Efficient and Reliable Computation of Birth-Death Process Performance Measures

Armann Ingolfsson
School of Business, University of Alberta, Edmonton, Alberta T6G 2R6, Canada,
armann.ingolfsson@ualberta.ca

Ling Tang
Industrial Engineering and Operations Research Department, Columbia University, New York, NY 10027,
lt2326@columbia.edu

We present an efficient, reliable, and easy-to-implement algorithm to compute steady-state probabilities for birth-death processes whose upper-tail probabilities decay geometrically or faster. The algorithm can provide any required accuracy and it avoids over- and underflow. In addition to steady-state probabilities, the algorithm can compute any performance measure that can be expressed as the expected value of a function of the population size, for non-negative functions that are bounded by a constant, linear, or quadratic function of population size. The algorithm works with conditional steady state probabilities, given that the population is in a range that is extended up and down as the algorithm progresses. These conditional probabilities facilitate the derivation of truncation error bounds. We illustrate the application of the algorithm to the Erlang B, C, and A queueing systems.

Key words: Queues, Algorithms; Queues, Birth-Death; Erlang B; Erlang C; Erlang A.
History: Accepted by Winfried K. Grassmann, Area Editor for Computational Probability & Analysis; received September 2009, accepted September 2010.

1. Introduction

Birth-death processes are among the most studied and used stochastic models in operations resarch. Such processes are the foundation of queueing theory, encompassing the $M/M/s$ (Erlang C), $M/M/s/s$ (Erlang B), $M/M/\infty$ (infinite server), and finite source queueing models. Additional applications are found in biology, genetics, and epidemiology (see Novozhilov et al. (2006) for a recent survey of biological applications). Birth-death processes are highly tractable compared to more general continuous-time Markov chains. The theoretical properties of birth-death processes have been studied extensively, for example see Karlin and McGregor (1957) and van Doorn (1980). The steady state probabilities for ergodic birth-death processes are easily expressed in closed-form and these expressions are well known.

Nevertheless, as several authors have pointed out (Garnett et al., 2002; Whitt, 2005) developing reliable algorithms to compute birth-death process performance measures requires care. These numerical difficulties have been cited (Garnett et al., 2002) as one of the reasons for developing and using asymptotic approximations to birth-death queueing models. A model that has recently generated renewed interest in birth-death process computations is the $M/M/s + M$ model, also called the Erlang A model, which adds customer reneging (or abandonment) from queue after an exponentially distributed "patience time" to the $M/M/s$ model. The importance of accounting for abandonment in such settings as telephone call centers has prompted suggestions (Gans et al., 2003; Mandelbaum and Zeltyn, 2007) that Erlang A should replace Erlang C as the "standard model" for call center staffing calculations.

In view of their widespread usage, it is important to have simple, efficient, and reliable algorithms to compute performance measures for birth-death processes, to test these algorithms, to make them widely known, and to implement them with easy-to-use software. Computations for the exact Erlang A model are more demanding than for asymptotic approximations of the model, as pointed out in Mandelbaum and Zeltyn (2007). However, for most systems of practical interest, the exact computations can be done essentially instantaneously. Therefore, we contend, the underlying computational complexity is less important than making the algorithms widely available and easy to use, or in other words, to improve Sam Savage's (1997) "keystroke metric" of the complexity of applying an analytical technique. Given that spreadsheets are probably the most commonly used computational platform today, implementing the algorithms as spreadsheet functions seems particularly useful—at least if the functions can be evaluated quickly, as is the case for the algorithms proposed in this paper. By spreadsheet functions we mean formulas that can be typed in spreadsheet cells to compute an output of interest, with the computations performed by an add-in program. The Queueing ToolPak (a spreadsheet add-in developed by Ingolfsson and Gallop, 2003) provides one example of such software, which makes the computation of performance measures for $M/M/s$ and $M/M/s/s+r$ queueing models as easy as computing areas under a normal distribution, or any other calculation that can be done with a built-in spreadsheet function. The development of the algorithm presented in this paper began with modifications, based on user feedback, to increase the reliability of the algorithms used in the Queueing ToolPak.

Although we question the need to use asymptotic approximations to compute numerical

values of birth-death process performance measures, we recognize the value of asymptotic analysis for providing insight about system behavior. Indeed, the parameter values we use in the computational examples later in the paper are motivated by the results of asymptotic many-server analysis.

Our contribution is an algorithm to compute steady state performance measures for a class of birth-death processes that includes many birth-death models of practical interest. The essential condition is that the process must reach steady state and the steady state probabilities must decay geometrically or faster as the population size goes to infinity. The algorithm computes performance measures that can be expressed as the expected value of a non-negative function of the steady state population size. We truncate the state space from below and above, and we provide error bounds on the relative error resulting from the truncation, for performance measures whose associated function can be bounded by a constant, a linear function, or a quadratic function of the state variable. The algorithm avoids problems with over- and underflow and the error from truncation can be made as small as desired. The algorithm is easy to implement, uses neither special functions nor numerical integration, and works regardless of whether the state space is finite or infinite.

The remainder of the paper is organized as follows. Section 2 defines notation and describes previous approaches; Section 3 presents our algorithm; Section 4 provides absolute and relative error bounds; Sections 5 to 7 illustrate the use of the algorithm to compute representative performance measures for Erlang B, C, and A systems; and Section 8 concludes. Appendix A contains derivations of absolute error bounds, Appendix B has a proof of a formula used to compute the distribution of virtual waiting times for the Erlang A model, and an online supplement provides pseudo code for previous algorithms and Matlab computer code for the algorithms used in Sections 5 to 7.

2. Previous Algorithms

A continuous-time birth-death process $N(t)$ has state space $\{0, 1, \dots\}$, birth rates λ_n for $n \geq 0$ (for $n \to n+1$ transitions), and death rates μ_n for $n \geq 1$ (for $n \to n-1$ transitions). We will view a finite state space $\{0, \dots, K\}$ as a special case, with $\lambda_n = 0$ for $n \geq K$ and $\mu_n = 0$ for $n > K$. This allows us to treat finite and infinite state space processes together. We assume that the process reaches steady state, with random variable N representing the population size (or number in system, in a queueing context) in steady state, with

distribution $\pi_n = \Pr\{N = n\} = \lim_{t\to\infty} \Pr\{N(t) = n\}$. When we discuss $M/M/\cdot$ queueing systems, we use λ to denote the arrival rate, μ to denote the service rate, $r = \lambda/\mu$ to denote the offered load, s to denote the number of servers, and for the Erlang A model, γ to denote the abandonment rate per waiting customer.

The local balance equations ($\lambda_n\pi_n = \mu_{n+1}\pi_{n+1},\ n = 0, 1, \dots$) and a normalization equation ($\sum_0^\infty \pi_n = 1$) determine the steady state probabilities for N. We begin by describing three standard and increasingly sophisticated algorithms for solving these equations and potential numerical difficulties. All of these algorithms have been proposed before and we discuss each of them in turn. Figures 1-3 in the online supplement show pseudo code for the three algorithms.

Algorithm 1 begins by computing π_0 and then computes π_1, π_2, etc., up to some truncation limit u, using the recursion $\pi_{n+1} = b_n\pi_n$ (where $b_n = \lambda_n/\mu_{n+1}$), which is obtained from the local balance equations. The difficulty with this algorithm is that computing π_0 often requires as much effort as computing all of the other steady state probabilities. In general, $\pi_0 = \left(1 + \sum_{n=1}^\infty \prod_{i=0}^{n-1} \lambda_i/\mu_{i+1}\right)^{-1}$. In some special cases, one can derive a simple closed-form solution for π_0, notably for $M/M/1$ and $M/M/\infty$ queues, but even in these simple cases, there is a potential problem—π_0 may evaluate to zero because of underflow. As Tijms (1994) points out, one can avoid underflow by taking logarithms of the expression for π_0 and the recursion $\pi_{n+1} = b_n\pi_n$, but this is only of value if an easily computed expression for π_0 is available. As an example, the $M/M/s$ queue has no expression for π_0 that is easy to compute before the probabilities $\pi_1, \dots, \pi_s$ have been computed.

To describe Algorithms 2 and 3, we will use the notation $\bar{\pi}_n$ for quantities that are not probabilities but will eventually be transformed into probabilities. Algorithm 2 begins by setting $\bar{\pi}_0$ equal to one, thus avoiding the potential for underflow at the beginning of the algorithm and the difficulty with computing π_0 before the other steady state probabilities have been computed. Then, after using $\bar{\pi}_{n+1} = b_n\bar{\pi}_n$ to compute $\bar{\pi}_1, \bar{\pi}_2, \dots$ up to some truncation limit u, the $\bar{\pi}_n$ are normalized to convert them into probabilities π_n. One difficulty with this algorithm is that if π_0 is much smaller than the largest state probability, then overflow may occur. A simple test case is an $M/M/s$ queue with service rate $\mu = 1$ and arrival rate $\lambda = s - 1$, which implies that $s - 1$ and $s - 2$ are the modes of the steady-state distribution. If $M/M/s$ software fails for $s > 715$ then that could be because the software employs Algorithm 2. By using bounds associated with Stirling's approximation, it is straightforward to verify that $\pi_{s-1}/\pi_0 > \exp\{s - 1 - (12(s-1))^{-1} - 0.5\ln[2\pi(s-1)]\}$,

which for $s = 714$ evaluates to 6.7^{307}. Increasing s to 715 brings the lower bound on π_{s-1}/π_0 above the double-precision overflow limit of 10^{308} and causes Algorithm 2 to fail. Thus, Algorithm 2 cannot be used for $M/M/s$ computations when the number of servers is, say, 1,000, as it could easily be in a modern call center.

