.
[16] Mengersen, K. L., Tweedie, R. L., et al. (1996), "Rates of convergence of the Hastings and Metropolis algorithms," *The annals of Statistics*, 24, 101–121.
[17] Ni, E. C., Ciocan, D. F., Henderson, S. G., and Hunter, S. R. (2015), "Efficient Ranking and Selection in Parallel Computing Environments," *arXiv preprint arXiv:1506.04986*.
[18] Ni, E. C., Hunter, S. R., and Henderson, S. G. (2013), "Ranking and selection in a high performance computing environment," in *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, IEEE Press, pp. 833–845.
[19] O'cinneide, C. A. (1999), "Phase-type distributions: open problems and a few properties," *Stochastic Models*, 15, 731–757.
[20] Prentice, R. L., Kalbfleisch, J. D., Peterson Jr, A. V., Flournoy, N., Farewell, V., and Breslow, N. (1978), "The analysis of failure times in the presence of competing risks," *Biometrics*, 541–554.
[21] Robinson, P. and Ho, T. Y. (1978), "Average run lengths of geometric moving average charts by numerical methods," *Technometrics*, 20, 85–93.
[22] Zhang, C., Chen, N., and Zou, C. (2016), "Robust multivariate control chart based on goodness-of-fit test," *Journal of Quality Technology*, 48, 139.