1 AN INTRODUCTION TO NUMERICAL TRANSFORM INVERSION AND ITS APPLICATION TO PROBABILITY MODELS

Joseph Abate[1]
Gagan L. Choudhury[2]
Ward Whitt[3]

[1]900 Hammond Road
Ridgewood, NJ 07450-2908

[2]AT&T Labs, Room 1L-238
Holmdel, NJ 07733-3030
gagan@buckaroo.att.com

[3]AT&T Labs, Room A117
180 Park Avenue
Florham Park, NJ 07932-0971
wow@research.att.com

(Chapter in *Computational Probability*, W. Grassman (ed.), Kluwer, Boston, 1999, pp. 257–323)

1.1 INTRODUCTION

Numerical transform inversion has an odd place in computational probability. Historically, transforms were exploited extensively for solving queueing and related probability models, but only rarely was numerical inversion attempted. The model descriptions were usually left in the form of transforms. Vivid examples are the queueing books by Takács [78] and Cohen [30]. When possible, probability distributions were calculated analytically by inverting transforms, e.g., by using tables of transform pairs. Also, moments of probability distributions were computed analytically by differentiating the transforms and, occasionally, approximations were developed by applying asymptotic methods to transforms, but only rarely did anyone try to compute probability distributions by numerically inverting the available transforms. However, there were exceptions, such as the early paper by Gaver [42]. (For more on the history of numerical transform inversion, see our earlier survey [7].) Hence, in the application of probability models to engineering, transforms became regarded more as mathematical toys than practical tools. Indeed, the conventional wisdom was that numerical transform inversion was very difficult. Even numerical analysts were often doubtful of the numerical stability of inversion algorithms. In queueing, both theorists and practitioners lamented about the "Laplace curtain" obscuring our understanding of system behavior.

Thus, the perceived difficulty of numerical transform inversion served as a motivation for much progress in computational probability. Instead of directly trying to invert available transforms, researchers primarily responded to this situation by developing new computational methods and new modeling approaches that avoided transforms. A good example is the collection of matrix-geometric methods in Neuts [64]. In his preface, Neuts concludes that "the oft-lamented Laplacian curtain, which covers the solution and hides the structural properties of many interesting stochastic models, is now effectively lifted." Indeed, matrix-geometric methods and other alternative approaches are remarkably effective, as can be seen from Chapter 5 in this book.

However, since then, it has been recognized that the concerns about numerical transform inversion were misplaced. Contrary to previous impressions, it is not difficult to numerically compute probabilities and other quantities of interest in probability models by directly inverting the transforms. For example, all the transforms in the queueing books by Takács [78] and Cohen [30] can be numerically inverted. To a large extent, the numerical inversion can be done by directly computing from the classical inversion formulas. However, there are complications, so that some additional thought is beneficial. But thought has been given, so that now there are a variety of effective inversion algorithms based on the classical inversion formulas.

The purpose of this chapter is to provide an introduction to numerical transform inversion and its application to compute probabilities in stochastic models. We focus on Laplace transforms and generating functions (z transforms), but similar inversion methods apply to other transforms such as characteristic functions (Fourier transforms). In Section 2 we present numerical inversion algorithms and in Sections 3–5 we present applications to queueing models with numerical examples. The queueing examples in Sections 3–5 include: computing transient characteristics in the M/M/c/0 Erlang loss model (Section 3), computing the steady-state waiting-time distribution in the general GI/GI/1 queue (Section 4) and computing steady-state characteristics of product-form closed queueing networks (Section 5). These three models are arguably the three most fundamental models in queueing theory. These three examples illustrate the extra work that often must be done to treat more difficult problems in which a simple closed-form transform is not available. Other examples can be found in the references.

Now we illustrate how transforms and numerical inversion can help by considering two simple examples. For these examples, the probability analysis is elementary or well known, but nevertheless calculation can be difficult without numerical inversion. Calculating cumulative distribution functions for these examples is rarely suggested in textbook discussions. On the other hand, calculation is elementary with numerical transform inversion, because in these examples a simple closed-form transform is readily available to be numerically inverted. Indeed, the numerical inversion approach to these problems is suitable for introductory textbooks. The introductory textbook on performance evaluation by Kobayashi [58] included a brief discussion of numerical transform inversion (an early variant of the Fourier-series method for numerically inverting Laplace transforms to be discussed in Section 2), but it did not strongly influence practice. The recent introductory textbook on stochastic processes by Kao [52] uses the inversion algorithms for Laplace transforms and generating functions presented in Section 2 here.

Example 1.1.1 (Project Completion Time) Suppose that a project is composed of n independent tasks, one performed after another. We want to know the probability distribution of the time to complete the entire project when there is uncertainty about the time to complete each task. Let the time to complete task k be a random variable X_k with probability density function f_k (depending on k). Then the time to complete the project is the sum $S_n = X_1 + \ldots + X_n$, which has a density g_n that is the convolution of the n densities $f_1, \ldots, f_n$; i.e., g_n can be defined recursively by the convolution integral.

$$g_n(t) = \int_0^\infty g_{n-1}(t-x) f_n(x) dx \tag{1.1}$$

and its associated cumulative distribution function (cdf) is

$$G_n(t) = \int_0^t g_n(x) dx \ . \tag{1.2}$$

By the central limit theorem for independent non-identically distributed random variables, e.g., p. 262 of Feller [41], the sum S_n is approximately normally distributed with mean and variance equal to the sum of the component means and variances, provided that n is not too small and that the individual task times X_k have finite variances and are suitably small compared to the sum. However, if one or two task times are much larger than the others, then the normal approximation can be quite inaccurate. Then it may be much better to compute the exact distribution.

Unfortunately, however, except for very small n, the desired cdf $G_n(t)$ in (1.2) is difficult to compute directly because it involves an $(n-1)$-dimensional integral. However, it is usually easy to compute by numerical transform inversion provided that we know the Laplace transforms of the densities f_k, i.e.,

$$\hat{f}_k(s) \equiv \int_0^\infty e^{-st} f_k(t)dt \ . \tag{1.3}$$

If we only know the means and variances of the task times, then we might fit appropriate distributions, such as gamma distributions, to these moments, and then compute the distribution of the project completion time. A gamma density with shape parameter α and scale parameter λ has Laplace transform

$$\hat{f}(s) = \left(\frac{\lambda}{\lambda + s}\right)^\alpha \ . \tag{1.4}$$

The associated mean and variance are α/λ and α/λ^2. Hence, the parameters α and λ can easily be chosen to match the mean and variance.

To obtain the Laplace transform of the density of the project completion time, we use the basic transform law stating that the transform of a convolution is the product of the transforms. Thus the Laplace transform of the density g_n is

$$\hat{g}_n(s) \equiv \int_0^\infty e^{-st} g_n(t)dt = \prod_{k=1}^n \hat{f}_k(s) \ , \quad n \geq 1 \ . \tag{1.5}$$

Moreover, we use another basic transform law to get the Laplace transform of the cdf G_n. Since G_n is the integral of g_n,

$$\hat{G}_n(s) \equiv \int_0^\infty e^{-st} G_n(t)dt = \frac{\hat{g}_n(s)}{s} \ . \tag{1.6}$$

Combining (1.5) and (1.6), we see that the Laplace transform $\hat{G}_n$ of the cdf G_n is conveniently expressed in terms of the given transforms $\hat{f}_1, \ldots, \hat{f}_n$. Thus we can easily apply an inversion algorithm to calculate the cdf $G_n(t)$ for any desired t by numerically inverting the Laplace transform $\hat{G}_n$.

In the next section we describe inversion algorithms that can be used to calculate the cdf values $G_n(t)$ for any t given the Laplace transform $\hat{G}_n$. These algorithms are intended for the case in which the function to be calculated, here G_n, is relatively smooth. For example, there is no difficulty if each task time density has a gamma distribution as in (1.4). The major source of numerical difficulty in the inversion algorithm stems from discontinuities in the function or its derivatives. However, discrete probability distributions can be calculated easily by numerically inverting their generating functions.

In the numerical inversion of Laplace transforms, the required computation turns out to be remarkably light: For each value of t, we typically need to add about 50 transform terms and, by (1.5) and (1.6), we need to perform n multiplications for each.

Example 1.1.1 illustrated two basic laws for manipulating transforms. Unlike numerical transform inversion, these basic transform laws are quite well known, so we will not dwell upon them. Sources for additional background on transforms are the appendices of Kleinrock [56], Giffin [44], Doetsch [35], [36], Van Der Pol [67] and Poularikas [70].

Example 1.1.2 (The Renewal Function) Let $M(t)$ be the renewal function, recording the expected number of renewals in the interval $(0, t]$, associated with a renewal process having an interrenewal-time cdf F with density f, i.e.,

$$M(t) = \sum_{n=1}^{\infty} F^{n*}(t) \ , \quad t \geq 0 \ , \tag{1.7}$$

where F^{n*} is the n-fold convolution of the cdf F with itself. The renewal function is of considerable interest because it arises in many probability models, but because of the convolution in (1.7) it is rather difficult to calculate directly. However, it is elementary to compute using numerical inversion. Let $\hat{f}$ be the Laplace transform of f. Then the Laplace transform of M is

$$\hat{M}(s) \equiv \int_0^{\infty} e^{-st} M(t) dt = \sum_{n=1}^{\infty} \frac{\hat{f}(s)^n}{s} = \frac{\hat{f}(s)}{s(1-\hat{f}(s))} \ . \tag{1.8}$$

Given the transform $\hat{f}$, numerical inversion algorithms apply to easily calculate $M(t)$ for any desired t by inverting the transform $\hat{M}$ in (1.8).

Numerical inversion applies easily to the two examples given here, because the transform is available in closed form. The challenge in more difficult examples is to compute the transform values, because transforms often are not available in closed form. For example, for the busy-period distribution in the M/G/1 queue, the transform is available implicitly via the Kendall functional equation. In that case, the transform values can be readily calculated by iteration; see Abate and Whitt [9]. The examples in Sections 3–5 illustrate how transform values can be obtained for numerical inversion in other more complicated queueing examples.

1.2 NUMERICAL INVERSION ALGORITHMS

In this section we present some numerical inversion algorithms. We primarily aim to explain how the algorithms work. Of course, in order to *apply* the inversion algorithms, it is not necessary to know how the algorithms work. Using the algorithms, transform expressions as in Example 1.1.1 and 1.1.2 above can be immediately applied to calculate the desired quantities. However, even if we only want to apply inversion algorithms, it is good to understand how they work.

We begin in Section 1.2.1 by presenting classical inversion formulas for Laplace transforms, which form the basis for the numerical inversion algorithms. Next, in Section 1.2.2 we present a variant of the Fourier-series algorithm for numerically inverting a Laplace transform. Afterwards, in Section 1.2.3 we provide further discussion of Euler summation to help explain why it is so effective in accelerating the convergence of the infinite series arising in the Fourier-series method. Finally, in Section 1.2.4 we present the Fourier-series algorithm for numerically inverting a generating function.

In this section we consider only one-dimensional transforms. For extensions of the inversion algorithms to multi-dimensional transforms; see Choudhury, Lucantoni and Whitt [24].

1.2.1 Inversion Formulas for Laplace Transforms

Given a real-valued function f on the nonnegative real line, its *Laplace transform* (LT) is defined by

$$\mathcal{L}(f)(s) \equiv \hat{f}(s) \equiv \int_0^\infty e^{-st} f(t)dt, \tag{2.9}$$

where s is a complex variable with $Re(s) > 0$. (Let $Re(s)$ be the real part and $Im(s)$ the imaginary part of s.) Conditions ensuring that the integral in (2.9) exists appear in the literature; e.g., Doetsch [36].

For probability applications, it is useful to consider Laplace-Stieltjes transforms as well as ordinary Laplace transforms. Given a nonnegative *random variable* X with *cumulative distribution function* (cdf) F, i.e., $F(t) = P(X \leq t)$, $t \geq 0$, and *probability density function* (pdf) f when it exists, i.e., when $F(t) = \int_0^t f(u)du$, $t \geq 0$, we define its Laplace transform by

$$Ee^{-sX} \equiv \mathcal{L}(f)(s) \equiv \hat{f}(s) \equiv \int_0^\infty e^{-st} dF(t) = \int_0^\infty e^{-st} f(t)dt; \tag{2.10}$$

i.e., $\hat{f}$ is the Laplace transform of the pdf f as in (2.9), but $\hat{f}$ is the *Laplace-Stieltjes transform* (LST) of the cdf F. The associated LT of F is thus $\hat{F}(s) = \hat{f}(s)/s$. In probability applications, we are typically interested in the cdf, which can be computed by numerically inverting the LT $\hat{F}$. We are also often interested in the *complementary cdf* (ccdf) $F^c(t) \equiv 1 - F(t)$, which has LT $\hat{F^c}(s) = (1 - \hat{f}(s))/s$. For these probability transforms $\hat{f}, \hat{F}$ and $\hat{F^c}$, the integrals always exist. We will always apply the inversion to LTs instead of LSTs. We usually aim to calculate the ccdf F^c by numerically inverting its transform $\hat{F^c}$.

In this section we present a way to numerically invert the LT $\hat{f}$ in (2.9) in order to calculate $f(t)$ for any given t. We have assumed that f is real-valued, as when f is a pdf, cdf or ccdf, but inversion also applies when f is complex-valued. Indeed, if $f(t) = f_1(t) + if_2(t)$, $t \geq 0$, for $i = \sqrt{-1}$, then $\hat{f}(s) = \hat{f}_1(s) + i\hat{f}_2(s)$, so it is clear that inversion extends easily to complex-valued functions. Complex-valued functions arise naturally in the iterated one-dimensional inversion of multidimensional transforms, as we illustrate here in Section 5. In that setting, the function to be calculated by the one-dimensional inversion will be complex-valued in all steps except the last, e.g., see [24].

A natural starting point for numerical inversion of Laplace transforms is the Bromwich inversion integral; e.g., see Theorem 24.4 on p. 257 of Doetsch [36].

Theorem 1 (*Bromwich inversion integral*) *Given the Laplace transform $\hat{f}$ in (2.1), the function value $f(t)$ can be recovered from the contour integral*

$$f(t) = \frac{1}{2\pi i} \int_{b-i\infty}^{b+i\infty} e^{st} \hat{f}(s) ds \ , \quad t > 0 \ , \tag{2.11}$$

where b is a real number to the right of all singularities of $\hat{f}$, and the contour integral yields the value 0 for $t < 0$.

As usual with contour integrals, there is flexibility in choosing the contour provided that it is to the right of all singularities of $\hat{f}$. However, there is no need to be bothered by the complexities of complex variables. If we choose a specific contour and perform a change of variables, then we obtain an integral of a real-valued function of a real variable. First, by making the substitution $s = b + iu$ in (2.11), we obtain

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{(b+iu)t} \hat{f}(b+iu) du \ . \tag{2.12}$$

Then, since

$$e^{(b+iu)t} = e^{bt}(\cos ut + i \sin ut) \ ,$$

$\sin ut = -\sin(-ut)$, $\cos ut = \cos(-ut)$, $Im(\hat{f}(b+iu)) = -Im(\hat{f}(b-iu))$ and $Re(\hat{f}(b+iu)) = Re(\hat{f}(b-iu))$, and from the fact that the integral in (2.11) is 0 for $t < 0$, we obtain

$$f(t) = \frac{2e^{bt}}{\pi} \int_0^{\infty} Re(\hat{f}(b+iu)) \cos(ut) du \tag{2.13}$$

and

$$f(t) = \frac{-2e^{bt}}{\pi} \int_0^{\infty} Im(\hat{f}(b+iu)) \sin(ut) du \ . \tag{2.14}$$

Theorem 1 implies that $f(t)$ can be calculated from the transform $\hat{f}$ by performing a numerical integration (quadrature). Since there are many numerical integration algorithms, e.g., see Davis and Rabinowitz [34], there are obviously many possible approaches to numerical transform inversion via the Bromwich inversion integral. In this context, the remaining goal is to exploit the special structure of the integrand in (2.13) in order to calculate the integral accurately and efficiently.

However, there also are quite different numerical inversion algorithms, because the Bromwich inversion integral is not the only inversion formula. To illustrate, we mention a few others. First, there is the Post-Widder inversion formula, which involves differentiation instead of integration. It is the basis for the Jagerman-Stehfest procedure in Section 8 of Abate and Whitt [7]. (See that source for further discussion.)

Theorem 2 (*Post-Widder inversion formula*) *Under regularity conditions,*

$$f(t) = \lim_{n\to\infty} \frac{(-1)^n}{n!} \left(\frac{n+1}{t}\right)^{n+1} \hat{f}^{(n)}((n+1)/t) \ , \tag{2.15}$$

where $\hat{f}^{(n)}$ *is the* n^{th} *derivative of* $\hat{f}$.