Another difficulty with Algorithm 2 is that it is not always clear how to choose the upper truncation limit. When the upper-tail probabilities decay geometrically above some limit (such as the $M/M/s$ queue, where the limit is s), then the truncation limit can be chosen at that limit, and the sum of the probabilities above the limit can be computed as a geometric series, thus avoiding any truncation error. A similar approach can be used if the tail is approximately geometric, as discussed in Tijms (1994, Section 2.3.2). However, this approach does not work when the tail probabilities decay *faster* than geometrically, as they do, for example, for the Erlang A queue.

Algorithm 3, instead of starting at state zero, begins at state c (the "center") by setting $\bar{\pi}_c = 1$. For an $M/M/s$ queue, $c = s$ is a convenient choice. Then, one uses $\bar{\pi}_{n+1} = b_n \bar{\pi}_n$ to compute probabilities of states above c recursively up to a truncation limit $c + u$ and one uses $\bar{\pi}_{n-1} = a_n \bar{\pi}_n$ (where $a_n = \mu_n/\lambda_{n-1}$) to compute probabilities of states below c, down to a lower truncation limit $c - l$ (possibly equal to zero). When the truncation limits have been reached, the $\bar{\pi}_n$ are normalized. Algorithm 3 makes overflow less likely, but if c is poorly chosen, so that some state above or below c has much higher probability than π_c, then overflow can still occur. Smith (2002) proposes using this algorithm and Whitt (2005) uses it for a generalization of the Erlang A model. An example where this algorithm results in overflow is an Erlang A model with $c = s = 10,000$, $\mu = 1$, $\lambda = 5,000$, and $\gamma = 1$. In this case, $\pi_{\lambda/\mu} = \pi_{5,000} \gg \pi_{10,000} = \pi_s$. The other main deficiency of this algorithm is that it is not clear how to choose the lower and upper truncation limits. In the next section, we present our algorithm, which overcomes both of these deficiencies.

3. Algorithm

We are interested in reliably and efficiently computing common performance measures for systems that are modeled as birth-death processes. Many performance measures can be expressed as the steady state expected value $\mathrm{E}[f(N)] = \sum_{n=0}^{\infty} f(n)\pi_n$ of an appropriately defined non-negative function f. We focus on functions that can be bounded by a constant, a linear function, or a quadratic function but our approach can be extended to functions that

are bounded by higher-degree polynomials. Examples include individual state probabilities ($f(n) = \mathbf{1}\{n = m\}$ to compute π_m), average number in a queueing system ($f(n) = n$), the second moment of the average number in a queueing system ($f(n) = n^2$), and virtual waiting time distributions for queueing models ($f(n) = \Pr\{W_n > t\}$ where W_n is the virtual waiting time for a test customer that arrives when there are n customers in the system). We will see an expression for $\Pr\{W_n > t\}$ for an Erlang A system in Section 7.

We assume that an easily computable estimate c of the center of the steady state distribution is available. We discuss the choice of c later in this section. Our algorithm works with conditional steady-state probabilities $p_n^{c-l,c+u}$ of being in state n, given that the process is in the range $\{c-l, \ldots, c+u\}$, for some lower limit l and upper limit u, defined as

$$p_n^{c-l,c+u} = \frac{\pi_n}{\sum_{m=c-l}^{c+u} \pi_m} \text{ for } n = c-l, \ldots, c+u. \tag{1}$$

Use of these conditional probabilities provides insight into how the algorithm works and facilitates derivation of error bounds.

The output of the algorithm is the expected value of $f(N)$, conditional on $c - l \leq N \leq c + u$, which we denote $\mathrm{E}[f(N)|l, u]$. In Section 4, we provide a bound on the relative error from approximating the unconditional expected value with the conditional one and use the bound to determine when the algorithm has computed a sufficiently accurate estimate.

Figure 1 outlines the algorithm, which begins at state c, by setting the conditional probability $p_c^{c,c} = 1$ and $l = u = 0$. At each iteration, one increases either l or u by one, computes a new state probability, increments the estimate of the performance measure $\mathrm{E}[f(N)]$, and normalizes all probabilities and the performance measure, which means assigning the correct values to them conditional on $c - l \leq N \leq c + u$. The normalization reduces the likelihood of under- and overflow, because the variable q in Figure 1 (the new state probability, before normalization) is bounded by $\max[\max\{a_n : n \geq c\}, \max\{b_n : n \leq c\}]$ and the performance measure estimate is bounded by $\max\{f(n) : c-l \leq n \leq c+u\}$. The choice between increasing l or u is made so as to extend the conditional distribution $\{p_n^{c-l,c+u}, n = c-l, \ldots, c+u\}$ in the direction where the probability of the extreme state is higher. The algorithm terminates when a relative error tolerance is met (discussed in Section 4). Thus, we avoid the difficulties of the previously described algorithms, in deciding how to truncate the state space.

How should state c, where the upward and downward recursions begin, be chosen? Although we interpret c as an estimate of the center of the steady state distribution, this estimate need not be precise. The algorithm computes probabilities for a contiguous range

Initialization: $p_c^{c,c} \leftarrow 1, l \leftarrow 0, u \leftarrow 0, \mathrm{E}[f(N)|l,u] \leftarrow f(c)$, converge $\leftarrow$ FALSE
While not converge
  If $p_{c+u}^{c-l,c+u} > p_{c-l}^{c-l,c+u}$ or $l = c$,
    $u \leftarrow u + 1$
    $q \leftarrow b_{c+u-1} p_{c+u-1}^{c-l,c+u-1}$
    $\mathrm{E}[f(N)|l,u] \leftarrow \mathrm{E}[f(N)|l,u] + q f(c+u)$
    Normalize: $\Sigma \leftarrow 1 + q$
      $p_{c+u}^{c-l,c+u} \leftarrow q/\Sigma$
      $p_n^{c-l,c+u} \leftarrow p_n^{c-l,c+u-1}/\Sigma$ for $c - l \le n \le s + u - 1$
      $\mathrm{E}[f(N)|l,u] \leftarrow \mathrm{E}[f(N)|l,u]/\Sigma$
  Else
    $l \leftarrow l + 1$
    $q \leftarrow a_{c-l+1} p_{c-l+1}^{c-l-1,c+u}$
    $\mathrm{E}[f(N)|l,u] \leftarrow \mathrm{E}[f(N)|l,u] + q f(c-l)$
    Normalize: $\Sigma \leftarrow 1 + q$
      $p_{c-l}^{c-l,c+u} \leftarrow q/\Sigma$
      $p_n^{c-l,c+u} \leftarrow p_n^{c-l+1,c+u}/\Sigma$ for $c - l + 1 \le n \le c + u$
      $\mathrm{E}[f(N)|l,u] \leftarrow \mathrm{E}[f(N)|l,u]/\Sigma$
  If termination criteria are met, then converge $\leftarrow$ TRUE
Return $\mathrm{E}[f(N)|l,u]$.

Figure 1: The general algorithm.

of states that is sufficiently wide to satisfy the relative error bound. Let us call this range the non-negligible probability range. As long as c is somewhere in this range, the number of operations performed by the algorithm will be the same. If c is above or below the non-negligible probability range, then the algorithm will perform additional operations (those needed for a downward or upward recursion from c to that range) but the accuracy of the output is not reduced—indeed, the accuracy is greater, because the algorithm includes more states in its computations. Besides computation time and accuracy, there is another consideration, namely, ease of implementation. Often, there will be a natural choice for c that simplifies the upward and downward recursions. In queueing models, the natural choice is often to set $c = s$, the number of servers, because the form of the expression for the death rate typically changes at $n = s$. The same is sometimes true of the birth rate, for example, if some customers balk when all servers are busy. Furthermore, for many performance measures (including all three measures discussed in Sections 5-7), one must do computations for state s regardless of the magnitude of π_s. Therefore, for multi-server queueing models, we suggest $c = s$ as a choice that simplifies the algorithm, causes no loss in accuracy, and only

results in increased computation time for systems whose offered load per server is either very high or very low, so that most of the probability mass is either above or below state s.

As presented in Figure 1, the algorithm stores all of the conditional probabilities from $p_{c-l}^{c-l,c+u}$ to $p_{c+u}^{c-l,c+u}$. We express the algorithm this way to clarify how the conditional probabilities are computed. However, typically it suffices to store $p_{c-l}^{c-l,c+u}$, $p_{c+u}^{c-l,c+u}$, Σ, and the current performance measure estimate $\mathrm{E}[f(N)|l,u]$ and this is how we will express the algorithm later when we specialize it for the Erlang B, C, and A systems.

The normalization that is performed at each iteration of our algorithm increases computation, compared to Algorithm 3, which normalizes only at the end of the algorithm. Our algorithm requires $(l+u)(o_1+o_2)$ operations, where o_1 is the number of operations required per iteration, not counting normalization, and o_2 is the number of operations required per iteration for normalization. From Figure 1, o_1 is at least eight (two additions, two multiplications, two logical tests, evaluating b_{c+u-1} or a_{c-l+1}, and evaluating $f(c+u)$ or $f(c-l)$) and $o_2 = 4$, assuming that only $p_{c-l}^{c-l,c+u}$, $p_{c+u}^{c-l,c+u}$, and $\mathrm{E}[f(N)|l,u]$ are normalized. Therefore, our algorithm requires up to $o_2/o_1 = 50\%$ more computation than a corresponding algorithm that normalizes only at the end, but this estimate is conservative for two reasons: (1) evaluating $f(c+u)$ or $f(c-l)$ may require considerably more than one operation (as we will see in Section 7) and (2) it is not clear how to implement termination criteria like the ones we develop in Section 4 without normalizing at every step. Therefore, it is not clear how to implement an algorithm that normalizes only at the end in a way that guarantees a specified accuracy. One could normalize only when the sum of the un-normalized probabilities that have been computed so far exceeds some threshold value and only check for termination in iterations where normalization is performed, but this would increase $l+u$, the number of iterations.

Example: To illustrate how the algorithm works, Figure 2 shows the conditional state probabilities after each iteration for an Erlang A system with $\lambda = 3, \mu = 1, s = 4$, and average patience rate $\gamma = 2$. The resulting birth and death rates are $\lambda_n = 3, n \geq 0$ and $\mu_n = n, n = 1, \ldots, 4, \mu_n = 4 + 2(n-4), n > 4$. The desired performance measure is the steady state probability $\pi_s = \pi_4$ and we set a relative error tolerance (discussed in Section 4) to $\epsilon_{\mathrm{RE}} = 0.01$. The algorithm begins by setting $p_4^{4,4} = 1$ and then extends the set of conditional probabilities either up or down at each iteration. Each panel shows the conditional probabilities after a particular iteration, the numerical value of the conditional

probability $p_4^{4-l,4+u}$ of being in state s, and a bound on the relative error (the bound is not shown until it is below 1). After 8 iterations, the algorithm has computed conditional probabilities for states 0 to 8, and this range is sufficiently wide to return an estimate of 0.1783 for π_4, which is guaranteed to be within 1% of the correct value.

4. Error Bounds

In this section, we provide bounds on absolute upper and lower truncation errors and use them to derive bounds on the relative error for the performance measure estimate. We define the following upper and lower truncation errors (and their sum) for the steady state probability mass in the tails:

$$\delta_1 = \sum_{n=0}^{c-l-1} \pi_n, \ \delta_2 = \sum_{n=c+u+1}^{\infty} \pi_n, \ \delta = \delta_1 + \delta_2. \tag{2}$$

We develop upper bounds Δ_i such that $\delta_i \leq \Delta_i, i = 1, 2$, with $\Delta = \Delta_1 + \Delta_2$. Our approach (motivated by Klar, 2000) assumes that the tails of the steady state distribution decay faster than a geometric series. Specifically, we assume for the upper tail that there exists an integer n_2 and a constant b such that for $n \geq n_2$, $b_n = \pi_{n+1}/\pi_n \leq b < 1$. This implies that $\pi_{n+m} = \prod_{i=0}^{m-1} b_{n+i}\pi_n \leq b^m \pi_n$ for $n \geq n_2$ and

$$\delta_2 = \sum_{n=c+u+1}^{\infty} \pi_n \leq \pi_{c+u} \sum_{n=1}^{\infty} b^n = \frac{b\pi_{c+u}}{1-b} \leq \Delta_2 \equiv \frac{bp_{c+u}^{c-l,c+u}}{1-b}, \tag{3}$$

where we have used the fact that $\pi_{c+u} \leq p_{c+u}^{c-l,c+u}$. We express the upper bound in terms of the conditional probability $p_{c+u}^{c-l,c+u}$, because that probability will be known when the algorithm uses the bound, while the unconditional probability π_{c+u} will be unknown. To get the tightest bound, one would typically set $b = \sup\{b_{c+u+i} \mid i = 0, 1, \dots\}$. If the sequence $\{b_{c+u}, b_{c+u+1}, \dots\}$ is non-increasing, as is often the case, then one can simply set $b = b_{c+u}$.

For the lower tail we similarly assume that there exists an integer n_1 and a constant a such that $a_n = \pi_{n-1}/\pi_n \leq a < 1$ for $n \leq n_1$. A similar argument as for the upper tail implies that

$$\delta_1 \leq \Delta_1 \equiv \frac{ap_{c-l}^{c-l,c+u}}{1-a}. \tag{4}$$

Now we turn to bounds on the relative error in estimating the performance measure $\mathrm{E}[f(N)]$. First, consider the simplest case where the performance measure is the steady

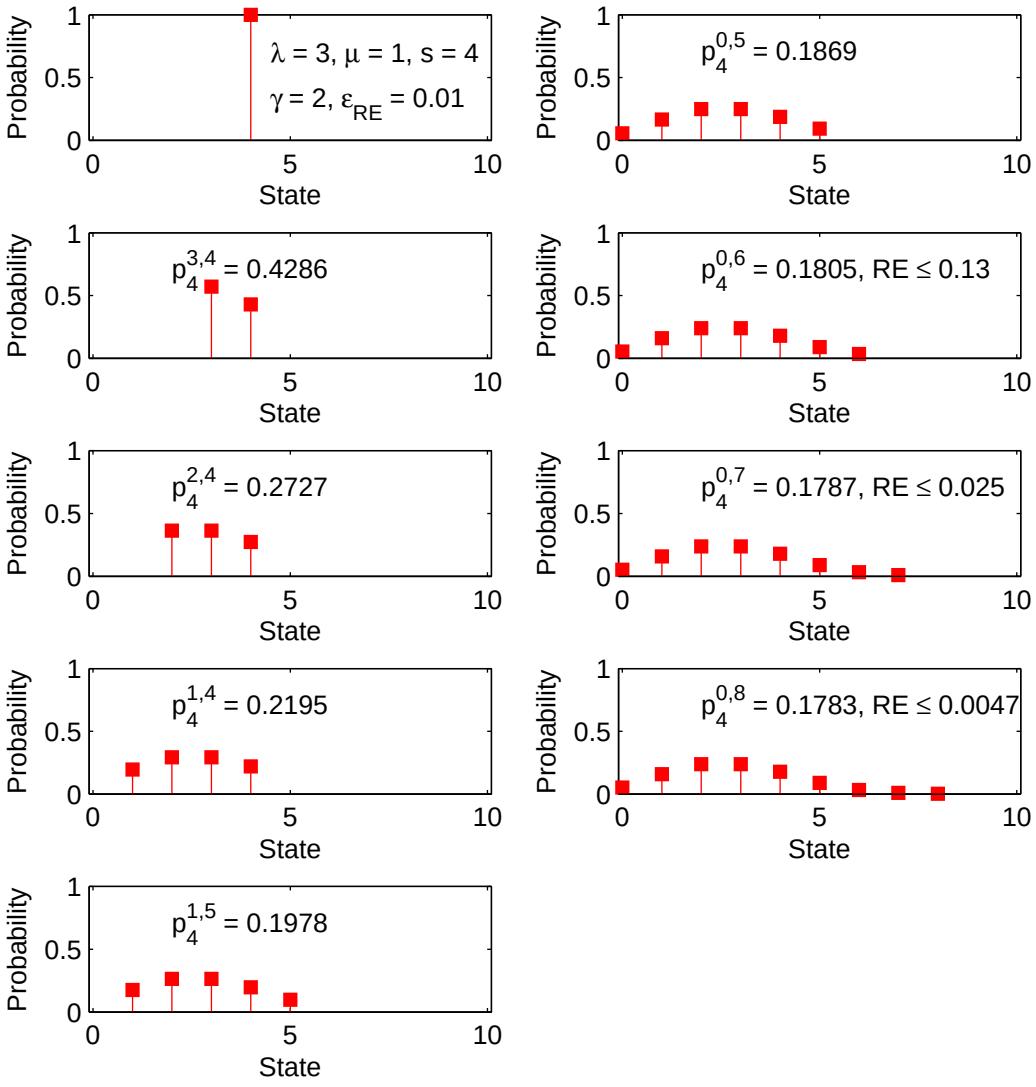

Figure 2: Numerical example.

state probability of an individual state m, so that $f(n) = \mathbf{1}(n = m)$. Assume that the lower and upper truncation limits have been extended sufficiently far so that m lies between $c - l$ and $c + m$ and the sum of the upper bounds on the probability mass in the tails, Δ, is less than 1. The relative error is $\mathrm{RE} \equiv (p_m^{c-l,c+u} - \pi_m)/\pi_m$. The conditional probability $p_m^{c-l,c+u}$ is an upper bound on the desired probability π_m, because $p_m^{c-l,c+u} = \pi_m/(1-\delta)$ and $\delta \leq \Delta < 1$, which implies $\mathrm{RE} \geq 0$. Using $p_m^{c-l,c+u} = \pi_m/(1-\delta)$ again, RE can be expressed as $\delta/(1-\delta)$, which is upper-bounded by $\Delta/(1-\Delta)$. To summarize, we have the following bounds on the relative error in estimating π_m with $p_m^{c-l,c+u}$:

$$0 \leq \mathrm{RE} \leq \frac{\Delta}{1-\Delta}. \tag{5}$$

Now we can state appropriate termination criteria to ensure that the relative error RE is less than a relative error tolerance ϵ_{RE}: (i) $c - l \leq m \leq c + u$ and (ii) $\Delta < \epsilon_{\mathrm{RE}}/(1 + \epsilon_{\mathrm{RE}})$. Condition (ii) ensures both that $\Delta < 1$ and $\mathrm{RE} < \epsilon_{\mathrm{RE}}$. It is straightforward to extend the analysis for the probability of a single state to the expected value of a function $f(n)$ that is finite within the range $c - l$ to $c + u$ and zero outside that range.