For small n, the terms on the right in (2.15) can serve as useful rough approximations, because they tend to inherit the structure of f; see Jagerman [49] and [50].

The next inversion formula is a discrete analog of the Post-Widder formula involving finite differences. It is the basis for the Gaver-Stehfest procedure in Section 8 of Abate and Whitt [7]. Let Δ be the difference operator, defined by $\Delta\hat{f}(n\alpha) = \hat{f}((n+1)\alpha) - \hat{f}(n\alpha)$ and let $\Delta^k = \Delta(\Delta^{k-1})$.

Theorem 3 (*discrete analog of Post-Widder formula*) *If f is a bounded real-valued function that is continuous at t, then*

$$f(t) = \lim_{n\to\infty} (-1)^n \frac{\ln 2}{t} \frac{(2n)!}{n!(n-1)!} \Delta^n \hat{f}(n \ln 2/t) \ .$$

Finally, we mention the Laguerre-series inversion formula, which is the basis for the Laguerre-series or Weeks' algorithm in Weeks [80] and Abate, Choudhury and Whitt [5] and [6].

Theorem 4 (*Laguerre-series representation*) *Under regularity conditions,*

$$f(t) = \sum_{n=0}^{\infty} q_n l_n(t), \ \ t \geq 0, \tag{2.16}$$

where

$$l_n(t) = e^{-t/2} L_n(t), \ \ t \geq 0, \tag{2.17}$$

$$L_n(t) = \sum_{k=0}^{n} \binom{n}{k} \frac{(-t)^k}{k!}, \ \ t \geq 0, \tag{2.18}$$

and

$$\hat{q}(z) \equiv \sum_{n=0}^{\infty} q_n z^n = (1-z)^{-1} \hat{f}((1+z)/2(1-z)) \ . \tag{2.19}$$

The function L_n in (2.18) are the *Laguerre polynomials*, while l_n in (2.17) are the associated *Laguerre functions*. The scalars q_n in (2.16) are the *Laguerre coefficients* and $\hat{q}$ in (2.19) is the *Laguerre generating function* (the generating function of the Laguerre coefficients).

Now we consider a specific algorithm based on the Bromwich inversion integral.

1.2.2 The Fourier-Series Method for Laplace Transforms

In this section we develop one variant of the Fourier-series method. The specific algorithm is concisely summarized at the end of the section. Pointers to the literature appear there as well.

There are two natural ways to develop the Fourier-series method. One way starts with the Bromwich inversion integral. As indicated before, we can directly apply a standard numerical integration procedure to perform the integration in (2.13). Somewhat surprisingly, perhaps, one of the most naive approaches – the trapezoidal rule – proves to be remarkably effective in this context. If we use a step size h, then the trapezoidal rules gives

$$f(t) \approx f_h(t) \equiv \frac{he^{bt}}{\pi} Re(\hat{f}(b)) + \frac{2he^{bt}}{\pi} \sum_{k=1}^{\infty} Re(\hat{f}(b+ikh)) \cos(kht) \ , \qquad (2.20)$$

where $Re(\hat{f}(b)) = \hat{f}(b)$ since b is real.

Formula (2.20) states that the desired function value $f(t)$ is approximated by the trigonometric series $f_h(t)$. The remaining problem is to control the errors associated with calculating $f(t)$ via this representation.

There are three sources of error associated with the trapezoidal rule: first, the *discretization error* associated with approximating $f(t)$ by $f_h(t)$ in (2.20); second, the *truncation error* associated with approximately calculating the infinite series in (2.20) (which might not be by simple truncation); and, third, the *roundoff error* associated with addition and multiplication in calculating $f_h(t)$ in (2.20). The remaining discussion is devoted to showing how to control these three sources of error when we use the trapezoidal rule.

The Discretization Error

The standard theory of numerical integration, as on p. 32 of Davis and Rabinowitz [34], shows that the discretization error for the trapezoidal rule is bounded by a term of order $O(h^2)$, which is not so good. However, the special trigonometric structure here tends to yield a much better approximation, of order $O(e^{-c/h})$ for some constant c. The better error bound follows from a second way to develop the Fourier-series method, which does not rely on the Bromwich contour integral. We can anticipate the second approach, though, by looking at (2.20). From (2.20), we can recognize that f_h is a trigonometric series. We thus can ask if f_h is a Fourier series of some, necessarily periodic, function f_p related to f (expressed in terms of f instead of its LT $\hat{f}$). The discretization error is then $f_h - f = f_p - f$.

The second approach starts, without considering the Bromwich inversion integral, by directly constructing a periodic function f_p approximating f and then constructing the Fourier series of this periodic function. However, to facilitate the procedure, we first damp f by letting $g(t) = e^{-bt} f(t)$ and then extend it over the entire real line by letting $g(t) = 0$ for $t < 0$. We then form a periodic function approximating g by considering an infinite series of translates of the original function, i.e., we let

$$g_p(t) = \sum_{k=-\infty}^{\infty} g\left(t + \frac{2\pi k}{h}\right) . \tag{2.21}$$

The damping ensures that the series in (2.21) converges, and tends to make the terms in the tails relatively small. The construction in (2.21) is called *aliasing*. We then work with g and g_p, recovering f from $f(t) = e^{bt}g(t)$ at the end.

The idea now is to construct the Fourier series of the periodic function g_p. (We elaborate in the proof of Theorem 5 below.) The result is equivalent to the *Poisson summation formula*. This procedure yields what we want because the coefficients of the Fourier series can be expressed directly in terms of the Laplace transform values. Indeed, the Fourier series of the periodic function g_p in (2.21) is a minor modification of the function f_h in (2.20). This second approach yielding the discretization error in the trapezoidal rule explains the name *Fourier-series method*.

We summarize the main result in the following theorem.

Theorem 5 *Under regularity conditions, the discretization error for the trapezoidal rule approximation in (2.20) is*

$$e_h(t) \equiv f_h(t) - f(t) = \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} f\left(t + \frac{2\pi k}{h}\right) e^{-2\pi kb/h} . \tag{2.22}$$

Proof. We start with the periodic function g_p in (2.21) with period $2\pi/h$, which we assume is well defined. (It clearly is when $|f(t)| \leq C$ for all t.) The term for $k = 0$ in the right side of (2.21) is clearly $g(t)$. Thus the other terms in the series make up the aliasing error for g.

We then represent the periodic function g_p by its *complex Fourier series*

$$g_p(t) = \sum_{k=-\infty}^{\infty} c_k e^{ikht}, \tag{2.23}$$

where c_k is the kth *Fourier coefficient* of g_p, i.e.,

$$c_k = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} g_p(t) e^{-ikht} dt . \tag{2.24}$$

We *assume* that the Fourier-series representation (2.23) is valid. The large literature on Fourier series provides conditions; e.g., [79]. For practical purposes, it is important that t be a continuity point of the function f, and thus of g.

Now we substitute the series (2.21) for $g_p(t)$ into (2.24) to obtain

$$c_k = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \sum_{k=-\infty}^{\infty} g\left(t + \frac{2k\pi}{h}\right) e^{-ikht} dt = \frac{h}{2\pi} \int_{-\infty}^{\infty} g(t) e^{-ikht} dt$$

$$= \frac{h}{2\pi}\int_0^\infty e^{-bt}f(t)e^{-ikht}dt = \frac{h}{2\pi}\hat{f}(b+ikh)\ . \tag{2.25}$$

Thus, we can write $g_p(t)$ in two different ways. First, from (2.21) and the relation between g and f, we have

$$g_p(t) = \sum_{k=-\infty}^{\infty} g\left(t+\frac{2\pi k}{h}\right) = \sum_{k=-\infty}^{\infty} f\left(t+\frac{2\pi k}{h}\right)e^{-b(t+2\pi k/h)}\ . \tag{2.26}$$

Second, from (2.23) and (2.24), we have

$$g_p(t) = \sum_{k=-\infty}^{\infty} c_k e^{ikht} = \frac{h}{2\pi}\sum_{k=-\infty}^{\infty}\hat{f}(b+ikh)e^{ikht}\ . \tag{2.27}$$

Combining the terms in (2.26) and (2.27) involving f and $\hat{f}$, we obtain a version of the *Poisson summation formula*

$$\sum_{k=-\infty}^{\infty} f\left(t+\frac{2\pi k}{h}\right)e^{-b(t+2\pi k/h)} = \frac{h}{2\pi}\sum_{k=-\infty}^{\infty}\hat{f}(b+ikh)e^{ikht}\ . \tag{2.28}$$

We then obtain what we want by focusing on the single term for $k=0$ on the left in (2.28); i.e.,

$$e^{-bt}f(t) = \frac{h}{2\pi}\sum_{k=-\infty}^{\infty}\hat{f}(b+ikh)e^{ikht} - \sum_{\substack{k=-\infty\\k\neq 0}}^{\infty} f\left(t+\frac{2\pi k}{h}\right)e^{-b(t+2\pi k/h)}\ . \tag{2.29}$$

Multiplying both sides of (2.29) by e^{bt} yields

$$f(t) = \frac{h}{2\pi}\sum_{k=-\infty}^{\infty}\hat{f}(b+ikh)e^{(b+ikh)t} - \sum_{\substack{k=-\infty\\k\neq 0}}^{\infty} f\left(t+\frac{2\pi k}{h}\right)e^{-2\pi k/h}\ . \tag{2.30}$$

Now, reasoning just as did to go from (2.12) to (2.13), from (2.30) we obtain $f(t) = f_h(t) - e_h(t)$ for $f_h(t)$ in (2.20) and $e_h(t)$ in (2.22). ∎

Because of the alternative route to approximation f_h in (2.20) via aliasing, the discretization error in (2.6) is also called the *aliasing error*. Having characterized this error, we next want to choose the parameters b and h in order to control it. Afterwards we will calculate the infinite series in (2.20). Since $f(t)=0$ for $t<0$, we obtain from (2.22) for $h<2\pi/t$ that

$$e_h(t) = \sum_{k=1}^{\infty} f(t+\frac{2\pi k}{h})e^{-2\pi kb/h}$$

and the aliasing error depends only on values $f(x)$ for $x > t+2\pi/h$. We exploit this property to make the aliasing error small. For example, if

$$|f(x)| \le C \quad \text{for all} \quad x > t+2\pi/h\ , \tag{2.31}$$

then

$$|e_h(t)| \le \sum_{k=1}^{\infty} Ce^{-2\pi kb/h} = \frac{Ce^{-2\pi b/h}}{1-e^{-2\pi b/h}}, \tag{2.32}$$

so that the aliasing error is easily controlled by making b/h suitably large.

Many unbounded functions can be treated by minor modifications of the same argument. For example, if

$$|f(x)| \le C_1 + C_2 x \quad \text{for} \quad x \ge t + 2\pi/h \;, \tag{2.33}$$

then

$$\begin{aligned} |e_h(t)| \quad &\le \quad \sum_{k=1}^{\infty} \left[C_1 + C_2 \left(t + \frac{2\pi k}{h} \right) \right] e^{-2\pi kb/h} \\ &\le \quad \frac{C_1 e^{-2\pi b/h}}{1-e^{-2\pi b/h}} + C_2 t \frac{\left(3e^{-2\pi k/h} - e^{-4\pi k/h}\right)}{\left(1-e^{-2\pi kb/h}\right)^2} \\ &\approx \quad C_1 e^{-2\pi b/h} + 3C_2 t e^{-2\pi k/h} \;. \end{aligned} \tag{2.34}$$

A concrete application of the linear form (2.33) and its associated aliasing error bound in (2.33) is to treat the renewal function in Example 1.1.2. Asymptotic results for the renewal function imply that

$$M(t) - At \to B \quad \text{as } t \to \infty$$

for known constants A and B. In the setting of Example 1.1.2, $A = 1/m$ and $B = (\sigma^2 + m^2)/2m^2$, where m and σ^2 are the mean and variance of F; see p. 366 of [41].

Summing the Infinite Series

The remaining task is to calculate $f_h(t)$ in (2.20). We will discuss measures to control the truncation and roundoff errors and then summarize the algorithm at the end of the section.

Summing the infinite series will clearly be an easy task (e.g., by simply truncating) if the transform $\hat{f}(u + iv)$, for u and v real and positive, decays rapidly as $v \to \infty$. Generally speaking, we understand when this will occur, because the tail behavior of the transform $\hat{f}$ depends on the smoothness of the function f. By the Riemann-Lebesgue lemma (e.g., pp. 513–514 of Feller [41]), if f has n integrable derivatives, then $\hat{f}(u + iv)$ is $o(v^{-n})$ as $\nu \to \infty$. Thus the inversion will be easy if f is very smooth, but may be difficult if f is not smooth. The ultimate lack of smoothness is a discontinuity at or near t, which will invariably cause a problem; see Section 14 of Abate and Whitt [7]. When there is initially not enough smoothness, it is often possible to introduce smoothness into the function before performing the inversion; see Section 6 of Abate and Whitt [7].

The infinite series in (2.20) can often be calculated by simply truncating, but a more efficient algorithm can be obtained by applying a summation acceleration method; e.g., see Wimp [82]. An acceleration technique is especially important for coping with slowly decaying tails. As with numerical integration, there are many alternatives. An acceleration technique that has proven to be effective in our context is Euler summation, after transforming the infinite sum into a nearly alternating series. (Euler summation is intended for alternating series, in which successive summands alternate in sign.)

In order to be able to control the roundoff error (discussed below), we also introduce a parameter ℓ, which is a positive integer. The parameter ℓ often can be given the default value $\ell = 1$. We convert (2.20) into a nearly alternating series by letting $h = \pi/\ell t$ and $b = A/2\ell t$. At first we get

$$f_h(t) \equiv f_{A,\ell}(t) = \frac{e^{A/2\ell t}}{\ell t} + \frac{2e^{A/2\ell t}}{\ell t} \sum_{k=1}^{\infty} \hat{f}\left(\frac{A}{2\ell t} + \frac{ik\pi}{\ell t}\right) e^{ik\pi/\ell} \ . \tag{2.35}$$

Next, by algebraic manipulation, we get

$$f_{A,\ell}(t) = \sum_{k=0}^{\infty} (-1)^k a_k(t), \tag{2.36}$$

where

$$a_k(t) = \frac{e^{A/2\ell}}{2\ell t} b_k(t), \quad k \geq 0, \tag{2.37}$$

$$b_0(t) = \hat{f}\left(\frac{A}{2\ell t}\right) + 2\sum_{j=1}^{\ell} Re\left[\hat{f}\left(\frac{A}{2\ell t} + \frac{ij\pi}{\ell t}\right) e^{ij\pi/\ell}\right] \tag{2.38}$$

and

$$b_k(t) = 2\sum_{j=1}^{\ell} Re\left[\hat{f}\left(\frac{A}{2\ell t} + \frac{ij\pi}{\ell t} + \frac{ik\pi}{t}\right) e^{ij\pi/\ell}\right], \quad k \geq 1 \ . \tag{2.39}$$

Also the aliasing error in (2.22) becomes

$$e_h(t) \equiv e_{A,\ell}(t) = \sum_{j=1}^{\infty} e^{-Aj} f\left((1+2j\ell)t\right) \ . \tag{2.40}$$

In (2.36)–(2.40) we have chosen the two parameters b and h to depend on the new parameters A and ℓ and the function argument t. This choice makes the period of the periodic function $2\pi/h = 2\ell t$. This choice makes the series in (2.36) correspond to an eventually alternating series if the real part of $\hat{f}(u+iv)$ are eventually of fixed sign as v gets large. This is what we mean by "nearly alternating."

If $|f(x)| \leq C$ for all $x \geq (1+2\ell)t$, as in probability applications (e.g., $C = 1$ when f is a cdf or ccdf), then the aliasing error in (2.40) is bounded by

$$|e_{A,\ell}(t)| \leq \frac{Ce^{-A}}{1-e^{-A}}, \tag{2.41}$$

which is approximately equal to Ce^{-A} when e^{-A} is small. Note that the aliasing error bound is independent of the parameter ℓ. Hence, to have at most $10^{-\gamma}$ aliasing error when $C = 1$, we let $A = \gamma \log 10$. (With $A = 18.4$, the aliasing error bound is 10^{-8}; with $A = 25.3$, the aliasing error bound is 10^{-11}.)