Now we consider more general performance measures $\mathrm{E}[f(N)]$. Define the following absolute truncation errors and their sum:

$$\epsilon_1 = \sum_{n=0}^{c-l-1} f(n)\pi_n, \ \epsilon_2 = \sum_{n=c+u+1}^{\infty} f(n)\pi_n, \ \epsilon = \epsilon_1 + \epsilon_2. \tag{6}$$

We develop upper bounds E_i (with $E = E_1 + E_2$) such that $\epsilon_i \leq E_i, i = 1, 2$. To develop the upper bounds we will separately consider three possibilities for the function f: that it is upper bounded by a constant d_0, a linear function $d_0 + d_1 n$, or a quadratic function $d_0 + d_1 n^2$, where $d_0, d_1 \geq 0$. Appendix A shows derivations of E_1 and E_2 for the constant, linear, and quadratic cases and the resulting bounds are shown in Table 1. Note that leaving a linear term out of the quadratic upper bound function is without loss of generality, because if $f(n) \leq d_0' + d_1' n + d_2' n^2$ where $d_0', d_1', d_2' \geq 0$, then we can take $d_0 = d_0'$ and $d_1 = d_1' + d_2'$, because $d_1' n + d_2' n^2 \leq (d_1' + d_2') n^2$ for $n \geq 0$.

Next, we use the absolute error bounds on the lower and upper tails to derive an upper bound on the relative truncation error $\mathrm{RE} = |\mathrm{E}[f(N)] - \mathrm{E}[f(N)|l, u]|/\mathrm{E}[f(N)]$ in estimating $\mathrm{E}[f(N)]$ with $\mathrm{E}[f(N)|l, u]$. We would like to have an upper bound on RE that is a function only of quantities that are available at each stage of our algorithm. From the preceding

Table 1: Absolute error bounds

Probability mass in the tails:

$$\Delta_1 = ap_{c-l}^{c-l,c+u}/(1-a) \qquad \Delta_2 = bp_{c+u}^{c-l,c+u}/(1-b)$$

Constant case: $f(n) \leq d_0$

$$E_1 = d_0\Delta_1 \qquad E_2 = d_0\Delta_2$$

Linear case: $f(n) \leq d_0 + d_1 n$

$$\begin{aligned} E_1 &= d_0\Delta_1 \\ &\quad + d_1(c-l)p_{c-l}^{c-l,c+u}/(1-a) \end{aligned} \qquad \begin{aligned} E_2 &= (d_0 + d_1(c+u))\Delta_2 \\ &\quad + d_1 b p_{c+u}^{c-l,c+u}/(1-b)^2 \end{aligned}$$

Quadratic case: $f(n) \leq d_0 + d_1 n^2$

$$\begin{aligned} E_1 &= d_0\Delta_1 \\ &\quad + d_1(c-l)^2 p_{c-l}^{c-l,c+u}/(1-a) \\ &\quad + a(1+a)p_{c-l}^{c-l,c+u}/(1-a)^3 \end{aligned} \qquad \begin{aligned} E_2 &= (d_0 + d_1(c+u)^2)\Delta_2 \\ &\quad + 2d_1 b(c+u)p_{c+u}^{c-l,c+u}/(1-b)^2 \\ &\quad + d_1 b(1+b)p_{c+u}^{c-l,c+u}/(1-b)^3 \end{aligned}$$

---

definitions, we have that

$$\mathrm{E}[f(N)|l,u] = \sum_{n=c-l}^{c+u} f(n)p_n^{c-l,c+u} = \frac{\sum_{n=c-l}^{c+u} f(n)\pi_n}{\sum_{n=c-l}^{c+u}\pi_n} = \frac{\mathrm{E}[f(N)] - \epsilon}{1-\delta}, \tag{7}$$

which implies

$$\begin{aligned} \mathrm{E}[f(N)] &= \mathrm{E}[f(N)|l,u](1-\delta) + \epsilon \\ &\geq \mathrm{E}[f(N)|l,u](1-\delta) \\ &\geq \mathrm{E}[f(N)|l,u](1-\Delta) \\ \Rightarrow \frac{1}{\mathrm{E}[f(N)]} &\leq \frac{1}{\mathrm{E}[f(N)|l,u](1-\Delta)}. \end{aligned} \tag{8}$$

(For the last step, we assumed that $\Delta < 1$, which will be true if l and u are large enough.)

Equation (7) also implies that

$$\begin{aligned} |\mathrm{E}[f(N)] - \mathrm{E}[f(N)|l,u]| &= |\mathrm{E}[f(N)|l,u](1-\delta) + \epsilon - \mathrm{E}[f(N)|l,u]| \\ &= |-\mathrm{E}[f(N)|l,u]\delta + \epsilon| \\ &\leq \mathrm{E}[f(N)|l,u]\delta + \epsilon \\ &\leq \mathrm{E}[f(N)|l,u]\Delta + E. \end{aligned} \tag{9}$$

Expressions (8) and (9) result in the following bound on the relative truncation error:

$$\mathrm{RE} = \frac{|\mathrm{E}[f(N)] - \mathrm{E}[f(N)|l,u]|}{\mathrm{E}[f(N)]} \leq \frac{\mathrm{E}[f(N)|l,u]\Delta + E}{\mathrm{E}[f(N)|l,u](1-\Delta)}. \tag{10}$$

The bound on the right-hand-side involves only quantities that are available in the algorithm when the lower and upper limits have reached l and u. Note that when $E = 0$, which will happen if $f(n)$ is zero below $c-l$ and above $c+u$, then we recover the upper bound from (5). The general termination criteria for our algorithm are:

$$\Delta < 1 \text{ and } \frac{\mathrm{E}[f(N)|l,u]\Delta + E}{\mathrm{E}[f(N)|l,u](1-\Delta)} < \epsilon_{\mathrm{RE}}. \tag{11}$$

In implementing the termination criteria, one can set Δ_2 to 1 and E_2 to ∞ if the condition $b_{c+u+i} < 1$ is not yet met for all $i > 0$ and one can set $\Delta_2 = E_2 = 0$ if the state space is finite and $c + u = K$ (and similarly for the lower tail).

Recall that the algorithm chooses between extending the conditional state space up or down based on which extreme state, $c - l$ or $c + u$, has higher conditional probability. Alternatively, one could base this choice on which absolute error bound is higher, Δ_1 or Δ_2.

In the remainder of the paper, we apply our general algorithm to the computation of representative performance measures for the Erlang B, C, and A models.

5. Computing the Erlang B loss Probability

For the Erlang B model, with offered load $r = \lambda/\mu$ and s servers, the birth rates are $\lambda_n = \lambda$ for $n = 0, \ldots, s-1$ and the death rates are $\mu_n = n\mu$ for $n = 1, \ldots, s$. We assume that the performance measure is the loss probability $\pi_s = B(r,s)$. It is instructive to consider two choices for the center estimate: $c = 0$ and $c = s$.