Remark 1.2.1 It should be evident that formula (2.37) presents problems as t becomes very small, because the prefactor $(e^{A/2\ell})/2\ell t$ approaches infinity as $t \to 0$. Thus the variant of the algorithm in (2.36) should not be used for extremely small t, e.g., for $t < 10^{-2}$. We can of course approximate $f(t)$ by $f(0)$ for small t, and $f(0)$ can be calculated by taking limits using the initial value theorem, which states that if $f(t) \to f(0)$ as $t \to 0$, then $s\hat{f}(s) \to f(0)$ as $s \to \infty$. The small t problem can also be solved "automatically" by scaling; see Choudhury and Whitt [29]. ■

Remark 1.2.2 The form of the aliasing error in (2.40) has important implications for practical implementation. Note that the aliasing error is likely to be smaller for large t if we invert the transform $(1-\hat{f}(s))/s$ of the ccdf $F^c(t) \equiv 1 - F(t)$ than if we invert the transform $\hat{f}(s)/s$ of the cdf $F(t)$. For the ccdf, the aliasing error is about $e^{-A}F^c((1+2\ell)t)$ when $F^c((k+1)t) \ll F^c(kt)$. Hence, for small ccdf values associated with large t, we can often achieve small *relative* aliasing error from (2.40). ■

We now indicate how to apply Euler summation to approximate the infinite series in (2.36). Euler summation can be very simply described as the weighted average of the last m partial sums by a binomial probability distribution with parameters m and $p = 1/2$. (It is not necessary to use $p = 1/2$, but it is usually not worthwhile to search for a better p.) In particular, let s_n be the approximation $f_{A,\ell}(t)$ in (2.36) with the infinite series truncated to n terms, i.e.,

$$s_n = \sum_{k=0}^{n} (-1)^k a_k, \tag{2.42}$$

where t is suppressed in the notation and $a_k \equiv a_k(t)$ is given in (2.37)–(2.39). We apply Euler summation to m terms after an initial n, so that the Euler sum approximation to (2.36) is

$$E(m,n) \equiv E(m,n,t) \equiv \sum_{k=0}^{m} \binom{m}{k} 2^{-m} s_{n+k}, \tag{2.43}$$

for s_n in (2.42). Hence, (2.43) is the binomial average of the terms $s_n, s_{n+1}, \ldots, s_{n+m}$. We typically use $m = 11$ and $n = 38$, increasing n as necessary. The overall computation is specified by (2.37)–(2.39), (2.42) and (2.43).

As in other contexts, the acceleration typically drastically reduces the required computation. The improvement in typical examples is very impressive; it should be tried on simple examples. We present additional supporting theory below. To quickly see how important this acceleration step is, note that the error in simple truncation to s_n can be estimated as the value of last term, a_n. For example, if $a_n = n^{-2}$, then we would need about $n = 10^4$ terms to achieve accuracy to 10^{-8}, whereas with Euler summation it typically suffices to have $n = 50$, as noted above.

In order to *estimate* the error associated with Euler summation, we suggest using the difference of successive terms, i.e., $E(m, n+1) - E(m, n)$. Unlike for the aliasing error, however, we have no simple general bound on the summation error associated with truncating the series at $n + m$ terms and applying Euler summation to average the last m terms. However, we discuss Euler summation further below. Experience indicates that the error estimate $E(m, n + 1) - E(m, n)$ is usually accurate. Moreover, under regularity conditions, it is also an upper bound. However, it can happen that the error estimate is optimistic, as indicated in Section 11 of [7].

Practically, the accuracy can be verified by performing the same computation again with different parameters, e.g., changing, m, n, ℓ and A. Note that changing ℓ or A produces a very different computation, so that we obtain an accuracy check by doing the calculation for two different choices of the parameter pair (ℓ, A).

Roundoff Error

In the setting of (2.41), we can obviously make the aliasing error arbitrarily small by choosing A sufficiently large. However, with limited precision (e.g., with double precision, usually 14–16 digits), we are constrained from choosing A too large, because the prefactor $(e^{A/2\ell})/2\ell t$ in (2.37) can then become very large, causing roundoff error. Given that high precision is often readily available, roundoff error should not to be regarded as a serious problem. From that perspective, our analysis below shows why higher precision may be needed. In most probability applications, an aliasing error of 10^{-8} is more than adequate, so that the required A does not produce too much roundoff error even with double precision.

We now indicate how to control both the roundoff and aliasing errors. The key to controlling the roundoff error is the fact that the aliasing error bound in (2.41) depends on A but not on ℓ. Suppose that we are able to compute the $b_k(t)$ in (2.39) with an error of about 10^{-m}. (This quantity is typically of the order of machine precision.) Then, after multiplying by the prefactor $(e^{A/2\ell})/2\ell t$ in (2.37), the final "roundoff" error in $f_{A,\ell}(t)$ from (2.36) will be about $10^{-m}(e^{A/2\ell})/2\ell t$. Since the roundoff error estimate is increasing in A, while the aliasing error estimate e^{-A} in (2.41) (assuming that $C = 1$) is decreasing in A, the maximum of the two error estimates is minimized where the two error estimates are equal. The estimated total error should thus be approximately minimized at this point. Thus we find an appropriate value for the parameter A by solving the equation

$$e^{-A} = 10^{-m}(e^{A/2\ell})/2\ell t, \tag{2.44}$$

which yields

$$A = \left(\frac{2\ell}{2\ell+1}\right)(m \log 10 + \log 2\ell t) \ . \tag{2.45}$$

Ignoring the final $\log 2\ell t$ term in (2.45), we see that (2.45) implies that the final error estimate is

$$e^{-A} \approx 10^{2\ell m/(2\ell+1)}, \tag{2.46}$$

which means that we lose a proportion of $1/(2\ell+1)$ of our m-digit precision due to multiplying by the prefactor. We obtain higher precision by increasing ℓ, but at a computational expense, because the computational effort is proportional to ℓ; see (2.38) and (2.39).

With $\ell = 1$, we lose about one third of the machine precision due to roundoff, and with $\ell = 2$, we lose about one fifth of the machine precision. If we take the typical double-precision value of $m = 14$ and let $t = 1$ and $\ell = 1$, then the error estimate (2.46) becomes about $10^{-9.3}$ and with $\ell = 2$ the error estimate is about $10^{-11.2}$.

From this analysis, we see that it is difficult to compute quantities that are smaller than or the same order as machine precision. This difficulty can be addressed by scaling; see [29].

We now summarize the variant of the Fourier-series method described above.

Algorithm Summary

Based on the parameters A, ℓ, m and n (e.g., $A = 19$, $\ell = 1$, $m = 11$ *and* $n = 38$) *and the function argument t, approximately compute $f(t)$ by* (2.43), (2.42) *and* (2.37)–(2.39). *The aliasing error is given by* (2.40). *If $|f(t)| \leq C$ for all $t \geq 0$, then the aliasing error is bounded by* (2.41) *and is approximately Ce^{-A}. The overall error* (*including the Euler summation error*) *is estimated by $E(m, n+1) - E(m, n)$ using* (2.43). *The overall error is also estimated by performing the computation with two different parameter pairs* (A, ℓ), e.g., $(18, 1)$ *and* $(19, 1)$ *or* $(18, 1)$ *and* $(18, 2)$.

We reiterate that we give no a priori error bound for the entire algorithm, primarily because we have no simple general bound on the error from Euler summation. (Under additional assumptions, bounds are possible, though; see below.) However, when the algorithm is applied, we invariably can see the achieved accuracy by comparing the results of computations with different parameters. If the achieved accuracy is not sufficient, then we suggest: first, increasing n; second, increasing the roundoff control parameter ℓ; and third, considering convolution smoothing, as described in Section 6 of [7].

This section is based on Abate and Whitt [7, 10] and Choudhury, Lucantoni and Whitt [24]. The Fourier-series method for numerically inverting Laplace transforms was proposed by Dubner and Abate [37]. The use of the parameter ℓ for roundoff error control was proposed for the case $l = 2$ by Durbin [39] and was also used more generally by Kwok and Barthez [59] and Choudhury, Lucantoni and Whitt [24]. The use of Euler summation in this context was proposed by Simon, Stroot and Weiss [75]. The Fourier-series method with Euler summation was later developed independently in Japan by Hosono [45], [46], [47].

There is an extensive body of related literature; for further discussion see Abate and Whitt [7]. We have presented only one variant of the one method for numerically inverting Laplace transforms. There are other variants of the Fourier-series method and there are entirely different methods. The Jagerman-Stehfest and Gaver-Stehfest procedures in Section 8 of Abate and Whitt [7] are based on the Post-Widder formula and its discrete analog in Theorems 2 and 3. The Laguerre-series algorithm in Abate, Choudhury and Whitt [5] is based on the Laguerre-series representation in Theorem 4. See these sources for additional background. See Davies and Martin [33] for a (somewhat dated) comparison of alternative methods. See Choudhury and Whitt [27] for a description of the Q^2 performance analysis tool based on numerical transform inversion.

1.2.3 More on Euler Summation

Since Euler summation proves to be so effective in accelerating convergence in the Fourier-series algorithm, it is interesting to examine it in more detail. This section is somewhat more technical than others, and so might be skipped. This section draws upon Johnsonbaugh [51], Section 6 of [7] and O'Cinneide [66]. For an advanced treatise on acceleration techniques, see Wimp [82].

A good way to understand how Euler summation performs is to try it (apply (2.42) and (2.43)) with some simple examples, e.g., $a_k = k^{-p}$ for $p > 0$. Because of the alternating series structure, with simple truncation the maximum of the errors $|s_n - s_\infty|$ and $|s_{n-1} - s_\infty|$ must be at least $a_n/2$. On the other hand, the averaging associated with Euler summation can lead to amazing improvement.

We now analyze the performance more carefully. First note that if $a_n \geq a_{n+1} \geq 0$ for all n in (2.42), then

$$s_{2n+2} \geq s_{2n} \geq s_\infty \geq s_{2n+1} \geq s_{2n-1} \tag{2.47}$$

and

$$|s_\infty - s_n| \leq a_n \tag{2.48}$$

for all $n \geq 1$.

Next observe that we can write $E(m, n)$ itself in the form of an alternating series by changing the order of summation, i.e.,

$$E(m,n) = \sum_{j=0}^{m} \binom{m}{j} 2^{-m} \sum_{k=0}^{n} (-1)^{k+j} a_{k+j} = \sum_{k=0}^{n} (-1)^k b_k, \tag{2.49}$$

where

$$b_k = \sum_{j=0}^{m} \binom{m}{j} 2^{-m} (-1)^j a_{k+j} \ . \tag{2.50}$$

Of course, in general b_k in (2.50) need not be of constant sign. However, from (2.48), we see that if $b_k \geq b_{k+1} \geq 0$ for all k, then

$$|E(m,n) - s_\infty| \leq b_n = |E(m,n) - E(m,n-1)| \tag{2.51}$$

and our error estimate $|E(m,n) - E(m,n-1)|$ becomes a bound on the actual error.

To see the quantitative benefits of Euler summation, it is useful to consider (2.42) and (2.43) with $a_n = (c+n)^{-k}$ for some constant c and positive integer k. (For any c, a_n is positive and decreasing for $n > -c$.) We can exploit the identity

$$\frac{1}{(c+n)^k} = \frac{1}{(k-1)!} \int_0^\infty e^{-(c+n)x} x^{k-1} dx, \tag{2.52}$$

which follows from the form of the gamma distribution, to obtain the following result. See O'Cinneide [66].

Theorem 6 *If $a_n = (c+n)^{-k}$, then the summands b_k in (2.50) indeed are positive and decreasing when $c+k > 0$, so that the bound (2.51) holds for n suitably large and*

$$|E(m,n) - E(m,n-1)| \leq \frac{(m+k-1)!}{2^m (k-1)! (c+n)^{m+k}} \ . \tag{2.53}$$

Hence, when $a_n = (c+n)^{-k}$, from (2.48) the error using s_n in (2.42) (direct truncation) is bounded by $(c+n)^{-k}$, whereas the error from Euler summation is bounded above by $C(c+n)^{-(m+k)}$ from (2.51) and (2.53). Asymptotically, as $n \to \infty$ for fixed m, the rate of convergence improves from n^{-k} to $n^{-(m+k)}$ when we use Euler summation.

We now show that in the inversion algorithm we actually have a_n asymptotically of the form $C(c+n)^{-k}$ for sufficiently large n, under mild smoothness conditions, so that the analysis above is directly applicable to the inversion problem. For this purpose, let

$$\mathrm{Re}(f)(u+iv) = \int_0^\infty e^{-ut} \cos vt\, f(t) dt = \int_0^\infty \cos vt\, g(t) dt \ , \tag{2.54}$$

where g is the damped function $g(t) \equiv e^{-ut} f(t)$.

If f is twice continuously differentiable, then so is g. If $g(\infty) = g'(\infty) = 0$ (which is satisfied when $f(\infty) = f'(\infty) = 0$), then we can apply integration by parts in (2.54) to get

$$\mathrm{Re}(\hat{f})(u+iv) = \frac{uf(0) - f'(0)}{v^2} - \frac{1}{v^2} \int_0^\infty \cos vt\, g''(t) dt \ . \tag{2.55}$$

If in addition g'' is integrable, then we can apply the Reimann-Lebesgue lemma to deduce that

$$\text{Re}(\hat{f})(u+iv) = \frac{uf(0) - f'(0)}{v^2} + o\left(\frac{1}{v^2}\right) \quad \text{as} \quad v \to \infty \; . \tag{2.56}$$

Similarly,

$$\begin{aligned} \text{Im}(f)(u+iv) &= \int_0^\infty \sin vt\, g(t)dt \\ &= \frac{-f(0)}{v} + \frac{1}{v}\int_0^\infty \cos vt\, g'(t)dt \\ &= \frac{-f(0)}{v} + o\left(\frac{1}{v}\right) \quad \text{as} \quad v \to \infty \end{aligned} \tag{2.57}$$

provided that g' is integrable.

From (2.56) and (2.56), we see that $Re(\hat{f})(u+iv)$ and $\text{Im}(\hat{f})(u+iv)$ will indeed be eventually of constant sign as v increases (including the cases in which $u = f'(0)$ and $f(0) = 0$, which require further analysis). From (2.39) we see that the summands a_n in the infinite series to be computed are asymptotically of the form $C(c+n)^k$ for $k = 2$ (real part) or $k = 1$ (imaginary part), where C and c are known constants. Thus the analysis above shows that Euler summation will be effective provided that the function f indeed has the smoothness and integrability properties required for (2.56) and (2.56).

Furthermore, with extra smoothness, we can repeat the integration by parts argument in (2.55) and obtain a more detailed analysis. Explicit error bounds were first determined by O'Cinneide [66]. The analysis shows that

$$|E(2m,n) - s_\infty| \sim Cn^{-(2m+2)} \text{ as } n \to \infty \tag{2.58}$$

if g has $2m+2$ derivatives with $g^{(k)}(\infty) = 0$ for $0 \leq k \leq 2m+1$ and $g^{(2m+2)}$ is integrable.

Example 1.2.1 To see that the desired properties supporting Euler summation do *not* hold in general, consider the ccdf of a unit probability point mass at x, with LT

$$\hat{F}^c(s) = \frac{1 - e^{-sx}}{s}, \tag{2.59}$$

The ccdf F^c is constant except for the one jump at x. Note that

$$\text{Re}(\hat{F}^c)(u+iv) = \frac{u(1 - e^{-ux}\cos vx) + v \sin vx}{u^2 + v^2} \tag{2.60}$$

From (2.60), we see that, for every u, there is no v_0 such that $Re(\hat{F}^c)(u+iv)$ is of constant sign for $v \geq v_0$. Moreover, in this case Euler summation does not provide improvement over simple truncation; see Table 5 of Abate and Whitt [7]. ∎

Example 1.2.2 To see that Euler summation does not perform well on all alternating series, consider the series

$$\sum_{k=1}^{\infty}(-1)^k(\sin kx)^2/k = \frac{\log(1/\cos x)}{2} \quad \text{for} \quad 0 < x < \pi/2 \; .$$

The performance of Euler summation degrades as x increases. For example, the error in $E(11, 30)$ is of order 10^{-9}, 10^{-3} and 10^{-1} for $x = 0.1$, 1.0 and 1.5, respectively. If $x = 1.5$, s_{41} had an error of order 10^{-2}, which is better.

1.2.4 Generating Functions

The Fourier-series method also applies to invert generating functions. Suppose that $\{q_k : k \geq 0\}$ is a sequence of complex numbers with generating function

$$\mathcal{G}(q) \equiv \hat{q}(z) \equiv \sum_{k=0}^{\infty} q_k z^k, \tag{2.61}$$

where z is a complex number, and we want to calculate q_k.