If $c = 0$, then our algorithm computes the sequence of conditional probabilities $p_1^{0,1}$, $\ldots$, $p_s^{0,s}$, which is simply the sequence of loss probabilities $B(r,1), \ldots, B(r,s)$, for systems with

offered load r and number of servers increasing from 1 to s. The algorithm iterates until $u = s$, because it must continue until it has computed $p_s^{0,s}$, and then it returns $B(r, s)$ with no truncation error. Each iteration of the algorithm involves extending $u - 1$ to u, and computing $p_u^{0,u}$ from $p_{u-1}^{0,u-1}$ as follows:

$$p_u^{0,u} = \frac{(r/u)p_{u-1}^{0,u-1}}{1 + (r/u)p_{u-1}^{0,u-1}}.$$

Dividing the numerator and denominator of the right-hand-side fraction with $(r/u)p_{u-1}^{0,u-1}$ and expressing $p_u^{0,u}$ as $B(r, u)$, we obtain the following known difference equation (e.g., see Jagerman, 1974):

$$B(r, u) = \left(1 + \frac{u}{rB(r, u-1)}\right)^{-1}.$$

Using $c = s$ instead of $c = 0$ can lead to considerable computational savings, because one may be able to obtain an accurate estimate of $B(r, s)$ while truncating the state space far above state 0. With $c = s$ the upper truncation error bound Δ_2 equals zero, because there is no probability mass above s. The lower truncation error bound, for $n = s - l < r$, is $\Delta_1 = (s - l)p_{s-l}^{s-l,s}/(r - (s - l))$. To ensure that $\Delta_1 < 1$, we must have $l > s - r/(1 + p_{s-l}^{s-l,s})$. The relative error termination criterion becomes

$$\frac{(s - l)p_{s-l}^{s-l,s}}{r - (s - l)} < \frac{\epsilon_{\mathrm{RE}}}{1 + \epsilon_{\mathrm{RE}}}.$$

Table 2 shows the lower truncation limit l for two sequences of systems, one where $r = s$ and another where $r = s - \sqrt{s}$. In both cases, we set the relative error tolerance to 10^{-4}. We see that the number of states, l, that the algorithm evaluates grows slightly faster than $\sqrt{s}$. Even for $s = 1,000,000$, the calculation of B is essentially instantaneous when implemented in Matlab. The computation times T in this and subsequent tables are elapsed times in microseconds, averaged over 1,000 calls to the algorithm, on a 2.1 GHz Windows PC with an AMD processor, a 64-bit operating system, and 4 GB of RAM. (The online supplement provides the Matlab code used in Sections 5-7.)

6. Computing the Erlang C Delay Probability

The Erlang C model has birth rates $\lambda_n = \lambda$ for $n \geq 0$ and death rates $\mu_n = \min(n, s)\mu$ for $n \geq 1$. We assume $r = \lambda/\mu < s$, for stability. This model has a geometrically decaying upper tail and therefore we have a choice: to use the general version of our algorithm, as shown in

Table 2: Erlang B numerical examples. Average elapsed times (T) are in microseconds.

	$r = s$				$r = s - \sqrt{s}$			
s	l	$l/\sqrt{s}$	B	T	l	$l/\sqrt{s}$	B	T
10	10	3.16	0.2146	2	10	3.16	0.0724	2
100	41	4.10	0.0757	6	50	5.00	0.0270	7
1,000	146	4.62	0.0248	25	178	5.62	0.0089	26
10,000	490	4.90	0.0079	74	597	5.97	0.0029	87
100,000	1,628	5.14	0.0025	227	1,971	6.23	0.00091	292
1,000,000	5,369	5.37	0.00080	570	6,455	6.45	0.00029	610

Figure 1, or to modify the algorithm to take advantage of the geometric upper tail. Figures 3 and 4 illustrate the two approaches, for computing the delay probability $C(r, s) = \sum_{n=s}^{\infty} \pi_n$. In both cases, we start the iterations at $c = s$. The geometric upper tail implies that

$$\sum_{i=1}^{\infty} p_{s+i}^{s,\infty} = p_s^{s,\infty} \frac{r}{s-r}. \tag{12}$$

Using this expression, one can replace the original birth-death process with a birth-death process with finite state space $\{0, \ldots, s+1\}$, where state $s+1$ corresponds to states $\{s+1, s+2, \ldots\}$ in the original process. The birth and death rates in the transformed process are the same as for the original process, except that $\lambda_s = r/(s-r)$ and $\mu_{s+1} = 1$, which results in the correct probability for state $s+1$, as equation (12) shows. In the transformed process, $C(r, s) = \pi_s + \pi_{s+1}$ and $\Delta_2 = 0$, after one has computed the conditional probabilities for states s and $s+1$.

Tables 3-5 show computational results using the two algorithms, with s varying from 10 to 1,000,000 and ϵ_{RE} set to 10^{-4}. The three tables illustrate what Gans et al. (2003) coined the efficiency-driven regime (where $(s-r)/\sqrt{s}$ approaches zero—we use $r = s-1$), the quality-and-efficiency-driven regime (where $(s-r)/\sqrt{s}$ approaches a positive constant—we use $r = s - \sqrt{s}$), and the quality-driven regime (where s/r approaches a constant less than one—we use $r = 0.99s$). The computations were essentially instantaneous, except for one case: $s = 1,000,000$ in the efficiency-driven regime, using the algorithm that does not take advantage of the geometric tail. In that case, the computations took 0.8 seconds. Comparing the two algorithms, the lower truncation limit l is always larger for the general algorithm, because the algorithm that takes advantage of the geometric tail has zero upper-tail truncation error, and can therefore "afford" higher lower-tail truncation error for a given

Initialization: $C \leftarrow 1, l \leftarrow 0, u \leftarrow 0, q_{c-l} \leftarrow 1, q_{c+u} \leftarrow 1, \text{converge} \leftarrow \text{FALSE}, b \leftarrow \lambda/(s\mu)$
$\Delta_1 \leftarrow 1, \Delta_2 \leftarrow 1, E_1 \leftarrow 0, E_2 \leftarrow 1$
While not converge
    If $q_{c+u} > q_{c-l}$ or $l = s$,
      $u \leftarrow u + 1$
      $q_{c+u} \leftarrow bq_{c+u}$
      $C \leftarrow C + q_{c+u}$
      Normalize: $\Sigma \leftarrow 1 + q_{c+u}$
                 $q_{c+u} \leftarrow q_{c+u}/\Sigma, q_{c-l} \leftarrow q_{c-l}/\Sigma, C \leftarrow C/\Sigma$
                 $\Delta_2 \leftarrow bq_{c+u}/(1-b), E_2 \leftarrow \Delta_2$
    Else
      $l \leftarrow l + 1$
      $a \leftarrow (s - l + 1)\mu/\lambda$
      $q_{c-l} \leftarrow aq_{c-l}$
      Normalize: $\Sigma \leftarrow 1 + q_{c-l}$
                 $q_{c+u} \leftarrow q_{c+u}/\Sigma, q_{c-l} \leftarrow q_{c-l}/\Sigma, C \leftarrow C/\Sigma$
                 If $a < 1$, then $\Delta_1 \leftarrow aq_{c-l}/(1-a)$
                 If $l = s$, then $\Delta_1 \leftarrow 0$
    If $\Delta_1 + \Delta_2 < 1$ and $(C(\Delta_1 + \Delta_2) + E_1 + E_2)/(C(1 - \Delta_1 - \Delta_2)) < \epsilon_{\text{RE}}$,
    then converge $\leftarrow$ TRUE
Return C.

Figure 3: The general algorithm specialized to computing the delay probability $C(r, s)$ for an Erlang C system.

Initialization: $l \leftarrow 0, q_{c-l} \leftarrow 1, \text{converge} \leftarrow \text{FALSE}, \Delta_1 \leftarrow 1, E_1 \leftarrow 0, \Delta_2 \leftarrow 0, E_2 \leftarrow 0$
Geometric upper tail: $\Sigma \leftarrow 1 + s/(s - r)$
                      $q_{c-l} \leftarrow q_{c-l}/\Sigma$
                      $C \leftarrow 1$
While not converge
    $l \leftarrow l + 1$
    $a \leftarrow (s - l + 1)\mu/\lambda$
    $q_{c-l} \leftarrow aq_{c-l}$
    Normalize: $\Sigma \leftarrow 1 + q_{c-l}$
               $q_{c-l} \leftarrow q_{c-l}/\Sigma, C \leftarrow C/\Sigma$
               If $a < 1$, then $\Delta_1 \leftarrow aq_{c-l}/(1-a)$
               If $l = s$, then $\Delta_1 \leftarrow 0$
    If $\Delta_1 + \Delta_2 < 1$ and $(C(\Delta_1 + \Delta_2) + E_1 + E_2)/(C(1 - \Delta_1 - \Delta_2)) < \epsilon_{\text{RE}}$,
    then converge $\leftarrow$ TRUE
Return C.

Figure 4: The general algorithm specialized to computing the delay probability $C(r, s)$ for an Erlang C system, modified to take advantage of the geometric upper tail.

Table 3: Erlang C numerical examples, showing the impact of taking advantage of the geometric tail, in the efficiency-driven regime. In all cases, μ was set to 1, λ was set to $s-1$, and ϵ_{RE} was set to 10^{-4}. Average elapsed times (T) are in microseconds.

	General						Geometric			
s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	C	T	l	$l/\sqrt{s}$	C	T
10	10	92	3.16	29.09	0.6687	17	10	3.16	0.6687	2
100	43	979	4.30	97.90	0.8828	129	33	3.30	0.8828	8
1,000	139	9,882	4.40	312.5	0.9612	985	96	3.04	0.9613	17
10,000	444	98,978	4.44	989.8	0.9876	8.835	269	2.69	0.9876	42
100,000	1,406	990,190	4.45	3,131	0.9960	81,513	725	2.29	0.9961	144
1,000,000	4,449	9,903,067	4.45	9,903	0.9987	803,373	1,840	1.84	0.9988	236

Table 4: Erlang C numerical examples, showing the impact of taking advantage of the geometric tail, in the quality-and-efficiency-driven regime. In all cases, μ was set to 1, λ was set to $s-\sqrt{s}$, and ϵ_{RE} was set to 10^{-4}. Average elapsed times (T) are in microseconds.

	General						Geometric			
s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	C	T	l	$l/\sqrt{s}$	C	T
10	10	24	3.16	7.59	0.1978	6	10	3.16	0.1979	2
100	50	89	5.00	8.90	0.2169	24	43	4.30	0.2170	7
1,000	168	293	5.31	9.27	0.2215	78	145	4.59	0.2215	22
10,000	541	940	5.41	9.40	0.2228	212	468	4.68	0.2228	72
100,000	1,722	2,984	5.45	9.44	0.2232	542	1,487	4.70	0.2232	212
1,000,000	5,457	9,447	5.46	9.45	0.2233	1,432	4,711	4.71	0.2233	461

relative error budget. So not only does the algorithm take advantage of the geometric tail to avoid the upward recursion, it also requires fewer iterations of the downward recursion. The two algorithms always return the same value for C to three significant digits.

Comparing the three limiting regimes, we see that $l+u$ (a measure of the non-negligible probability range) appears to grow linearly with s for the efficiency-driven regime and appears to grow linearly with $\sqrt{s}$ for the quality-and-efficiency-driven regime. In the quality-driven regime, u does not grow and l grows faster than $\sqrt{s}$, while C approaches zero, implying that state s is above the non-negligible probability range and essentially all of the steady state probability mass is below state s. These numerical results are consistent with asymptotic results for these limiting regimes.

Table 5: Erlang C numerical examples, showing the impact of taking advantage of the geometric tail, in the quality-driven regime. In all cases, μ was set to 1, λ was set to $0.99s$, and ϵ_{RE} was set to 10^{-4}. Average elapsed times (T) are in microseconds.

	General						Geometric			
s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	C	T	l	$l/\sqrt{s}$	C	T
10	10	983	3.16	311	0.9637	102	8	2.53	0.9638	2
100	43	979	4.30	97.9	0.8828	155	33	3.30	0.8828	6
1,000	147	969	4.65	30.6	0.6591	165	121	3.83	0.6591	20
10,000	541	940	5.41	9.40	0.2228	211	468	4.68	0.2228	76
100,000	2,675	916	8.46	2.90	0.0008	378	2,173	6.87	0.0008	274
1,000,000	20,839	916	20.8	0.92	0.0000	2,181	13,714	13.7	0.0000	1,182

7. Computing Erlang A Virtual Waiting Time Probabilities

The Erlang A model has birth rates $\lambda_n = \lambda$ for $n \geq 0$ and death rates $\mu_n = n\mu$ for $n = 1, \ldots, s$ and $\mu_n = s\mu + (n-s)\gamma$ for $n > s$. Straightforward computations reveal that $a_n = n\mu/\lambda$ for $n < s$ and $b_n = \lambda/(s\mu + (n-s)\gamma)$ for $n \geq s$.

The general algorithm can be used to compute Erlang A steady state performance measures that can be expressed as the expected value of a function of the number of customers in the system, including the average number in system, the average number in queue, average time in system, and the probability of abandonment. We illustrate how to compute the probability that the virtual waiting time exceeds a threshold value, because this calculation is more involved than any of the other performance measures just mentioned.

Let W be the steady-state virtual waiting time for a customer that does not abandon (an "infinitely patient customer"), and let W_n be W conditional on $N = n$. We wish to compute $A \equiv A(\lambda, \mu, \gamma, s, t) = \Pr\{W > t\}$. To do so with our algorithm, we require a way to compute $f(s+u) = \Pr\{W_{s+u} > t\}$. Computing this function is not trivial. Riordan (1962, p. 111) derived an expression (shown in Appendix B) for $f(s+u)$ that is a summation of u terms. Unfortunately, the terms in the summation have alternating signs and can grow quickly, leading to numerical difficulties. As an example of how these difficulties lead to nonsensical results even for moderate parameter values, suppose $\mu = 2, \gamma = 1, s = 10$, and $t = 1$. Then $f(s+60)$, evaluated using double precision and Riordan's expression, evaluates to -36.53. Garnett et al. (2002) and Deslauriers et al. (2007) comment on these numerical

difficulties. To avoid these difficulties, we use the following expression, which was derived by Richard Simard and first published in Deslauriers et al. (2007):

$$f(s+u) = \Pr\{W_{s+u} > t\} = \xi^{\phi} \sum_{j=0}^{u} (\phi)_j \frac{(1-\xi)^j}{j!} \quad \text{for } u \geq 0 \tag{13}$$

where $\phi = s\mu/\gamma$, $(\phi)_0 = 1$, $(\phi)_j = (\phi)(\phi+1)\cdots(\phi+j-1)$, $\xi = \exp(-\gamma t)$, and $f(n) = 0$ for $n < s$. This expression starts at $f(s) = \xi^{\phi} = \exp(-s\mu t)$, is strictly increasing in u, and approaches 1 as $u \to \infty$ (the summation consists of the first u terms of the binomial series for $\xi^{-\phi}$). From a numerical point of view, this expression is much better behaved than Riordan's expression. We provide a new proof of (13) in Appendix B.

Figure 5 shows the general algorithm, specialized for computing A, setting c to s and using expression (13) to compute $f(n)$, except for one modification. The modification addresses the problem that if $s\mu t$ is large, then $\xi^{\phi} = \exp(-s\mu t)$ will underflow. To avoid this, we rescale the expression in (13) at each iteration where u is increased, so that if the current value of the summation is x, then the summation is scaled down to 1 and the term multiplying the summation (initially ξ^{ϕ}) is multiplied by x. Further, we use the logarithm of the term multiplying the summation instead of the term itself.

Since the function $f(n)$ is a probability, $f(n) \leq 1$, and therefore $E_i = \Delta_i, i = 1, 2$. The upper bound on the relative error reduces to $\Delta(1+A)/(A(1-\Delta))$.

Note that to evaluate $f(n)$ using (13), one needs to evaluate terms in the summation starting at $n = s$. Therefore, in this case, there is no computational advantage to starting the iterations of the general algorithm in a state other than $c = s$. Even if almost all of the probability is concentrated above or below s, state s needs to be evaluated regardless, in order to evaluate $f(n)$.

In contrast to Erlang C systems, in an Erlang A system with many servers, the quality of service for infinitely patient customers can be excellent even if the offered load $r = \lambda/\mu$ equals or exceeds the number of servers. We illustrate this in Tables 6 and 8, where the offered load increases as $r = s$ and $r = s + \sqrt{s}$, respectively, and in both cases A, the probability that an infinitely patient customer has to wait more than $t = 0.01$ average service times, approaches zero when s grows. In Tables 7 and 9, we show the probability of delay for an infinitely patient customer, obtained by setting $t = 0$, and we observe that this probability approaches a constant that is strictly between zero and one, consistent with known asymptotic results for the Erlang A model. All of the computations in Tables 6-9 were essentially instantaneous (0.021 seconds or less).

Initialization: $q_{c-l} \leftarrow 1, q_{c+u} \leftarrow 1, l \leftarrow 0, u \leftarrow 0, \phi \leftarrow s\mu/\gamma, \xi \leftarrow \exp(-\gamma t)$,
$\theta \leftarrow -s\mu t, \kappa \leftarrow 1-\xi, A \leftarrow \exp(\theta), f \leftarrow 1, g \leftarrow 1$,
$\Delta_1 \leftarrow 1, \Delta_2 \leftarrow 1, \text{converge} \leftarrow \text{FALSE}$
While not converge
  If $q_{c+u} > q_{c-l}$ or $l = s$,
    $u \leftarrow u+1$
    $q_{c+u} \leftarrow \lambda/(s\mu + u\gamma) q_{c+u}$
    $g \leftarrow g(\phi + u - 1)\kappa/u, f \leftarrow f + g, \theta \leftarrow \theta + \ln(f)$
    $A \leftarrow A + q_{c+u}\exp(\theta)$
    Normalize: $\Sigma \leftarrow 1 + q_{c+u}$
      $q_{c-l} \leftarrow q_{c-l}/\Sigma, q_{c+u} \leftarrow q_{c+u}/\Sigma$
      $A \leftarrow A/\Sigma$
  If $s\mu + u\gamma > \lambda$, then $\Delta_2 \leftarrow \lambda q_{c+u}/(s\mu + u\gamma - \lambda)$
  Else
    $l \leftarrow l+1$
    $q_{c-l} \leftarrow (s-l+1)\mu/\lambda q_{c-l}$
    Normalize: $\Sigma \leftarrow 1 + q_{c-l}$
      $q_{c-l} \leftarrow q_{c-l}/\Sigma, q_{c+u} \leftarrow q_{c+u}/\Sigma$
      $A \leftarrow A/\Sigma$
  If $l = s$, then $\Delta_1 \leftarrow 0$,
  Else if $(s-l)\mu < \lambda$, then $\Delta_1 \leftarrow (s-l)\mu q_{c-l}/(\lambda - (s-l)\mu)$
  If $\Delta_1 + \Delta_2 < 1$ and $A > 0$ then
    If $(\Delta_1 + \Delta_2)(1+A)/(A(1-\Delta_1-\Delta_2)) < \epsilon_{\text{RE}}$, then converge $\leftarrow$ TRUE
Return A.

Figure 5: The general algorithm specialized to computing virtual waiting time tail probabilities for an Erlang A system.