At the outset, note that this problem should be regarded as easier than the Laplace transform inversion, because the coefficients q_k in (2.61) can be recovered from the generating function $\hat{q}$ by repeated differentiation and evaluation at $z = 0$. This means that numerical differentiation techniques and symbolic mathematical software are often readily applicable. However, it is often difficult to achieve desired accuracy with numerical differentiation techniques, especially for large n. It is also difficult to invoke symbolic mathematical software when the generating function is only expressed implicitly. Fortunately, in this setting numerical inversion is also a viable alternative. We will apply the numerical inversion algorithm for generating functions of complex numbers in Section 5 to calculate normalization constants in product-form closed queueing networks. In that context, it is important to allow q_k to be complex numbers in order to invert multidimensional transforms using iterative one-dimensional inversion.

Just as for Laplace transforms, numerical inversion of generating functions can be based on an inversion integral. The Cauchy contour integral plays the role for generating functions that the Bromwich contour integral plays for Laplace transforms. The following parallels Theorem 1.

Theorem 7 *Suppose that $\{q_k : k \geq 0\}$ is a sequence of complex numbers with $|q_k| \leq Kb^k$ for all $k \geq 1$, where K and b are positive constants. Then*

$$\begin{aligned} q_k &= \frac{1}{2\pi i}\int_{\Gamma}\frac{\hat{q}(z)}{z^{n+1}}dz, \qquad (2.62)\\ &= \frac{1}{2\pi r^k}\int_0^{2\pi}\hat{q}(re^{iu})e^{-iku}du \; . \end{aligned}$$

where the contour Γ is a circle centered at the origin with radius r, where r is less than the radius of convergence b^{-1}.

Paralleling the inversion of Laplace transforms, a good way to proceed is to apply the trapezoidal rule in the integral in (2.62). Just as in Section 1.2.1, we can apply the Fourier series method to determine the discretization error associated with the trapezoidal rule. The following result thus parallels Theorem 5. Note that $q_0 = \hat{q}(0)$, so it suffices to produce an algorithm to calculate q_k for $k \geq 1$.

Theorem 8 *Under the assumptions of Theorem 7, for $0 < r < b^{-1}$ and $k \geq 1$,*

$$q_k = q_k^a - e_a, \tag{2.63}$$

where the trapezoidal-rule approximation *to (2.62) is*

$$q_k^a = \frac{1}{2k\ell r^k} \sum_{j=1}^{2k} (-1)^j a_j \tag{2.64}$$

with

$$a_j \equiv a_j(k, \ell, r) = \sum_{j_1=0}^{\ell-1} e^{-\pi i j_1/\ell} \hat{q}(re^{\pi i (j_1 + \ell j)/\ell k}), \quad 1 \leq j \leq 2k, \tag{2.65}$$

and the associated discretization or aliasing error *is*

$$e_a = \sum_{j=1}^{\infty} q_{k(1+2j\ell)} r^{2jk\ell} \ . \tag{2.66}$$

Proof. To establish (2.64) and (2.66), we form the damped sequence $a_k = q_k r^k$ for $0 < r < b^{-1}$ and let $a_k = 0$ for $k < 0$. Then we apply the discrete Fourier transform to (calculate the discrete Fourier series of) the aliased sequence

$$a_k^p = \sum_{j=-\infty}^{\infty} a_{k+jm} \ . \tag{2.67}$$

Note that $\{a_k^p\}$ is a periodic sequence with period m. The assumptions imply that $\sum_{k=-\infty}^{\infty} |a_k| < \infty$, so that the series in (2.67) converges.

In preparation for the next step, let the Fourier transform of the sequence $\{a_k\}$ be

$$\phi(u) = \sum_{k=-\infty}^{\infty} a_k e^{iku} = \hat{q}(re^{iu}) \ , \tag{2.68}$$

which has inverse

$$a_k = \frac{1}{2\pi} \int_0^{2\pi} \phi(u) e^{-iku} du \ . \tag{2.69}$$

Now the *discrete Fourier transform* of the periodic sequence $\{a^p_k\}$ is

$$\begin{aligned}
\hat{a}^p_k &= \frac{1}{m}\sum_{j=0}^{m-1} a^p_j e^{i2\pi kj/m} \\
&= \frac{1}{m}\sum_{j=0}^{m-1}\sum_{\ell=-\infty}^{\infty} a_{j+\ell m} e^{i2\pi jk/m} \\
&= \frac{1}{m}\sum_{j=-\infty}^{\infty} a_j e^{i2\pi jk/m} = \frac{1}{m}\phi(2\pi k/m), \qquad (2.70)
\end{aligned}$$

Now apply the inversion formula for discrete Fourier transforms to obtain

$$a^p_k = \sum_{j=0}^{m-1} \hat{a}^p_j e^{-i2\pi jk/m} = \frac{1}{m}\sum_{j=0}^{m-1} \phi(2\pi j/m) e^{-i2\pi jk/m} \ . \qquad (2.71)$$

The combination of (2.67) and (2.71) is the *discrete Poisson summation formula*

$$\sum_{j=-\infty}^{\infty} a_{k+jm} = \frac{1}{m}\sum_{j=0}^{m-1} \phi(2\pi j/m) e^{-i2\pi jk/m} \ . \qquad (2.72)$$

Putting the term for $k=0$ on the left, we obtain for integers k and $m>0$

$$a_k = \frac{1}{m}\sum_{j-0}^{m-1} \phi(2\pi j/m) e^{-i2\pi jk/m} - \sum_{\substack{j=-\infty \\ j\neq 0}}^{\infty} a_{k+jm} \ . \qquad (2.73)$$

Finally, we obtain (2.64) and (2.66) by setting $m = 2k\ell$ in (2.73). Note that $a_{k+jm} = 0$ for $j<0$ with this choice. In particular,

$$q_k r^k \equiv a_k = \frac{1}{2k\ell}\sum_{j=0}^{2k\ell-1} \phi(\pi j/k\ell) e^{-i\pi j/\ell} - \sum_{j=1}^{\infty} q_{k(1+2j\ell)} r^{k+2jk\ell} \ , \qquad (2.74)$$

so that

$$q_k = \frac{1}{2k\ell r^k}\sum_{j=0}^{2k\ell-1} \hat{q}(re^{i\pi j/k\ell}) e^{-i\pi j/\ell} - \sum_{j=1}^{\infty} q_{k(1+2j\ell)} r^{2jk\ell} \qquad (2.75)$$

which gives (2.63)–(2.66). ∎

Note that if $|q_n| \le C$ for all $n \ge (1+2\ell)k$, then

$$|e_a| \le \sum_{j=1}^{\infty} C r^{2jk\ell} \le \frac{Cr^{2k\ell}}{1-r^{2k\ell}}, \qquad (2.76)$$

which is approximately $Cr^{2k\ell}$ when $r^{2k\ell}$ is suitably small. To make the aliasing error about $10^{-\eta}$ when $C = 1$, we let $r = 10^{-\eta/2k\ell}$, where ℓ is chosen to control the roundoff error (the default value being $\ell = 1$). In typical applications we may choose $\eta = 8$. However, in some applications there is no bound on $|q_k|$. We then may need to apply an additional scaling algorithm, as illustrated in Section 1.5.

It is possible to reduce the computations by a factor of 2 if q_k is real-valued by using the fact that $\hat{q}(\overline{z}) = \overline{\hat{q}(z)}$. Then (2.64) can be replaced by

$$q_k^a = \frac{1}{2k\ell r^k} \sum_{j=1}^{2k} (-1)^j Re(a_j) \tag{2.77}$$

$$= \frac{1}{2k\ell r^k} \left(a_0(k,\ell,r) + (-1)^k a_k(k,\ell,r) + 2\sum_{j=1}^{k-1} (-1)^j Re\left(a_j(k,\ell,r)\right) \right)$$

for $a_j \equiv a_j(k,\ell,r)$ in (2.65).

Unlike for the Laplace transforms in Section 1.2.1, the series (2.64) is finite, so that there is no need to approximate an infinite series. Hence there is no need to apply Euler summation with generating functions. However, if the index k is very large, then it may still be advantageous to apply Euler summation to accelerate convergence of the finite sum (2.64), as we illustrate in the closed queueing network example in Section 1.5.4. The representation (2.64) has been chosen to be nearly an alternating series, so that it is directly in the form to apply Euler summation, just as in (2.43).

The roundoff control procedure is essentially the same as for Laplace transforms. If we can compute the sum in (2.64) without the prefactor $(2k\ell r^k)^{-1}$ with a precision (error estimate) of 10^{-m}, then the roundoff error after multiplying by the prefactor will be approximately $(2k\ell r^k)^{-1}10^{-m}$. Since the roundoff error estimate is decreasing in r as r approaches 1 from below, while the aliasing error estimate is increasing in r, the maximum of the two error estimates will be minimized when the two estimates are equal. Thus the total error should be approximately minimized at this point. This leads to the equation

$$r^{2k\ell} = (2k\ell r^k)^{-1} 10^{-m}, \tag{2.78}$$

Assuming that we can ignore the term $2k\ell$ on the right in (2.78), we get $r^{(2\ell+1)k} \approx 10^{-m}$ or

$$r \approx 10^{-m/(2\ell+1)k} \quad \text{and} \quad r^{2\ell k} \approx 10^{-2\ell m/(2\ell+1)} \; . \tag{2.79}$$

As in Section 1.2.1, this analysis shows that we approximately lose a proportion $1/(2\ell+1)$ of our precision due to roundoff error. If $m = 12$, then we can achieve an overall error estimate of about 10^{-8} by setting $\ell = 1$ and $r = 10^{-4/k}$. By increasing ℓ, we can get close to 10^{-m} but never below it. To accurately calculate smaller numbers than 10^{-m} we need to apply scaling; see Section 1.5.3.

We now summarize the algorithm.

Generating Function Algorithm Summary.. *Based on the desired sequence index k and the parameters ℓ and r (e.g., $\ell = 1$ and $r = 10^{-4/k}$ corresponding to $\eta = 8$), approximately compute q_k from its generating function $\hat{q}$ in* (2.61) *by* (2.64)*. If q_k is real-valued, then replace* (2.64) *by* (2.77)*. The aliasing error is* (2.66)*. If $|q_k| \leq C$ for all k, then the aliasing error is bounded by* (2.76) *and approximated by $Cr^{2k\ell}$ (which is 10^{-8} when $C = 1$, $\ell = 1$ and $r = 10^{-4/k}$). If the index k is large, it may be advantageous to apply Euler summation using (2.43).*

The algorithm in this section is based on Abate and Whitt [7, 8] and Choudhury, Lucantoni and Whitt [24], but as they indicate, there is a substantial body of related literature. Nevertheless, surprisingly, this algorithm was not well known.

1.3 TRANSIENT CHARACTERISTICS OF THE ERLANG LOSS MODEL

This section contains our first nontrivial example illustrating how the numerical inversion of Laplace transforms can be applied. We apply the Fourier-series algorithm in Section 1.2.1, but we could also apply other algorithms, such as the Laguerre-series algorithm based on Theorem 4.

Given that inversion algorithms are available, typically the major challenge in applications is efficiently computing the required Laplace transform values. Fortunately, much previous work in applied probability has been devoted to deriving transforms of random quantities of interest. As indicated at the outset, excellent examples are the queueing books by Takács [78] and Cohen [30]. Nevertheless, computing transform values can be a challenge. Sometimes transforms are only available as integrals, as in the Pollaczek contour integral expression for the GI/G/1 waiting time, to be discussed in Section 1.4. On other occasions, transforms are only available implicitly, as in Kendall functional equation for the M/G/1 busy period.

In this section we consider the classical Erlang loss model, i.e., the M/M/c/0 system with Poisson arrival process, exponential service times, c servers and no extra waiting space, where blocked calls are lost. We let the individual service rate be 1 and the arrival rate (which coincides with the offered load) be a. The way to compute steady-state characteristics for this model is very well known, but that is not the case for transient (time-dependent) characteristics. Transience arises by considering arbitrary fixed initial states. We show how to compute several transient characteristics by numerical transform inversion. This section draws on Abate and Whitt [11].

Before starting, we mention other applications of numerical transform inversion to calculate transient characteristics of queueing models. The M/G/1 busy period distribution is treated in Abate and Whitt [9]. The time-dependent queue-length and workload processes in the M/G/1, BMAP/G/1 and $M_t/G_t/1$ queues are treated in Choudhury, Lucantoni and Whitt [24], Lucantoni, Choudhury and Whitt [62], and Choudhury, Lucantoni and Whitt [26], respectively.

Both steady-state and time-dependent distributions in polling models are calculated in Choudhury and Whitt [28]. The time-dependent distributions of semi-Markov processes are calculated in Duffield and Whitt [38].

Here we develop algorithms for computing four quantities in the M/M/c/0 model: the time-dependent blocking probability starting at an arbitrary initial state i, i.e., the transition probability

$$P_{ic}(t) \equiv P(N(t) = c|N(0) = i) \ , \tag{3.80}$$

where $N(t)$ is the number of busy servers at time t; the complementary cumulative distribution function (ccdf) $F^c_{ic}(t)$ of the time T_{ic} all servers first become busy starting at an arbitrary initial state i; i.e., where

$$F^c_{ic}(t) \equiv 1 - F_{ic}(t) \equiv P(T_{ic} > t) \ , \tag{3.81}$$

and

$$T_{ic} \equiv \inf\{t \geq 0 : N(t) = c|N(0) = i\} \ ; \tag{3.82}$$

the time-dependent mean

$$M_i(t) \equiv E(N(t)|N(0) = i) \ ; \tag{3.83}$$

and the (stationary) covariance function

$$\begin{aligned} R(t) &\equiv Cov(N_s(u), N_s(u+t)) \\ &= E(N_s(u)N_s(u+t)) - EN_s(u)EN_s(u+t) \ , \end{aligned} \tag{3.84}$$

where $\{N_s(t) : t \geq 0\}$ is a stationary version of $\{N(t) : t \geq 0\}$, i.e., where $N_s(u)$ in (3.84) is distributed according to the steady-state distribution

$$\pi_j \equiv P(N_s(u) = j) = \frac{a^j/j!}{\sum_{k=0}^{c} a^k/k!} \ . \tag{3.85}$$

We also show how to compute these quantities for very large systems by performing computations for moderately sized systems and using scaling based on the established heavy-traffic limit in which $(N^{(a)}(t)-a)/\sqrt{a}$ converges to the reflected Ornstein-Uhlenbeck (ROU) process as $a \to \infty$ with $i(a) - a \sim \gamma_1\sqrt{a}$ and $c(a) - a \sim \gamma_2\sqrt{a}$, where $f(a) \sim g(a)$ means that $f(a)/g(a) \to 1$ as $a \to \infty$; see p. 177 of Borovkov [17] and Srikant and Whitt [77]. The ROU process is the ordinary OU process modified to have a reflecting upper barrier. The OU process is a diffusion process with constant diffusion coefficient and proportional state-dependent drift, i.e., with drift $-\delta x$ in state x. However, we will not focus on the ROU process; we will only use the scaling.

For example, suppose that we want to compute $P_{ic}(t)$ for some large a such as $a = 10^8$, where c and i are allowed to depend on a via $c(a) = \lfloor a + \sqrt{a} \rfloor$ and $i(a) = \lfloor a - 2\sqrt{a} \rfloor$, with $\lfloor x \rfloor$ being the greatest integer less than or equal to x. We will write $P^{(a)}_{i(a)c(a)}$ to indicate the dependence upon the offered load a. The heavy-traffic limit implies that $P_{i(a)c(a)}^{(a)}/B(c(a),a)$ should be approximately independent of a, where $B(c(a),a) \equiv P_{i(a)c(a)}^{(a)}(\infty) \equiv \pi_{c(a)}^{(a)}$ is the steady-state Erlang blocking probability, which is known to have the asymptotic relation

$$B(c(a),a) \sim \frac{1}{\sqrt{a}} \frac{\phi(\gamma)}{\Phi(-\gamma)} \quad \text{as} \quad a \to \infty \ , \tag{3.86}$$

where ϕ is the density and Φ is the cdf of a standard (mean 0, variance 1) normal distribution and γ is the limit of $(a-c)/\sqrt{a}$; see Jagerman [48], Whitt [81], and (15) of Srikant and Whitt [77]. Hence, we can compute $P_{i(10^8)c(10^8)}^{(10^8)}(t)$ approximately using results for $a = 400$ as follows:

$$\begin{aligned} P_{i(10^8)c(10^8)}^{(10^8)}(t) &\approx \frac{B(10^8+10^4,10^8)}{B(400+20,400)} P_{i(400)c(400}^{(400)}(t) \\ &\approx \left(\frac{20}{10^4}\right) P_{i(400)c(400}^{(400)}(t) \ , \end{aligned} \tag{3.87}$$

with $i(a) = \lfloor a - 2\sqrt{a} \rfloor$ and $c(a) = \lfloor a + \sqrt{a} \rfloor$ in each case; e.g., $i(a) = 360$ for $a = 400$ and $i(a) = 10^8 - 2(10^4)$ for $a = 10^8$. We will show the effectiveness of the scaling in numerical examples.