We note from these tables that $(l+u)/\sqrt{s}$ increases slightly with s, and is similar for $t = 0.01$ and $t = 0$, except for $s = 1,000,000$, where $(l+u)/\sqrt{s}$ is considerably larger for $t = 0.01$ than for $t = 0$. To understand why, note that as s increases, $f(s) = \exp(-s\mu t)$ approaches zero. Therefore, in the initial iterations of the algorithm, A will be close to zero and the relative error bound $\Delta(1+A)/(A(1-\Delta))$ will be large. To reduce the relative error bound below the tolerance ϵ_{RE}, Δ must be reduced below $\epsilon_{\text{RE}}A/(1+A(1+\epsilon_{\text{RE}})) \approx \epsilon_{\text{RE}}A$. Therefore, if A is close to zero (as it is for $t = 0.01$ and $s = 1,000,000$), $l+u$ must be large. Note that the code could be simplified by using the more conservative termination condition $\Delta < \epsilon_{\text{RE}}A$, which avoids division-by-zero errors when A underflows.

Table 6: Erlang A numerical examples, with $\mu = 1, \gamma = 0.5, t = 0.01, r = s$, and $\epsilon_{\text{RE}} = 10^{-4}$. Average elapsed times (T) are in microseconds.

s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	A	T
10	10	20	3.1623	6.3246	0.6093	54
100	39	60	3.9000	6.0000	0.5637	132
1,000	129	189	4.0793	5.9767	0.4856	383
10,000	423	604	4.2300	6.0400	0.2820	1,185
100,000	1,521	2,160	4.8098	6.8305	0.0149	4,319
1,000,000	8,302	11,768	8.3020	11.7680	9.412×10^{-13}	21,425

Table 7: Erlang A numerical examples, with $\mu = 1, \gamma = 0.5, t = 0, r = s$, and $\epsilon_{\text{RE}} = 10^{-4}$. Average elapsed times (T) are in microseconds.

s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	A	T
10	10	20	3.1623	6.3246	0.6196	45
100	39	60	3.9000	6.0000	0.5967	120
1,000	128	187	4.0477	5.9135	0.5893	370
10,000	411	587	4.1100	5.8700	0.5869	1,141
100,000	1,306	1,852	4.1299	5.8565	0.5861	3,405
1,000,000	4,135	5,854	4.1350	5.8540	0.5859	10,831

Table 8: Erlang A numerical examples, with $\mu = 1, \gamma = 0.5, t = 0.01, r = s + \sqrt{s}$, and $\epsilon_{\text{RE}} = 10^{-4}$. Average elapsed times (T) are in microseconds.

s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	A	T
10	9	29	2.8460	9.1706	0.9221	73
100	29	82	2.9000	8.2000	0.9174	143
1,000	92	249	2.9093	7.8741	0.8910	411
10,000	294	781	2.9400	7.8100	0.7672	1,225
100,000	999	2,552	3.1591	8.0701	0.2074	4,261
1,000,000	6,070	12,118	6.0700	12.1180	7.942×10^{-9}	20,728

Table 9: Erlang A numerical examples, with $\mu = 1, \gamma = 0.5, t = 0, r = s + \sqrt{s}$, and $\epsilon_{\mathrm{RE}} = 10^{-4}$. Average elapsed times (T) are in microseconds.

s	l	u	$l/\sqrt{s}$	$u/\sqrt{s}$	A	T
10	9	29	2.8460	9.1706	0.9261	52
100	29	82	2.9000	8.2000	0.9291	135
1,000	92	249	2.9093	7.8741	0.9305	401
10,000	292	777	2.9200	7.7700	0.9310	1,171
100,000	923	2,446	2.9188	7.7349	0.9312	3,712
1,000,000	2,920	7,725	2.9200	7.7250	0.9312	11,667

8. Conclusions

We presented a general algorithm to compute performance measures that can be expressed as expected values of functions of the population size of birth-death processes with a steady state distribution whose tails decay geometrically or faster, a class that includes many performance measures of interest for multi-server queueing systems, for example, the Erlang B, C, and A systems with finite or infinite waiting room. The algorithm avoids under- and overflow and is simple to implement, requiring neither numerical integration nor the use of special functions. The algorithm computations are, in most cases, essentially instantaneous even for systems with a million servers.

Appendix A: Derivations of Truncation Error Bounds

We make repeated use of the following power series, where $0 < \rho < 1$:

$$\sum_{n=0}^{\infty} \rho^n = \frac{1}{1-\rho}, \ \sum_{n=0}^{\infty} \rho^n n = \frac{\rho}{(1-\rho)^2}, \ \sum_{n=0}^{\infty} \rho^n n^2 = \frac{\rho(1+\rho)}{(1-\rho)^3}$$

We begin with the absolute upper truncation error ϵ_2. In the constant case, where $f(n) \leq d_0$, $\epsilon_2 \leq d_0\delta_2$ so we can take $E_2 = d_0\Delta_2$. In the linear case, where $f(n) \leq d_0 + d_1 n$,

$$\begin{aligned}
\epsilon_2 &\leq \sum_{n=c+u+1}^{\infty} (d_0 + d_1 n)\pi_n \\
&\leq d_0\delta_2 + d_1\pi_{c+u} \sum_{n=c+u+1}^{\infty} b^{n-c-u} n \\
&= d_0\delta_2 + d_1\pi_{c+u} \sum_{n=1}^{\infty} b^n (n + c + u) \\
&\leq d_0\delta_2 + d_1(c+u)p_{c+u}^{c-l,c+u} \sum_{n=1}^{\infty} b^n + d_1 p_{c+u}^{c-l,c+u} \sum_{n=1}^{\infty} n b^n \\
&\leq (d_0 + d_1(c+u))\Delta_2 + d_1 p_{c+u}^{c-l,c+u} \sum_{n=0}^{\infty} n b^n \\
&= E_2 \equiv (d_0 + d_1(c+u))\Delta_2 + \frac{d_1 b p_{c+u}^{c-l,c+u}}{(1-b)^2}.
\end{aligned}$$

In the quadratic case, where $f(n) \leq d_0 + d_1 n^2$, we have

$$\begin{aligned}
\epsilon_2 &\leq \sum_{n=c+u+1}^{\infty} (d_0 + d_1 n^2)\pi_n \\
&\leq d_0\delta_2 + d_1\pi_{c+u} \sum_{n=c+u+1}^{\infty} b^{n-c-u} n^2 \\
&= d_0\delta_2 + d_1\pi_{c+u} \sum_{n=1}^{\infty} b^n (n + c + u)^2 \\
&\leq d_0\delta_2 + d_1(c+u)^2 p_{c+u}^{c-l,c+u} \sum_{n=0}^{\infty} b^n + 2d_1(c+u)p_{c+u}^{c-l,c+u} \sum_{n=0}^{\infty} n b^n + d_1 p_{c+u}^{c-l,c+u} \sum_{n=0}^{\infty} n^2 b^n \\
&\leq E_2 \equiv (d_0 + d_1(c+u)^2)\Delta_2 + \frac{2d_1 b(c+u)p_{c+u}^{c-l,c+u}}{(1-b)^2} + \frac{d_1 b(1+b)p_{c+u}^{c-l,c+u}}{(1-b)^3}.
\end{aligned}$$

The necessary manipulations for the lower tail are similar, so we do not show them here.

Appendix B: Proof of Erlang A Virtual Waiting Time Formula

Consider an Erlang A system with service rate μ, abandonment rate γ, and s servers. Let $\phi = s\mu/\gamma$, $\xi = \exp(-\gamma t)$, and $\tau = \gamma t$ denote time measured in average patience durations. Use $(\phi)_j$ to denote a rising factorial, with $(\phi)_0 = 1$ and $(\phi)_j = (\phi)(\phi+1)\cdots(\phi+j-1)$ for

$j = 1, 2, \ldots$. Let W_n be the steady-state virtual waiting time for an infinitely patient test customer, conditional on n customers in the system when the test customer arrives. Here are two ways to compute $f(s+u;\tau) \equiv \Pr\{W_{s+u} > \tau/\gamma\}$:

$$f(s+u;\tau) = g(s+u;\tau) = \xi^{\phi} \sum_{j=0}^{u} (\phi)_j \frac{(1-\xi)^j}{j!} \tag{A1}$$

$$= h(s+u;\tau) = \xi^{\phi} \frac{(\phi)_{u+1}}{u!} \sum_{j=0}^{u} (-1)^j \binom{u}{j} \frac{\xi^j}{\phi + j}. \tag{A2}$$

Expression (A2) is originally from Riordan (1962, p. 111), with equivalent formulas in Koole (2004) and Ross (2007, p. 302). Richard Simard proved expression (A1) using transform methods (that proof has not been published) and Deslauriers et al. (2007) and Avramidis et al. (2009) used expression (A1). Here, we provide an alternative proof, using induction. We take as given the following differential-difference equation from Riordan (1962, eqn. (78a), p. 110):

$$\frac{dg(s+u;\tau)}{du} = -(\phi + n)(g(s+u;\tau) - g(s+u-1;\tau)),\ u = 1, 2, \ldots,\ \tau \geq 0. \tag{A3}$$

Proof. We use induction on u. The waiting time W_{s+u} can viewed as the time to absorption in state $s-1$, starting in state $s+u$, in a pure death process with state space $\{s-1, \ldots, s+u\}$ and death rates $\mu_n = s\mu + (n-s)\gamma$. Therefore, W_{s+u} can be expressed as $\sum_{i=0}^{u} X_i$ where the X_i are independent exponentially distributed random variables with means $\gamma/(\phi+i)$. Expression (A1) is valid for $u = 0$ because $g(s;\tau) = \xi^{\phi} = \exp(-\phi\tau) = \exp(-s\mu t) = \Pr\{X_0 > t\}$.