The algorithms here are based on computing the Laplace transforms of these quantities with respect to time and then applying the Fourier-series method. For the most part, algorithms for computing the transforms are available in the literature. In particular, an algorithm to calculate the Laplace transform of $P_{ij}(t)$ is given on pp. 81–84 of Riordan [74], but it does not seem to be widely known. Formulas for the Laplace transform of the mean and the covariance are given in Beneš [14], [15] and Jagerman [49], but the formula for the covariance transform in (15) on p. 209 of [15] and (15) on p. 136 of [14] has a sign error. Abate and Whitt [11] derived a new formula for the covariance transform, given below in Theorem 3.2.

The numerical inversion algorithm is an alternative to the spectral expansion described in Beneš [14], [15] and Riordan [74]. The spectral expansion is efficient for computing values at many time points, because the eigenvalues and eigenvectors need only be computed once. However, the inversion algorithm is also fast, and remarkably simple.

The numerical inversion algorithm is also an alternative to the numerical solution of a system of ordinary differential equations (ODEs), which is discussed here in Chapter 2. Numerical solution of ODEs has the advantage that it applies to time-dependent models as well as the transient behavior of stationary models with nonstationary initial conditions. However, when the numerical inversion algorithm applies, it has the advantage that it can produce calculations at any desired t without having to compute the function over a large set of time points in the interval $[0,t]$.

Finally, asymptotic formulas can serve as alternatives to exact numerical algorithms in the appropriate asymptotic regimes. Such asymptotic formulas are given in Mitra and Weiss [63] and Knessl [57]. These asymptotic formulas are very attractive when they are both simple and sufficiently accurate, but many of the asymptotic formulas are not simple. Then they properly should be viewed as alternatives to numerical algorithms. It appears that the numerical algorithm here is much more accurate than the asymptotic approximations.

1.3.1 Time-Dependent Blocking Probabilities

As shown on pp. 81–84 of Riordan [74], the Laplace transform

$$\hat{P}_{ij}(s) \equiv \int_0^\infty e^{-st} P_{ij}(t)dt \tag{3.88}$$

is easily computed recursively, exploiting relations among the Poisson-Charlier polynomials. Since Riordan was not developing a numerical inversion algorithm, he was not interested in a numerical algorithm for computing the transform, so it is not highlighted, but it is there. The key relation is (8) on p. 84 of [74] using the recursions (3) and (4). The determinant $|D|$ in (8) is evaluated in (6).

We will focus on $P_{ij}(t)$ only for $j = c$, but the general case can be computed as well. To express the result for $P_{ic}(t)$, let

$$d_n \equiv d_n(s,a) = (-1)^n C_n(-s,a) \ , \tag{3.89}$$

where s is a complex variable and $C_n(s,a)$ are the Poisson-Charlier polynomials; i.e.,

$$d_n = \frac{1}{a^n} \sum_{k=0}^{n} \binom{n}{k} s(s+1)\dots(s+k-1)a^{n-k} \ ; \tag{3.90}$$

e.g.,

$$d_0 = 1 \ , \quad d_1 = \frac{1}{a}(a+s) \tag{3.91}$$

$$d_2 = \frac{1}{a^2}(a^2 + (2a+1)s + s^2) \ . \tag{3.92}$$

We now specify the algorithm for computing $\hat{P}_{ic}(s)$ for any desired i, c and complex s. We use the polynomials d_n, but we do not compute them via (3.90); instead we compute them recursively. Our algorithm follows from the recursive relations in Riordan [74].

Theorem 9 *The Laplace transform of the time-dependent blocking probability is*

$$\hat{P}_{ic}(s) = d_i \hat{P}_{0c}(s) \ , \tag{3.93}$$

where

$$\hat{P}_{0c}(s) = \frac{1}{a(d_{c+1} - d_c)} \ , \tag{3.94}$$

d_0 and d_1 are given in (3.91) and

$$d_{n+1} = (1 + \frac{n}{a} + \frac{s}{a})d_n - \frac{n}{a}d_{n-1} \ , \quad n \geq 1 \ . \tag{3.95}$$

Since $\{N_s(t) : t \geq 0\}$ is a stationary reversible process, e.g., see p. 26 of Keilson [53], $\pi_i P_{ic}(t) = \pi_c P_{ci}(t)$. Hence, we can also calculate $P_{ci}(t)$ directly from $P_{ic}(t)$ by

$$P_{ci}(t) = (\pi_i/\pi_c)P_{ic}(t) = \frac{a^i c!}{a^c i!}P_{ic}(t) \ . \tag{3.96}$$

As indicated in the introduction, $P_{ic}^{(a)}(t)/B(c,a)$ should be approximately independent of a provided that $i \equiv i(a) \approx a + \gamma_1\sqrt{a}$ and $c \equiv c(a) \approx a + \gamma_2\sqrt{a}$ for arbitrary constants γ_1 and γ_2 (which we think of as being in the interval $[-5, 5]$). To calculate the Erlang blocking probability $B(c,a)$, we use the well known recurrence

$$B(c,a) = \frac{1}{1 + \frac{c}{aB(c-1,a)}} \ . \tag{3.97}$$

The Erlang blocking probability B is related to the polynomial d_n by $d_n(1,a) = 1/B(n,a)$. The recurrence relation (3.97) itself follows directly from another recurrence relation for d_n, namely,

$$d_n(s,a) = d_n(s+1,a) - \frac{n}{a}d_{n-1}(s+1,a) \ ; \tag{3.98}$$

see Corollary 3 on p. 549 of Jagerman [48]. The polynomials d_n are related to the sigma functions used in Beneš [15] and other early references by $\sigma_s(n) = a^n d_n(s,a)/n!$

We now illustrate the algorithm with a numerical example. We will consider five cases with five different values of a, ranging from $a = 100$ to $a = 10,000$, where $\gamma_1 = (i(a) - a)/\sqrt{a} = -3$ and $\gamma_2 = (c(a) - a)/\sqrt{a} = 2$. The five cases with steady-state performance measures are displayed in Table 3.1. Let M and V be the mean and variance of the steady-state number of busy servers, i.e., $M = a(1 - B)$ and

$$V = M - aB(c - M) = M - aB(c - a) - (aB)^2 \ . \tag{3.99}$$

The effectiveness of the scaling is shown in Table 3.1 through the values of $\sqrt{a}B$ and V/a, which are nearly independent of a.

Numerical values of $P_{i(a),c(a)}^{(a)}(t)/B(c(a),a)$ for nine time points are displayed in Table 3.2. The values of B are computed from (3.97), while the values of $P_{i(a)c(a)}^{(a)}(t)$ are computed by the Fourier-series method as in Section 1.2.1 after computing the transform values by the algorithm in Theorem 9. The inversion parameters were set so that the transform was computed at 40 values of complex s in each case. For the largest case, $a = 10^4$, the computation took about two minutes using UBASIC on a PC. (See [7] for more on UBASIC.) As in Table 3.1, the effectiveness of the scaling in Table 3.2 is evident in the similarity of values in each row.

Table 3.1 The five cases ($\gamma_1 = -3$ and $\gamma_2 = 2$).

cases	c	a	i	B	M	V	$\sqrt{a}B$	V/a
I	120	100	70	.0056901	99.43	87.73	.056901	.877271
II	440	400	340	.0028060	398.88	352.72	.056120	.881806
III	960	900	810	.0018613	898.33	795.01	.055840	.883341
IV	2600	2500	2350	.0011122	2497.22	2211.45	.055608	.884579
V	10200	10000	9700	.0005543	9994.46	8855.13	.055430	.885513

Table 3.2 Values of $P^{(a)}_{i(a),c(a)}(t)/B(c(a),a)$ in the five cases of Table 3.1.

time	$I(a = 100)$	$II(a = 400)$	$III(a = 900)$	$IV(a = 2,500)$	$V(a = 10,000)$
1.0	.038920	.040993	.041755	.042435	.042836
1.5	.220241	.225617	.227479	.227581	.230147
2.0	.459358	.464459	.466181	.467744	.468612
2.5	.657298	.660662	.661786	.662651	.663363
3.0	.792636	.794518	.795143	.795656	.796044
4.0	.928489	.928951	.929102	.929222	.929311
5.0	.976022	.976108	.976135	.976156	.976171
7.0	.9973498	.9973442	.9973420	.9973401	.9973386
10.0	.99990311	.99990208	.99990172	.99990141	.99990118

1.3.2 Other Descriptive Characteristics

Let $f_{ij}(t)$ be the probability density function (pdf) of the first passage time T_{ij} from state i to state j in the M/M/c/0 model. Clearly,

$$P_{ij}(t) = f_{ij}(t) * P_{jj}(t) \tag{3.100}$$

for all i and j, where $*$ denotes convolution. Hence, if

$$\hat{f}_{ij}(s) \equiv \int_0^\infty e^{-st} f_{ij}(t)dt \ , \tag{3.101}$$

then

$$\hat{f}_{ij}(s) = \hat{P}_{ij}(s)/\hat{P}_{jj}(s) \ . \tag{3.102}$$

Since

$$\hat{F}^c_{ij}(s) = \frac{1 - \hat{f}_{ij}(s)}{s} \tag{3.103}$$

where

$$\hat{F}^c_{ij}(s) \equiv \int_0^\infty e^{-st} F^c_{ij}(t) dt \tag{3.104}$$

and $F^c_{ij}(t)$ is the ccdf of T_{ij}, we can calculate $F^c_{ij}(t)$ by numerical inversion too. In particular, given the algorithm for calculating $\hat{P}_{ic}(s)$ in Theorem 9, we can calculate $\hat{F}^c_{ic}(s)$ and $F^c_{ic}(t)$.

It is also possible to derive a recursion for the transform $\hat{f}_{i,i+1}(s)$ directly. Considering the possible times and locations of the first transition, we have $\hat{f}_{01}(s) = a/(a+s)$ and

$$\hat{f}_{i,i+1}(s) = \left(\frac{a+i}{a+i+s}\right) \left(\frac{a}{a+i} + \left(\frac{i}{a+i}\right) \hat{f}_{i-1,i}(s) \hat{f}_{i,i+1}(s)\right), \quad i \geq 1 \ , \tag{3.105}$$

From (3.105), we obtain for $i \geq 1$

$$\hat{f}_{i,i+1}(s) = \frac{a}{a+i+s-i\hat{f}_{i-1,i}(s)} \ . \tag{3.106}$$

On the other hand, we can derive (3.106) from (3.93) because

$$\hat{f}_{i,i+1}(s) = \frac{\hat{f}_{i,c}(s)}{\hat{f}_{i+1,c}(s)} = \frac{\hat{P}_{ic}(s)}{\hat{P}_{i+1,c}(s)} = \frac{d_i(s,a)}{d_{i+1}(s,a)} \tag{3.107}$$

and

$$\hat{f}_{0,i}(s) = 1/d_i(s,a) \ . \tag{3.108}$$

For example, the first relation in (3.107) holds because the first passage time from i to c is necessarily the sum of the independent first passage times from i to $i+1$ and from $i+1$ to c. The recursion (3.106) also follows from (3.95) and (3.107).

By the scaling for large a, the distribution of T_{ic} should be approximately independent of a when $c(a) = \lfloor a + \gamma_1\sqrt{a}\rfloor$ and $i(a) = \lfloor a + \gamma_2\sqrt{a}\rfloor$. Indeed, as $a \to \infty$ with $c(a) - a \sim \gamma_1\sqrt{a}$ and $i(a) - a \sim \gamma_2\sqrt{a}$, $T^{(a)}_{i(a)c(a)}$ converges in distribution to the first passage time τ_{γ_2,γ_1} of the Ornstein-Uhlenbeck (OU) diffusion process from γ_2 to γ_1; see Darling and Siegert [32] and Keilson and Ross [54].

We now give a numerical example. We compute the cdf $F_{ac}(t)$ for several values of t in the five cases given in Table 3.1. We let the initial state here be a instead of i; i.e., $\gamma_1 = 0$ instead of $\gamma_1 = -3$. The results are shown in Table 3.3.

We now turn to the time-dependent mean in (3.83). It has Laplace transform

$$\hat{M}_i(s) \equiv \int_0^\infty e^{-st} M_i(t) dt = \frac{i}{1+s} + \frac{a}{1+s}\left(\frac{1}{s} - \hat{P}_{ic}(s)\right) \ ; \tag{3.109}$$

see p. 215 of Beneš [15]. Clearly $\hat{M}_i(s)$ is easily computed once we have $\hat{P}_{ic}(s)$.

Table 3.3 Values of the first-passage-time cdf $F_{ac(a)}(t)$ in the five cases given in Table 3.1 with $\gamma_1 = 0$ and $\gamma_2 = 2$.

time	$I(a = 100)$	$II(a = 400)$	$III(a = 900)$	$IV(a = 2,500)$	$V(a = 10,000)$
2	.1755	.1694	.1674	.1657	.1644
4	.3318	.3230	.3199	.3175	.3156
6	.4564	.4461	.4426	.4397	.4375
8	.5576	.5467	.5429	.5398	.5375
10	.6400	.6291	.6252	.6221	.6197
20	.8715	.8638	.8611	.8588	.8571
30	.9541	.9500	.9485	.9473	.9463
40	.9836	.9817	.9809	.9803	.9798
80	.9997	.9997	.9996	.9996	.9996

Since $(N(t)-a)/\sqrt{a}$ converges to the ROU process as $a \to \infty$ with $i(a)-a \sim \gamma_1\sqrt{a}$ and $c(a) - a \sim \gamma_2\sqrt{a}$, we should have

$$m_{i(a)}^{(a)}(t) \equiv \frac{M_i^{(a)}(t) - a}{\sqrt{a}} \to m_i(t) \quad \text{as} \quad a \to \infty , \tag{3.110}$$

where $m_i(t)$ is the corresponding ROU mean function, provided that $i(a)$ and $c(a)$ are defined as above. We confirm the effectiveness of this scaling by computing the scaled mean $m_{i(a)}^{(a)}(t)$ in (3.110) for several different values of a. In particular, values of $-m_{i(a)}^{(a)}(t)$ are displayed in Table 3.4 for the same five cases as in Tables 3.1 and 3.2. Now we let $\gamma_1 = -3$ again, as in Tables 3.1 and 3.2.

We conclude this section by considering the covariance function in (3.84). We give two expressions for its Laplace transform derived in [11].

Theorem 10 *The covariance function $R(t)$ has Laplace transform*

$$\begin{aligned} \hat{R}(s) &\equiv \int_0^\infty e^{-st}R(t)dt \\ &= \frac{V}{1+s} - \frac{(M-V)}{(1+s)^2} + \frac{(aB)^2}{(1+s)^2}\left(\frac{\hat{P}_{cc}(s)}{B} - \frac{1}{s}\right) \qquad (3.111) \\ &= \frac{V}{1+s} - \frac{(a-M)(\hat{M}_c(s) - (M/s))}{1+s} , \qquad (3.112) \end{aligned}$$

where $B \equiv B(c,a) \equiv \pi_c$ in (3.85), $M \equiv M_i(\infty) = a(1-B)$ and $V \equiv R(0)$ is given in (3.99).

We can apply (3.112) to obtain a useful direct expression for the covariance function.

Table 3.4 Values of the normalized mean $[a - M^{(a)}_{i(a)}(t)]/\sqrt{a}$ in the five cases given in Table 3.1 with $\gamma_1 = -3$.

time	$I(a = 100)$	$II(a = 400)$	$III(a = 900)$	$IV(a = 2,500)$	$V(a = 10,000)$
0.1	2.714512	2.714512	2.714512	2.714512	2.714512
0.5	1.819592	1.819592	1.819592	1.819592	1.819592
1.0	1.103903	1.103920	1.103925	1.103930	1.103638
1.5	.672385	.672445	.672466	.672483	.669390
2.0	.415669	.415718	.415733	.415743	.415751
3.0	.177146	.176943	.176865	.176800	.176748
5.0	.070190	.069547	.069316	.069124	.068976
7.0	.058365	.057607	.057335	.057111	.056938
10.0	.056954	.056174	.055895	.055664	.055486

Corollary. *The covariance can be expressed as*

$$R(t) = Ve^{-t} - (a - M)\int_0^t e^{-(t-u)}[M_c(u) - M)]du \leq Ve^{-t} \ . \tag{3.113}$$

The Corollary to Theorem 10 yields a bound which is approached as $c \to \infty$; i.e., it is known that $R(t) = Ve^{-t}$ in the M/M/∞ model. Beneš proposes a simple approximation

$$R(t) \approx Ve^{-Mt/V} \ , \quad t \geq 0 \ ,$$

which is easy to compute and reasonably accurate; see p. 188 of [15].

Since

$$Cov\left(\frac{N_s(u) - a}{\sqrt{a}} \ , \ \frac{N_s(u + t) - a}{\sqrt{a}}\right) = \frac{Cov(N_s(u), N_s(u + t))}{a}$$

we conclude that $C^{(a)}(t)/a$ should be approximately independent of a provided that $c(a) = a + \gamma\sqrt{a}$. We confirm this scaling in our numerical example below. In particular, values of the normalized covariance function $R(t)/a$ are displayed in Table 3.5. We use the same five cases (values of a) and same nine time points as in Table 3.4. From the evident convergence, it is clear that the values can be used to approximate the covariance function of the limiting ROU diffusion process as well.

1.4 STEADY-STATE WAITING TIMES IN THE GI/G/1 QUEUE

This section contains a second nontrivial example illustrating how numerical inversion of Laplace transforms can be applied. In this section we consider

Table 3.5 Values of the normalized covariance function $R(t)/a$ for the five cases in Table 3.1.

time	$I(a = 100)$	$II(a = 400)$	$III(a = 900)$	$IV(a = 2,500)$	$V(a = 10,000)$
0.1	.784019	.788345	.789814	.791000	.791895
0.5	.502346	.505750	.506913	.507853	.508564
1.0	.288786	.291173	.291990	.292652	.293153
1.5	.166203	.167816	.168370	.168819	.169159
2.0	.095700	.096765	.097132	.097429	.097655
3.0	.031748	.032192	.032345	.032469	.032564
5.0	.003496	.003219	.003589	.003608	.003623
7.0	.0003850	.0003948	.0003982	.0004010	.0004032
10.0	.00001407	.00001455	.00001472	.00001486	.00001496

the steady-state waiting-time distribution in the GI/G/1 queue, drawing upon Abate, Choudhury and Whitt [2], [3], [4].

There is a single server with unlimited waiting space and the first-in first-out service discipline. The interarrival times and service times come from independent sequences of i.i.d. random variables. Let U and V be generic interarrival and service times with cdf's F and G, respectively. Let $\hat{f}$ and $\hat{g}$ be their Laplace Stieltjes transforms, e.g.,

$$\hat{f}(s) = \int_0^\infty e^{-st} dF(t) \; . \tag{4.114}$$

We assume that $EV < EU < \infty$, so that the system is stable; i.e., the steady-state waiting time, denoted by W, is well defined. (The steady-state limit exists and is proper; see [12] for details.) Without loss of generality, we assume that $EV = 1$ and $EU = 1/\rho$. Then $\rho < 1$ becomes the assumed stability condition.

We calculate the complementary cdf (ccdf) $P(W > x)$ by numerically inverting its Laplace transform

$$\hat{W}^c(s) \equiv \int_0^\infty e^{-st} P(W > t) dt = \frac{1 - \hat{w}(s)}{s} \; , \tag{4.115}$$

where

$$\hat{w}(s) \equiv Ee^{-sW} = \int_0^\infty e^{-st} dP(W \le t) \; . \tag{4.116}$$

As in Section 1.3, the main challenge is computing the Laplace transform values $\hat{w}(s)$ for appropriate complex numbers s. The easiest special case is M/G/1 (when the interarrival time has an exponential distribution), in which the waiting-time Laplace transform $\hat{w}$ is available in closed form; see (4.117) below. In Section 1.4.1 we discuss a slightly more difficult case, in which the interarrival-time transform is rational. Then the waiting-time transform $\hat{w}$ is available once some roots of an equation have been found. Such roots can typically be found without difficulty; for further discussion see Chaudhry, Agarwal and Templeton [18] and Chapter ?. The case in which the polynomial in the denominator has degree 2 is especially convenient; we discuss that case in detail.

In Section 1.4.2 we consider the general case. We apply Pollaczek's [68] contour integral representation of the Laplace transform $\hat{w}$. In that case we must perform a numerical integration in order to calculate the transform values. This numerical integration approach applies directly when the service-time moment generating function is finite in a neighborhood of the origin. We show how to compute the required transform values more generally by this approach using exponential damping in Section 4.4.

We give numerical examples in Sections 1.4.3 and 1.4.4. In Section 4.3 we consider gamma distribution examples, i.e., $\Gamma_\alpha/\Gamma_\beta/1$ queues where Γ_α denotes the gamma distribution with shape parameter α. To include a case that is difficult for some algorithms (but not inversion), we consider the $E_k/E_k/1$ model (with Erlang distributions) with very high order k, namely, up to $k = 10^4$.

We conclude in Section 1.4.4 by considering long-tail service-time distributions. We show how exponential damping can be used together with numerical integration to get the Laplace transform values $\hat{w}(s)$ for general interarrival-time and service-time distributions (provided that Laplace transforms of these basic distributions are known). We also show that asymptotic results nicely complement the inversion algorithm by providing accurate values at very large arguments where the inversion gets difficult.

In this section we only consider single-server queues with renewal arrival processes. An inversion algorithm for single-server queues with a non-renewal arrival process (a batch Markovian arrival process) is described in Choudhury, Lucantoni and Whitt [25].

1.4.1 Rational Interarrival-Time Transform

The GI/G/1 model simplifies when one of the transforms $\hat{f}$ or $\hat{g}$ is rational, e.g., if $\hat{f} = \hat{\alpha}/\hat{\beta}$ where $\hat{\alpha}$ and $\hat{\beta}$ are polynomials. As shown by Smith [76], if the service-time transform $\hat{g}$ is rational, then the waiting-time transform $\hat{w}(s)$ itself is rational, and it is possible to obtain an explicit expression for the waiting-time ccdf $P(W > x)$; see p. 324 of Cohen [30]. Hence, numerical inversion is especially attractive when the service-time transform is *not* rational, but it can be used for all service-time distribution.

The most familiar special case is the M/G/1 model, i.e., when $\hat{f}(s) = \rho/(\rho + s)$. Then the Laplace transform $\hat{w}$ is given by the Pollaczek-Khintchine formula

$$\hat{w}(s) = \frac{1-\rho}{1-\rho\hat{g}_e(s)} \;, \tag{4.117}$$

where

$$\hat{g}_e(s) = \int_0^\infty e^{-st} dG_e(t) \tag{4.118}$$

and G_e is the service-time stationary-excess cdf, defined by

$$G_e(t) = \frac{1}{EV} \int_0^t G^c(u)du \ , \quad t \geq 0 \ . \tag{4.119}$$

Since $EV = 1$,

$$\hat{g}_e(s) = (1 - \hat{g}(s))/s \ . \tag{4.120}$$

If we can compute the transform values $\hat{g}(s)$, then the waiting-time ccdf $P(W > x)$ can be computed easily by inverting the transform $(1-\hat{w}(s))/s$ for $\hat{w}$ in 4.117 and $\hat{g}_e$ in (4.120). Numerical examples are given in [7].

More generally, we can calculate the waiting-time ccdf $P(W > x)$ whenever the interarrival-time transform $\hat{f}$ is rational. Henceforth in this subsection we assume that $\hat{f} = \hat{\alpha}/\hat{\beta}$, where $\hat{\beta}$ is a polynomial of degree m and $\hat{\alpha}$ is a polynomial of degree at most $m-1$. The model is then denoted $K_m/G/1$. In order to compute the transform, we must solve for the zeros of the equation

$$\hat{f}(s)\hat{g}(-s) = 1 \ . \tag{4.121}$$

The following theorem comes from p. 329 of Cohen [30].

Theorem 11 *Consider the $K_m/G/1$ queue with $\rho < 1$ in which the interarrival-time transform is rational, i.e., $\hat{f}(s) = \hat{\alpha}(s)/\hat{\beta}(s)$, where $\hat{\beta}(s)$ has degree m and $\hat{\alpha}(s)$ has degree at most $m-1$. Let the coefficient of s^m in $\hat{\beta}(s)$ be 1. Then equation (4.121) has m zeros with $Re(s) \leq 0$, exactly one of which is 0. Let $-\delta_i, 1 \leq i \leq m-1$, be the $m-1$ zeros with $Re(s) < 0$. Then the steady-state waiting time has Laplace transform.*

$$\hat{w}(s) = \frac{-\hat{\beta}(0)cs(1-\rho)}{\hat{\beta}(-s) - \hat{g}(s)\hat{\alpha}(-s)} \prod_{i=1}^{m-1} \frac{\delta_i - s}{\delta_i} \ , \tag{4.122}$$

where

$$c = \frac{\beta'(0) - \alpha'(0)}{\beta(0)} \ .$$

The mean waiting time and the probability of emptiness are

$$EW = \frac{\rho}{2(1-\rho)} \left\{ EV^2 + EU^2 + 2EV\frac{\alpha'(0)}{\alpha(0)} - 2EU\frac{\beta'(0)}{\beta(0)} \right\} + \sum_{i=1}^{m-1} \delta_i^{-1}$$

and

$$P(W = 0) = (1-\rho)E[U]\beta(0) \prod_{i=1}^{m-1} \delta_i^{-1} \ .$$

The idle time within a busy cycle has Laplace transform

$$\hat{i}(s) = 1 - \frac{s}{\beta(s)} \prod_{i=1}^{m-1} (\delta_i + s) \ .$$

When all the zeros with $Re(s) < 0$ of (4.121) can be found, which can usually be done without difficulty numerically, and which is easy in the case of $m = 2$, the waiting-time cdf can easily be calculated by numerically inverting the transform $(1 - \hat{w}(s))/s$ for $\hat{w}$ in (4.122).

We now give more explicit formulas for the case $m = 2$. In the case $m = 2$, equation (4.121) has precisely three roots: $\eta, 0$ and $-\delta$, where $\eta > 0$ and $\delta > 0$. Since the roots are all real, it is elementary to find them.

Let the interarrival time have transform

$$\hat{f}(s) = \frac{1 + (c_1 + c_2 - \rho^{-1})s}{(1 + c_1 s)(1 + c_2 s)} \tag{4.123}$$

for c_1 and c_2 real and positive with $c_1 \le c_2$, so that the mean and squared coefficient of variation (SCV) are

$$f_1 = \rho^{-1} \ \text{ and } \ c_a^2 = 2(\rho c_1 + \rho c_2 - \rho^2 c_1 c_2) - 1 \ . \tag{4.124}$$

Expanding (4.123) into partial fractions yields for $c_1 \neq c_2$:

$$\hat{f}(s) = \left(\frac{c_2 - \rho^{-1}}{c_2 - c_1}\right)(1 + c_1 s)^{-1} + \left(\frac{\rho^{-1} - c_1}{c_2 - c_1}\right)(1 + c_2 s)^{-1} \ . \tag{4.125}$$

We see that the pdf is hyperexponential (H_2) with $c_a^2 > 1$ if $c_2 > \rho^{-1} > c_1$; i.e.,

$$f(t) = p\lambda_1 e^{-\lambda_1 t} + (1 - p)\lambda_2 e^{-\lambda_2 t} \ , \quad t \ge 0 \ , \tag{4.126}$$

for $\lambda_i = 1/c_i$ and $p = (c_2 - \rho^{-1})/(c_2 - c_1)$ with $0 < \rho < 1$. On the other hand, if $\rho^{-1} > c_2 > c_1$, then (4.126) still holds but with $p < 0$. Then the pdf $f(t)$ in (4.126) is a difference of two exponentials and is called hypoexponential. Then the SCV satisfies $1/2 < c_a^2 < 1$.

For the special case of the hypoexponential with $c_1 + c_2 - \rho^{-1} = 0$, we can express the transform $\hat{f}(s)$ in (4.123) as

$$\hat{f}(s) = (1 + c_1 s)^{-1}(1 + c_2 s)^{-1} \ , \tag{4.127}$$

so that the pdf $f(t)$ is the convolution of two exponential pdf's with means c_1 and c_2, which we refer refer to as generalized Erlang of order 2 (GE_2). When $c_1 = c_2$, we must have $c_1 = c_2 = 1/2\rho$ and the distribution becomes Erlang (E_2), i.e.,

$$\hat{f}(s) = (1 + [s/2\rho])^{-2} \ . \tag{4.128}$$

The degenerate exponential case is approached as $c_1 \to 0$ with $c_2 \to \rho^{-1}$.

The following is a direct consequence of Theorem 11.

Corollary. *Consider the $K_2/G/1$ queue having service-time transform $\hat{g}(s)$ with mean 1 and interarrival-time transform $\hat{f}(s)$ in (4.123) with mean ρ^{-1} for $0 < \rho < 1$. Then the steady-state waiting time has Laplace transform*

$$\hat{w}(s) = \frac{(1-\rho)(1-(s/\delta))}{(1-\rho\hat{g}_e(s)) + (\rho c_1 + \rho c_2 - 1)(1-\hat{g}(s)) - \rho c_1 c_2 s} \ . \tag{4.129}$$

The mean waiting time and the emptiness probability are

$$EW = \frac{\rho(\rho^{-2}c_a^2 + c_s^2)}{2(1-\rho)} + \frac{1-\rho}{2\rho} + \frac{1}{\delta} - c_1 - c_2 \tag{4.130}$$

and

$$P(W=0) = \lim_{s\to\infty} \hat{w}(s) = \frac{1-\rho}{\rho c_1 c_2 \delta} \ . \tag{4.131}$$

The idle time transform is

$$\hat{\imath}(s) = \frac{1 + (c_1 + c_2 - \delta c_1 c_2)s}{(1+c_1 s)(1+c_2 s)} \ . \tag{4.132}$$

The first two idle-time moments are

$$i_1 = \delta c_1 c_2 \tag{4.133}$$

and

$$\frac{i_2}{2i_1} = c_1 + c_2 - \delta^{-1} \ . \tag{4.134}$$

Remark 1.4.1 It is of interest to compare the idle-time transform $\hat{\imath}(s)$ in (4.132) to $\hat{f}_e(s)$, the LST of the stationary-excess cdf F_e of the interarrival-time cdf F, defined as in (4.119). From (4.125), we see that

$$\hat{f}_e(s) = \frac{1 + \rho c_1 c_2 s}{(1+c_1 s)(1+c_2 s)} \ . \tag{4.135}$$

For any cdf H, let h_k be its k^{th} moment. Then the first moment of the cdf F_e is

$$f_{e1} = \frac{f_2}{2f_1} = \frac{c_a^2 + 1}{2\rho} = c_1 + c_2 - \rho c_1 c_2 \tag{4.136}$$

If we approximate i_1 by f_{e1}, we obtain from (4.133) and (4.136) an approximation for the root δ, namely,

$$\delta \approx \frac{1 + c_a^2}{2\rho c_1 c_2} \approx \frac{c_1 + c_2 - \rho c_1 c_2}{c_1 c_2} \ , \tag{4.137}$$

which can be used as an initial guess when applying the Newton-Raphson root finding procedure. As indicated in [4], a good initial guess for η is EW/ρ.

Table 4.6 The root δ and other characteristics as functions of ρ in the $K_2/G/1$ model in Example 1.4.1.

ρ	δ	$P(W=0)$	EW	$\frac{\rho(c_a^2+c_s^2)}{2(1-\rho)}$	i_1	f_{e1}
.1	.105	.857	.25	.22	10.50	15.00
.2	.220	.729	.55	.50	5.49	7.50
.4	.475	.506	1.44	1.33	2.97	3.75
.5	.613	.408	2.13	2.00	2.45	3.00
.6	.757	.317	3.16	3.00	2.10	2.60
.8	1.058	.151	8.20	8.00	1.65	1.88
.9	1.214	.074	18.21	18.00	1.50	1.67
.99	1.357	.007	198.23	198.00	1.38	1.52

Example 1.4.1 (An $H_2/G/1$ Example) We conclude this subsection by giving a numerical example. Let the interarrival-time transform be as in (4.123) with $c_1 = 1/2\rho$ and $c_2 = 2/\rho$. Then the pdf is H_2, in particular,

$$f(t) = \frac{2}{3c_1}e^{-t/c_1} + \frac{1}{3c_2}e^{-t/c_2} , \quad t \geq 0 , \tag{4.138}$$

so that $f_1 = 1/\rho, f_2 = 3/\rho^2$ and $f_3 = 33/2\rho^3$. Let the service-time pdf be gamma with mean 1 and shape parameter 1/2 ($\Gamma_{1/2}$), i.e.,

$$g(t) = (2\pi t)^{-1/2}e^{-t/2} , \quad t \geq 0 , \tag{4.139}$$

with $\hat{g}(s) = (1+2s)^{-1/2}$.