For $u > 0$, $W_{s+u} = W_{s+u-1} + X_u$ and W_{s+u-1} and X_u are independent. Assume that $g(s+i;\tau)$ is correct for $i < u$. Then $g(s+u;\tau)$ can be expressed as:

$$\begin{aligned} g(s+u;\tau) = \Pr\{W_{s+u} > \tau/\gamma\} &= \Pr\{W_{s+u-1} > \tau/\gamma\} + \Pr\{W_{s+u} > \tau/\gamma \text{ and } W_{s+u-1} \leq \tau/\gamma\} \\ &= g(s+u-1;\tau) + \Pr\{X_n + W_{s+u-1} > \tau/\gamma \text{ and } W_{s+u-1} \leq \tau/\gamma\} \end{aligned}$$

The probability in the last expression can be computed as follows:

$$\begin{aligned} \Pr\{X_u + W_{s+u-1} > \tau/\gamma \text{ and } W_{s+u-1} \leq \tau/\gamma\} &= \int_{w=0}^{\tau/\gamma} f_{W_{s+u-1}}(w) \Pr\{X_u > \tau/\gamma - w\} dw \\ &= \int_{w=0}^{\tau/\gamma} f_{W_{s+u-1}}(w) e^{-(\phi+u)(\tau/\gamma - w)} dw \\ &= \xi^{\phi+u} \int_{w=0}^{\tau/\gamma} f_{W_{s+u-1}}(w) e^{(\phi+u)w} dw \\ &= -\xi^{\phi+u} \int_{w=0}^{\tau/\gamma} \frac{d}{dw} \Pr\{W_{s+u-1} > w\} e^{(\phi+u)w} dw. \end{aligned}$$

From (A1) and (A3), the derivative can be expressed as follows:

$$\begin{aligned}\frac{d}{dw}\Pr\{W_{s+u-1} > w\} &= -(\phi+u-1)\left(\Pr\{W_{s+u-1} > w\} - \Pr\{W_{s+u-2} > w\}\right) \\ &= -(\phi+u-1)e^{-\phi\gamma w}(\phi)_{s+u-1}\frac{(1-e^{-\gamma w})^{s+u-1}}{(u-1)!}.\end{aligned}$$

Therefore,

$$\begin{aligned}\int_{w=0}^{\tau/\gamma}\frac{d}{dw}\Pr\{W_{s+u-1} > w\}e^{(\phi+u)w}dw &= -\int_{w=0}^{\tau/\gamma}(\phi+u-1)e^{-\phi\gamma w}(\phi)_{u-1}\frac{(1-e^{-\gamma w})^{u-1}}{(u-1)!}e^{(\phi+u)\gamma w}dw \\ &= -(\phi+u-1)(\phi)_{u-1}\int_{w=0}^{\tau/\gamma}\frac{(1-e^{-\gamma w})^{u-1}}{(u-1)!}e^{u\gamma w}dw.\end{aligned}$$

The integral in the last line can be evaluated using the binomial theorem:

$$\begin{aligned}&\int_{w=0}^{\tau/\gamma}\frac{(1-e^{-\gamma w})^{u-1}}{(u-1)!}e^{u\gamma w}dw = \frac{1}{\gamma(u-1)!}\int_{x=0}^{\tau}\left(1-e^{-x}\right)^{u-1}e^{ux}dx \\ &= \frac{1}{\gamma(u-1)!}\sum_{j=0}^{u-1}\binom{u-1}{j}(-1)^j\int_{x=0}^{\tau}\left(e^{-x}\right)^j e^{ux}dx \\ &= \frac{1}{\gamma(u-1)!}\sum_{j=0}^{u-1}\binom{u-1}{j}(-1)^j\int_{x=0}^{\tau}e^{(u-j)x}dx \\ &= \frac{1}{\gamma(u-1)!}\sum_{j=0}^{u-1}\binom{u-1}{j}\frac{(-1)^j}{u-j}\left[e^{(u-j)\tau}-1\right] \\ &= \frac{e^{u\tau}}{\gamma u!}\sum_{j=0}^{u-1}\binom{u}{j}\left(-e^{-\tau}\right)^j - \frac{1}{\gamma u!}\sum_{j=0}^{u-1}\binom{u}{j}(-1)^j \\ &= \frac{e^{u\tau}}{\gamma u!}\left(\left(1-e^{-\tau}\right)^u - \binom{u}{u}\left(-e^{-\tau}\right)^u\right) - \frac{1}{\gamma u!}\left((1-1)^u - \binom{u}{u}(-1)^u\right) \\ &= \frac{e^{u\tau}\left(1-e^{-\tau}\right)^u}{\gamma u!} - \frac{(-1)^u}{u!\gamma} + \frac{(-1)^u}{u!\gamma} = \frac{e^{u\tau}\left(1-e^{-\tau}\right)^u}{\gamma u!} = \frac{(1-\xi)^u}{\xi^u\gamma u!}.\end{aligned}$$

Combining all of the above results in:

$$\begin{aligned}\Pr\{W_{s+u} > t \text{ and } W_{s+u-1} \le t\} &= -\xi^{\phi+u}\int_{w=0}^{\tau/\gamma}\frac{d}{dw}\Pr\{W_{s+u-1} > w\}e^{(\phi+u)\gamma w}dw \\ &= \xi^{\phi+u}(\phi+u-1)\gamma(\phi)_{u-1}\frac{(1-\xi)^u}{\xi^u\gamma u!} \\ &= \xi^{\phi}(\phi)_u\frac{(1-\xi)^u}{u!},\end{aligned}$$

which implies that $g(s+u;\tau)$ is correct. □

Acknowledgments

This work was partially funded by the Natural Sciences and Engineering Research Council of Canada under grant 203534-07 and an Undergraduate Student Summer Support grant. This support is gratefully acknowledged. We also thank the associate editor and two referees for useful comments.

References

Avramidis, A. N., W. Chan, P. L'Ecuyer. 2009. Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions* **41** 483–497.

Deslauriers, A., P. L'Ecuyer, J. Pichitlamken, A. Ingolfsson, A. N. Avramidis. 2007. Markov chain models of a telephone call center with call blending. *Computers & Operations Research* **34** 1616–1645.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management* **5** 71–141.

Garnett, O., A. Mandelbaum, M. Reiman. 2002. Designing a call center with impatient customers. *Manufacturing & Service Operations Management* **4** 208–227.

Ingolfsson, A., F. Gallop. 2003. Queueing Toolpak 4.0. Available from `https://apps.business.ualberta.ca/aingolfsson/qtp/`.

Jagerman, D. L. 1974. Some properties of the Erlang loss function. *The Bell System Technical Journal* **53** 525–551.

Karlin, S., J. McGregor. 1957. The classification of birth and death processes. *Transactions of the American Mathematical Society* **86** 366–400.

Klar, B. 2000. Bounds on tail probabilities of discrete distributions. *Probability in the Engineering and Informational Sciences* **14** 161–171.

Koole, G. 2004. A formula for tail probabilities of Cox distributions. *Journal of Applied Probability* **41** 935–938.

Mandelbaum, A., S. Zeltyn. 2007. Service engineering in action: The Palm/Erlang-A queue, with applications to call centers. D. Spath, K.-P. Fähnrich, eds., *Advances in Services Innovation*. Springer, Berlin, 17–45. doi:10.1007/978-3-540-29860-1_2.

Novozhilov, A. S., G. P. Karev, E. V. Koonin. 2006. Biological applications of the theory of birth-and-death processes. *Briefings in Bioinformatics* **7** 70–85. doi:10.1093/bib/bbk006.

Riordan, J. 1962. *Stochastic Service Systems*. The SIAM series in applied mathematics, John Wiley & Sons, Inc., New York.

Ross, S. M. 2007. *Introduction to Probability Models*. 9th ed. Academic Press, Boston, MA.

Savage, S. 1997. Statistical analysis for the masses. B. D. Spencer, ed., *Statistics and Public Policy*. Oxford University Press, New York, 262–276.

Smith, D. K. 2002. Calculation of steady-state probabilities of M/M queues: further approaches. *Journal of Applied Mathematics & Decision Sciences* **6** 43–50.

Tijms, H. C. 1994. *Stochastic Models: An Algorithmic Approach*. John Wiley & Sons, Inc., New York.

van Doorn, E. A. 1980. Stochastic monotonicity of birth-death processes. *Advances in Applied Probability* **12** 59–80.

Whitt, W. 2005. Engineering solution of a basic call-center model. *Management Science* **51** 221–235.