In Table 4.1 we display the root δ as a function of ρ. We also display several related quantities computable directly from δ, in particular, EW, $P(W=0)$ and i_1. We compare EW to the heavy-traffic approximation $\rho(c_a^2+c_s^2)/2(1-\rho)$ and we compare i_1 to f_{e1}. In this case the heavy-traffic approximation for the mean EW is quite good for all ρ. The mean stationary excess of an interarrival time f_{e1} consistently exceeds the idle-time mean i_1.

To do further analysis, we consider the case $\rho = 0.75$. Then we find that $\delta = 0.98115392$ and $P(W=0) = 0.19110151$. (The high precision in $P(W=0)$ would rarely be needed. On the other hand, the high precision in δ may be needed because it appears in the transform $\hat{w}(s)$ in (4.129) that we intend to invert. When the tail probability $P(W > x)$ is very small, we need very small absolute error to achieve reasonable relative error.) We compute the exact values of the ccdf $P(W > x)$ for several values of x in Table 4.2. We compare it to the Cramer-Lundberg asymptotic approximation

$$P(W > x) \sim \alpha e^{-\eta x} \quad \text{as } x \to \infty , \tag{4.140}$$

Table 4.7 A comparison of the Cramér-Lundberg approximation with exact values obtained by numerical inversion for the waiting-time ccdf $P(W > x)$ for the queue $H_2/\Gamma_{1/2}/1$ with traffic intensity $\rho = 0.75$ in Example 1.4.1.

x	numerical transform inversion	Cramér-Lundberg approximation
10^{-8}	0.808898	0.785
.5	0.747832	0.736
1.0	0.697692	0.691
2.0	0.611125	0.608
4.0	0.472237	0.4716
8.0	0.283501	0.28346
16.0	0.1023909	0.1023905
30.0	0.01723289	same
50.0	0.00135140	same
70.0	0.00010598	same
80.0	0.00002968	same

where η is the positive real root of (4.121) with minimum real part and $f(x) \sim g(x)$ as $x \to \infty$ means that $f(x)/g(x) \to 1$ as $x \to \infty$; see p. 269 of Asmussen [12] and Abate, Choudhury and Whitt [4].

For the Cramér-Lundberg approximation with $\rho = 0.75$, we obtain

$$\eta^{-1} = 7.85645477 \text{ and } \alpha = 0.78472698 \ .$$

In this example, $EW = 6.185875$, so that the two rough estimates of η^{-1} are $EW/\rho = 8.2478$ and $(c_a^2 + c_s^2)/2(1-\rho) = 8.0$. Both are reasonable approximations that work well as initial guesses in the Newton-Raphson procedure.

From Table 4.2 we see that the Cramér-Lundberg approximation is excellent, even when x is not large. This numerical example illustrates a general phenomenon: When the Cramér-Lundberg approximation applies, it often serves as well as the exact values in applications. However, one should be careful about generalizing; asymptotic approximations do not always perform this well; see Section 1.4.4 below.

Finally, we remark that numerical inversion can also be used to calculate asymptotic parameters such as α and η in (4.140); e.g., see [23] and [1].

1.4.2 The Pollaczek Contour Integral

Pollaczek [68] derived a contour-integral expression for the Laplace transform of the steady-state waiting-time distribution in the general GI/G/1 queue. Let

H be the cumulative distribution (cdf) of $V - U$ and let ϕ be its moment generating function, defined by

$$\phi(z) = Ee^{z(V-U)} \equiv \int_{-\infty}^{\infty} e^{zt} dH(t) = \hat{f}(z)\hat{g}(-z) , \qquad (4.141)$$

which we assume is analytic for complex z in the strip $|Re\ z| < \delta$ for some $\delta > 0$. A natural sufficient condition for this analyticity condition is for the service-time and interarrival-time distributions to have finite moment generating functions in a neighborhood of the origin, and thus moments of all orders, but *neither the transform of the interarrival-time distribution nor the transform of the service-time distributions need be rational.*

Moreover, as noted on p. 40 of Pollaczek [69] and in Section II.5.9 on p. 31 of Cohen [30], it is possible to treat the case of more general service-time distributions by considering limits of service-time distributions that satisfy this analyticity condition. We discuss this extension here in Section 1.4.4. Now we assume that $\phi(z)$ in (4.141) is indeed analytic for complex z in the strip $|Re\ z| < \delta$ for some $\delta > 0$.

Here is Pollaczek's contour integral representation; see Chapter 5 of Cohen [30].

Theorem 12 *In the GI/GI/1 model, the waiting-time Laplace transform is*

$$\hat{w}(s) \equiv Ee^{-sW} = \exp\left\{-\frac{1}{2\pi i}\int_C \frac{s}{z(s-z)}\log[1-\phi(-z)]dz\right\} , \qquad (4.142)$$

where s is a complex number with $Re(s) \geq 0$, C is a contour to the left of, and parallel to, the imaginary axis, and to the right of any singularities of $\log[1-\phi(-z)]$ *in the left half plane, for ϕ in (4.141).*

We have described algorithms for computing tail probabilities $P(W > x)$ by numerically inverting the Laplace transform $\hat{W}^c$ in (4.115). For example, the algorithm in Section 1.2.2 reduces to a finite weighted sum of terms $Re(\hat{W}^c(u + kvi))$ over integers k for appropriate real numbers u and v (the number of different k might be as low as 30.) To apply this algorithm, it suffices to compute Re $\hat{W}^c(s)$ for s of the required form $s = u + kvi$. For this purpose, it suffices to compute $\hat{w}(s)$ in (4.142) for s of this same form.

The standard expression for (4.142) has the contour just to the left of the imaginary axis, but this poses numerical difficulties because of the singularity in the first portion of the integrand, $s/z(s-z)$, and in the second portion, $\log[1-\phi(-z)]$, at $z = 0$. However, this difficult is easily avoided by moving the vertical contour of integration to the left, but still keeping it to the right of the singularity of $\log[1-\phi(-z)]$ in the left halfplane closest to the origin, which we denote by $-\eta$. It turns out that this critical singularity of $\log[1-\phi(-z)]$ also corresponds to the singularity of Ee^{-sW} in the left halfplane closest to the origin; i.e., the dominant singularity of (4.121) or

$$\eta = \sup\{s > 0 : Ee^{sW} < \infty\} . \qquad (4.143)$$

Moreover, η is the asymptotic decay rate in the Cramér-Lundberg approximation in (4.140)

Given a reasonable estimate of η, we perform the integration (4.142) by putting the contour at $-\eta/2$. On this contour, $z = -\eta/2 + iy$ and y ranges from $-\infty$ to $+\infty$. Equation (4.142) becomes $Ee^{-sW} = \exp(-I)$, where

$$
\begin{aligned}
I &= \frac{1}{2\pi}\left(\int_{-\infty}^{0} \frac{s}{z(s-z)} \log[1-\phi(z)]dy \int_{0}^{\infty} \frac{s}{z(s-z)} \log[1-\phi(-z)]dy\right) \\
&= \frac{1}{2\pi}\int_{0}^{\infty} \left(\frac{s}{\overline{z}(s-\overline{z})} \log[1-\phi(-\overline{z})] \frac{s}{z(s-z)} \log[1-\phi(-z)]\right) \, dy \qquad (4.144)
\end{aligned}
$$

with $\overline{z} = -\eta/2 - iy$. In general, I in (4.143) is complex; we compute its real and imaginary parts by integrating the real and imaginary parts of the integrand, respectively. However, if s is real, then so is I. In that case, the real parts of the two components of the integrand are the same, thereby simplifying the computation somewhat.

For the GI/G/1 queue, the desired parameter η in (4.143) can usually be easily found (by search algorithm) by solving the transform equation (4.121) for the positive real root with minimum real part. In order to find η, it suffices to restrict attention to the interval $(0, \eta_s)$, where

$$\eta_s = \sup\{s \geq 0 : Ee^{sV} < \infty\} \qquad (4.145)$$

with V being a service time. (Of course η_s can be infinite, but that presents no major difficulty; in that case we start the search in the interval $(0, 1)$. If the interval does not contain a root of (4.121), then we geometrically increase the upper limit until it contains the root.)

However, it can happen that transform equation (4.121) does not have a root even though the transform ϕ in (4.141) satisfies the analyticity condition; This means that $\eta = \eta_s > 0$ for η in (4.143) and η_s in (4.145), so that we can still put the vertical contour at $-\eta/2$.

A specific numerical integration procedure that can be used is fifth-order Romberg integration, as described in Section 4.3 of Press, Flannery, Teukolsky and Vetterling [71]. First divide the integration interval $(0, \infty)$ in (4.143) into a number of subintervals. If η is not too close to 0, then no special care is needed and it suffices to use the two subintervals $(0, 1)$, and $(1, \infty)$ and then transform the infinite interval into $(0, 1)$ using the transformation in (4.4.2) of [71].

However, more care is required for less well behaved distributions (e.g., highly variable, nearly deterministic, or when η is close to 0). Then we examine the integrand more carefully and choose subintervals so that the ratio of the maximum to the minimum value within any subinterval is at most 10 or 100. This helps ensure that computational effort is expended where it is needed. Indeed, a version of the algorithm was developed to do this automatically. In this automatic procedure, the integration interval $(0, \infty)$ in (4.143) is divided into $m + 1$ subintervals: $(0, b_1), (b_1, b_2), \ldots, (b_{m-1}, b_m), (b_m, \infty)$. The last infinite subinterval (b_m, ∞) is transformed into the finite interval $(0, b_m^{-1})$ using the transformation in (4.4.2) of [71]. Within each subinterval, a fifth-order Romberg integration procedure is performed. An error tolerance of 10^{-12} is specified and the program generates successive partitions (going from n to $2n$ points) until the estimated improvement is no more than either the tolerance value itself or the product of the tolerance and the accumulated value of the integral so far (in the current subinterval as well as in earlier subintervals).

A specific procedure used for choosing the subintervals is as follows. If the integrand doesn't differ by more than a factor of 10 in the interval $(0,1)$ then b_1 is chosen as 1. Otherwise, b_1 is chosen such that the integrand roughly changes by a factor of 10 in the interval $(0,b_1)$. The endpoint b_1 is roughly determined by evaluating the integrand at 0 and at the points 10^{-n} with $n = 10, 9, \ldots, 0$. For $2 \le i \le m$, the ratio b_i/b_{i-1} is assumed to be a constant K, where K is an input parameter. The number m is determined by looking at the ratio of the contribution from the subinterval (b_{i-1}, b_i) to the total contribution so far. If this ratio is less than a constant ϵ, where ϵ is a second input parameter, then m is set to i, i.e., the next interval is made the last interval. A good choice of K and ϵ depends on the service-time and interarrival-time distributions. Typically less well behaved distributions require smaller K and/or ϵ. Our numerical experience indicates that $K = 3$ and $\epsilon = 10^{-4}$ works pretty well for most cases of interest.

The Laplace transform inversion algorithm also gives an estimate of the final error. If it is close to or below the 10^{-8} precision specified, we can be fairly confidence of a good computation.

1.4.3 Gamma Distribution Examples

In this subsection and the next we illustrate the numerical inversion algorithms for the GI/G/1 queue. In this subsection we consider $\Gamma_\alpha/\Gamma_\beta/1$ queues, where Γ denotes the gamma distribution, and α and β are the shape parameters of the interarrival-time and service-time distributions, respectively. The gamma distribution with scale parameter λ and shape parameter α has density

$$f(x) = \frac{1}{\Gamma(\alpha)} \lambda^\alpha x^{\alpha-1} e^{-\lambda x}, \quad x > 0 \, , \tag{4.146}$$

mean α/λ, variance α/λ^2 and Laplace transform

$$Ee^{-sV} \equiv \int_0^\infty e^{-sx} f(x) dx = \left(\frac{\lambda}{\lambda + s} \right)^\alpha \, . \tag{4.147}$$

The transform in (4.147) is rational if, and only if, the shape parameter α is a positive integer. When $\alpha = k$ for an integer k, the gamma distribution is also called Erlang of order k (E_k). Since convolutions of exponential distributions are smooth, we expect that this distribution will not be very difficult, at least when α is not too small; see Section 12 of Abate and Whitt [7].

We stipulate that the mean service time is 1 and that the arrival rate is ρ. The remaining two parameters α and β of the $\Gamma_\alpha/\Gamma_\beta/1$ queue are the shape parameters of the interarrival-time and service-time distribution. Since the squared coefficient of variation (SCV) is the reciprocal of the shape parameter, it suffices to specify the SCVs c_a^2 and c_s^2 of the interarrival-time and service-time distributions.

The algorithm can be checked against known results by considering the $\mathrm{E_k}/\Gamma/1$ and $\Gamma/\mathrm{E_k}/1$ special cases. These are special cases of the PH/G/1 and GI/PH/1 queues, for which there are alternative algorithms exploiting results for the M/G/1 and GI/M/1 paradigms in Neuts [64, 65]. Another alternative for comparison is the root finding algorithm as in Chaudhry, Agarwal and Templeton [18]; e.g., we found good agreement with results for the $\mathrm{E}_{10}/\mathrm{E}_{100}/1$ queue in Table 10 on p. 141 of Chaudhry, Agarwal and Templeton [18].

Some other algorithms for $\mathrm{E}_k/\mathrm{E}_m/1$ queues get more difficult as k and m increase. Hence, we performed calculations for $\mathrm{E}_k/\mathrm{E}_k/1$ models with large k. Other $\Gamma_\alpha/\Gamma_\beta/1$ examples are given in [2].

Example 1.4.2 ($\mathbf{E_k/E_k/1}$ Queues) We did calculations for the $\mathrm{E}_k/\mathrm{E}_k/1$ queue for $k = 10$, $k = 100$, $k = 1000$ and $k = 10,000$. In this case the transform equation in (4.121) for the asymptotic decay rate η becomes

$$\left(\frac{k}{k-\eta}\right)^k \left(\frac{k}{k+\eta/\rho}\right)^k = 1 \ , \tag{4.148}$$

from which we easily obtain

$$\eta = k(1-\rho) \ . \tag{4.149}$$

Since E_k is approaching a deterministic distribution as k increases, to avoid having negligible probabilities we let $\rho \equiv \rho_k$ increase with k. In particular, we let $\rho_k = 1 - k^{-1}$. With this choice, $\eta \equiv \eta_k = 1$ for all k. Also W_k, the steady-state waiting time in model k, converges to an exponential random variable with mean 1 as $k \to \infty$, as can be seen by applying the heavy-traffic argument of Kingman [55] using (4.142).

Numerical values of some tail probabilities and cumulants are given for $\mathrm{E}_k/\mathrm{E}_k/1$ queues for these cases in Table 4.3. (The cumulants are calculated by other Pollaczek contour integrals; see [2]. The exponential limit is displayed as well under the heading $k = \infty$. None of these presented any numerical difficulties.

Interestingly, from Table 4.3, we see that for these cases W is quite well approximated by a mixture of an atom at 0 with probability $1/\sqrt{k} = \sqrt{1-\rho}$ and an exponential with mean 1 with probability $1 - 1/\sqrt{k}$.

1.4.4 Long-Tail Service-Time Distributions

Pollaczek's contour integral representation in (4.142) depends on an analyticity condition that is satisfied when the interarrival-time and service-time distributions have finite moment generating functions in a neighborhood of the

Table 4.8 Tail probabilities and cumulants of the steady-state waiting time in the $E_k/E_k/1$ model with traffic intensity $\rho = 1 - k^{-1}$, as a function of k. The case $k = \infty$ is an exponential with mean 1.

Congestion Measure	k 10	100	1,000	10,000	∞
$P(W > 0)$	0.7102575	0.9035808	0.9687712	0.9900406	1.0000000
$P(W > 1)$	0.2780070	0.3385844	0.3584117	0.3648607	0.3678794
$P(W > 3)$	0.0376169	0.0458224	0.0485057	0.0493785	0.0497871
$P(W > 5)$	0.0050909	0.0062014	0.0065645	0.0066825	0.0067379
$P(W > 7)$	0.0006890	0.0008393	0.0008884	0.0009044	0.0009119
$c_1(W)$	0.7484185	0.9195281	0.9741762	0.9917852	$0! = 1$
$c_2(W)$	0.9491038	0.9951389	0.9995064	0.9999502	$1! = 1$
$c_3(W)$	1.982543	1.995377	1.9999854	1.9999996	$2! = 2$
$c_4(W)$	5.992213	5.999948	5.999999	6.000000	$3! = 6$
$c_5(W)$	23.995966	23.999995	24.000000	24.000000	$4! = 24$
$c_6(W)$	119.997754	120.000000	120.000000	119.999993	$5! = 120$

origin; i.e., when $Ee^{sU} < \infty$ and $Ee^{sV} < \infty$ for some $s > 0$. However, it is possible to treat the general case by representing a general distribution as a limit of a sequence of distributions each of which satisfies this analyticity condition. It is known that the associated sequence of steady-state waiting-time distributions will converge to a proper limit provided that the distributions and their means also converge to proper limits; see p. 194 of Asmussen [12]. (The moment condition is actually on $(V - U)^+ = \max\{V - U, 0\}$.)

In fact, the long-tail interarrival-time distributions actually present no difficulty. It suffices to have $\phi(z)$ in (4.141) analytic in the strip $0 < \text{Re}\,(z) < \delta$ for some $\delta > 0$. However, the service-time distribution poses a real problem.

Hence, if $G^c(x)$ is the given service-time ccdf with Laplace transform $\hat{G}^c(s)$ and mean $m = \hat{G}^c(0)$, then it suffices to find an approximating sequence of service-time complementary cdf's $\{G_n^c(x) : n \geq 1\}$ with associated Laplace transforms $\{\hat{G}_n^c(s) : n \geq 1\}$ and means $\{m_n = \hat{G}_n^c(0) : n \geq 1\}$ such that $\hat{G}_n^c(s) \to \hat{G}^c(s)$ as $n \to \infty$ for all s. Then $G_n^c(x) \to G^c(x)$ as $n \to \infty$ for all x that are continuity points of the limiting complementary cdf G^c and $m_n \to m$ as $n \to \infty$.

A natural way to obtain a sequence of approximating service-time distributions with finite moment generating functions in some neighborhood of the origin when this condition is not satisfied originally is to introduce *exponential damping* in the Laplace-Stieltjes transform with respect to G. In particular, for any $\alpha > 0$ let the α-damped ccdf be

$$G_\alpha^c(x) = \int_x^\infty e^{-\alpha t} dG(t), \quad x \geq 0 \,. \tag{4.150}$$

Since we want a proper probability distribution, we put mass $1 - G_\alpha^c(0)$ at 0. If the original service-time distribution has mean 1 and we want the new service-time distribution also to have mean 1, then we also divide the random variable V_α with cdf G_α by the new mean m_α, i.e., we let the complimentary cdf be $G_\alpha^c(m_\alpha x)$.

The direct approximation in (4.150) makes the service-time distribution stochastically smaller than the original service-time distribution, which in turn makes the new steady-state waiting-time distribution stochastically smaller than the original one, which may be helpful for interpretation. However, keeping the same mean seems to give substantially better numbers. Here we keep the mean fixed at 1.

From (4.150), it is easy to see that, if $\hat{g}$ is the original Laplace-Stieltjes transform of G, then the Laplace-Stieltjes transform of $G_\alpha(m_\alpha x)$ with mean 1 is

$$\hat{g}_\alpha(s) = \hat{g}(\alpha + (s/m_\alpha)) + 1 - \hat{g}(\alpha) \ , \tag{4.151}$$

where

$$m_\alpha = -\hat{g}'_\alpha(0) = -\hat{g}'(\alpha) \ . \tag{4.152}$$

Thus, the Laplace transform $\hat{G}_\alpha^c$ of $G_\alpha^c(m_\alpha x)$ for G_α^c in (4.150) is

$$\hat{G}_\alpha^c(s) = \frac{1 - \hat{g}_\alpha(s)}{s} \tag{4.153}$$

for $\hat{g}_\alpha$ in (4.151). Hence, given $\hat{g}$, we can readily calculate values of $\hat{G}_\alpha^c$ for any $\alpha > 0$.

However, this approach is not trivial to implement, because it often requires a very small α before $G_\alpha(x)$ is a satisfactory approximation for $G(x)$, and a small α means a small η in (4.121). Indeed, $0 < \eta \le \eta_s = \alpha$. In turn, a small η means a relatively difficult computation, because the contour at $-\eta/2$ is near the singularity at 0. However, this can be handled by being careful with the numerical integration. Indeed, the algorithm employing an adaptive choice of integration subintervals was originally developed to handle this case.

Before considering an example illustrating exponential damping, we mention that it is also possible to approximate in other ways; e.g., see Asmussen, Nerman and Ollson [13], Feldmann and Whitt [40] and Gaver and Jacobs [43].

Example 1.4.3 (An M/G/1 Queue) To illustrate how the exponential damping approach works, we consider an M/G/1 queue with service-time density

$$g(x) = x^{-3}(1 - (1 + 2x + 2x^2)e^{-2x}), \quad x \ge 0 \ . \tag{4.154}$$

This distribution has first two moments $m_1 = 1$ and $m_2 = \infty$, so that there is a proper steady-state waiting-time distribution which has infinite mean; see pp. 181-184 of Asmussen [12]. It is easy to see that our service-time distribution has complementary cdf

$$G^c(x) = (2x^2)^{-1}(1 - (1 + 2x)e^{-2x}), \quad x \ge 0, \tag{4.155}$$

Table 4.9 A comparison of approximations for tail probabilities $P(W > x)$ with exact values in the M/G/1 model with $\rho = 0.8$ and the long-tail service-time distribution in Example 4.3.

Cases	x: 4	20	100	500	2500
$\alpha = 0$ (exact)	0.4653	0.1558	0.02473	0.004221	0.000811
$\alpha = 10^{-2}$	0.4539	0.1175	0.0036	0.000002	0.000000
$\alpha = 10^{-3}$	0.4631	0.1474	0.0017	0.00096	0.000006
$\alpha = 10^{-4}$	0.4650	0.1544	0.02315	0.0032	0.00032
$\alpha = 10^{-6}$	0.4653	0.1557	0.02469	0.004193	0.000788
$\alpha = 10^{-8}$ (M/G/1)	0.4653	0.1558	0.02473	0.004221	0.000810
$\alpha = 10^{-8}$ (Pollaczek)	0.4653	0.1558	0.02473	0.004224	0.000815
asymptotics 1 term	0.50	0.10	0.020	0.0040	0.00080
2 terms	1.33	0.165	0.0239	0.00421	0.000810

and Laplace transform

$$\hat{g}(s) = 1 - s + \frac{s^2}{2} \ln(1 + (2/s)) \ . \tag{4.156}$$

This service-time distribution is a *Pareto mixture of exponentials* (PME) distribution introduced in Abate, Choudhury and Whitt [3].

We use the M/G/1 queue to make it easier to compare our numerical results with other known results. In particular, we can also apply inversion directly to the Pollaczek-Khintchine (transform) formula (4.117). We also can compare the inversion results to a two-term asymptotic expansion derived by J. W. Cohen (personal communication). The two-term asymptotic expansion is

$$P(W > x) \sim \frac{\rho}{2(1-\rho)x}\left(1 + \frac{\rho}{(1-\rho)x}\right) \quad \text{as} \quad x \to \infty, \tag{4.157}$$

where $\gamma \equiv 0.5772\ldots$ is Euler's constant. The first term is just $P(W > x) \sim \rho/(1-\rho)x$. The inclusion of Euler's constant in equation (4.157) corrects an error in the second term of a conjecture on the bottom of p. 328 of [3]. The reasoning on p. 329 of [3] can be used, but the asymptotics for one term needs to be corrected. In particular, $\mathcal{L}^{-1}(s \log^2 s) \sim -2(\gamma - 1 + \log x)/x^2$ as $x \to \infty$.

In Table 4.4 we display the tail probabilities $P(W > x)$ for this M/G/1 example with $\rho = 0.8$ for five values of x: $x = 4$, $x = 20$, $x = 100$, $x = 500$ and $x = 2500$. The table shows the exact results (no damping, $\alpha = 0$) obtained from the Pollaczek-Khintchine formula and asymptotic approximations based on (4.157); as well as the approximations obtained from five values of the damping parameter α: $\alpha = 10^{-2}$, $\alpha = 10^{-3}$, $\alpha = 10^{-4}$, $\alpha = 10^{-6}$ and $\alpha = 10^{-8}$. The numerical results based on the algorithm here and the Pollaczek-Khintchine algorithm agreed to the stated precision for all values of α except $\alpha = 10^{-8}$, so only in the single case $\alpha = 10^{-8}$ are both numerical results given. Table 4.4 shows that the exact ($\alpha = 0$) results and the two algorithms with $\alpha = 10^{-8}$ are all quite close, even for $x = 2500$. Table 4.4 shows that the damping parameter α needs to be smaller and smaller as x increases in order for the calculations based on the approximating cdf G_α to be accurate. However, the calculation gets more difficult as x increases and α decreases. For the smaller α values reported, it was important to carefully choose the subintervals for the Romberg integration so that the integrand does not fluctuate too greatly within the subinterval. This was done by the automatic procedure described earlier.

In this example we are able to obtain a good calculation for all x because the asymptotics apply before the computation gets difficult. The relative percent error for the one-term (two-term) approximations at $x = 100$, $x = 500$ and $x = 2,500$ are, respectively, 19%, 5.2% and 1.4% (3.2%, 0.2% and $< 0.1\%$). This example illustrates a general phenomenon: Numerical methods work well together with asymptotics. One approach tends to work well when the other breaks down. They also serve as checks on each other.

1.5 CLOSED QUEUEING NETWORKS

This final section contains an example illustrating how the numerical inversion algorithm for generating functions of sequences of complex numbers in Section 1.2.4 can be applied. In this section we consider the product-form steady-state distribution of a closed (Markovian) queueing network (CQN), drawing upon Choudhury, Leung and Whitt [20] and Choudhury and Whitt [29]. This section also illustrates how inversion of multi-dimensional transforms can be applied; see Choudhury, Lucantoni and Whitt [24] for more on multi-dimensional inversion. Quantities of interest in stochastic loss networks can be calculated in the same way; see Choudhury, Leung and Whitt [21], [22], [19].

It is known that the steady-state distribution of a CQN can be expressed in terms of a normalization constant. We show that steady-state characteristics of interest can be calculated by numerically inverting the multidimensional generating function of the normalization constant (regarded as a function of chain populations).

In Section 1.5.1 we introduce the CQN model and display the generating functions. In Section 1.5.2, we discuss the technique of dimension reduction for reducing the effective dimension of the inversion, which is important since the computational effort grows exponentially with the dimension. In Section 1.5.3 we discuss scaling to control the aliasing error, which is important since the normalization constants are not bounded. Finally, in Section 1.5.4 we illustrate by solving a challenging numerical example.

1.5.1 The Model and the Generating Functions

We start by describing the general class of probability distributions that we consider. We start somewhat abstractly, but below we will consider a special class of closed queueing networks. Let the state variable be a *job vector* $\mathbf{n} = (n_1, \ldots, n_L)$; n_l is the number of jobs of *type* ℓ; n_l might be the number of customers of a particular class at a particular queue. Let there be a specified *population vector* $\mathbf{K} = (K_1, \ldots, K_p)$; K_j is the population of *chain* j, a fixed quantity specified as part of the model data. The *state space* is the set of allowable job vectors, which depends on the population vector $\mathbf{K}$ and is denoted by $S(\mathbf{K})$. In this setting, the probability distributions that we consider have the form

$$p(\mathbf{n}) = g(\mathbf{K})^{-1} f(\mathbf{n}), \tag{5.158}$$

where

$$g(\mathbf{K}) = \sum_{\mathbf{n} \in S(\mathbf{K})} f(\mathbf{n}) \tag{5.159}$$

and f is a (known) nonnegative real-valued function on the L-fold product of the nonnegative integers. (For example, we might have $f(\mathbf{n}) = \prod_{\ell=1}^{L} f_l(n_l)$ with $f_l(n_l) = \rho_l^{n_l}$.) The term $g(\mathbf{K})$ in (5.158) and (5.159) is called the *normalization constant* or the *partition function*. For the closed queueing network models we will consider (and many other models), the state space has the special form

$$S(\mathbf{K}) = \left\{ \mathbf{n} | n_l \geq 0, \quad \sum_{l \in C_j} n_l = K_j, \quad 1 \leq j \leq p \right\} \tag{5.160}$$

for special sets C_j, $1 \leq j \leq p$.

Given a probability distribution as in (5.158), where the function f is relatively tractable, the major complication is determining the normalization constant $g(\mathbf{K})$ for the relevant population vector $\mathbf{K}$. In this setting, the convolution algorithm calculates $g(\mathbf{K})$ by expressing it in terms of values $g(\mathbf{K}')$ where $\mathbf{K}' < \mathbf{K}$ (i.e., $K_l' \leq K_l$ for all ℓ and $K_l' < K_l$ for at least one ℓ, e.g., see Conway and Georganas [31] or Lavenberg [61]. Other existing non-asymptotic algorithms proceed in a similar recursive manner. See Conway and Georganas [31] for a unified view.

In contrast, we calculate $g(\mathbf{K}) \equiv g(K_1, \ldots, K_p)$ by numerically inverting its multi-dimensional generating function

$$\hat{g}(\mathbf{z}) \equiv \sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} g(\mathbf{K}) \prod_{j=1}^{p} z_j^{K_j} \tag{5.161}$$

where $\mathbf{z} \equiv (z_1, \ldots, z_p)$ is a vector of complex variables. To quickly see the potential advantage of this approach, note that we can calculate $g(\mathbf{K})$ for one vector $\mathbf{K}$ without calculating $g(\mathbf{K}')$ for all the $\prod_{j=1}^{p} K_j$ nonnegative vectors $\mathbf{K}'$ less than or equal to $\mathbf{K}$, as is done with the convolution algorithm.

There are two obvious requirements for carrying out this program. First, we need to be able to compute the generating function values in (5.161) and, second, we need to be able to perform the numerical inversion. The first requirement often turns out to be surprisingly easy, because the generating function of a normalization constant often has a remarkably simple form. This has long been known in statistical mechanics. In that context, the normalization constant is usually referred to as the partition function and its generating function is referred to as the *grand partition function*; e.g., see pp. 213 and 347 of Reif [72]. Reiser and Kobayashi [73] used generating functions of normalization constants to derive their convolution algorithm. For more on generating functions of normalization constants in CQNs, see Bertozzi and McKenna [16].

We invert the p-dimensional generating function $\hat{g}(\mathbf{z})$ in (5.161) by recursively performing p one-dimensional inversions, using the algorithm in Section 1.2.4. To represent the recursive inversion, we define *partial generating functions* by

$$g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_j=0}^{\infty} g(\mathbf{K}) \prod_{i=1}^{j} z_i^{K_i} \quad \text{for} \quad 0 \le j \le p, \tag{5.162}$$

where $\mathbf{z}_j = (z_1, z_2, \ldots, z_j)$ and $\mathbf{K}_j = (K_j, K_{j+1}, \ldots, K_p)$ for $1 \le j \le p$. Let $\mathbf{z}_0$ and $\mathbf{K}_{p+1}$ be null vectors. Clearly, $\mathbf{K} = \mathbf{K}_1, \mathbf{z} = \mathbf{z}_p$, $g^{(p)}(\mathbf{z}_p, \mathbf{K}_{p+1}) = \hat{g}(\mathbf{z})$ and $g^{(0)}(\mathbf{z}_0, \mathbf{K}_1) = g(\mathbf{K})$.

Let I_j represent inversion with respect to z_j. Then the step-by-step nested inversion approach is

$$g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j) = I_j \left[g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1}) \right], \quad 1 \le j \le p, \tag{5.163}$$

starting with $j = p$ and decreasing j by 1 each step. In the actual program implementation, we attempt the inversion shown in (5.163) for $j = 1$. In order to compute the righthand side we need another inversion with $j = 2$. This process goes on until at step p the function on the righthand side becomes the p-dimensional generating function and is explicitly computable. By simply relabeling the p transform variables, we see that the scheme above can be applied to the p variables in any order. In all steps except the last we have a sequence of complex numbers, as in Section 1.2.4. For further discussion of the multi-dimensional inversion, see [24] and [20].

We now consider multi-chain closed queueing networks with only single-server queues (service centers with load-independent service rates) and (optionally) infinite-server queues. In this model all jobs are divided into *classes.* The combination of a class r job at queue i, (r, i) is called a *stage*. Two classes r and s communicate with each other if for some i and j, stage (s, j) can be reached from stage (r, i) in a finite number of steps (transitions) and vice versa. With respect to the relation of communication, all the classes can be divided into mutually disjoint equivalence classes called (closed) chains (ergodic sets in Markov chain theory). All classes within a chain communicate. No two classes belonging to different chains communicate. Since we are considering the steady-state distribution of a model with only closed chains, we do not need to consider any transient stages, i.e., stages (r, i) that will not be reached infinitely often. We now introduce further notation and give additional details about the model. For additional background, see Conway and Georganas [31], Lavenberg [61] or Chapter 12. We introduce the following features and notation:

- $p =$ number of closed chains
- $M =$ number of job classes ($M \geq p$).
- $N =$ number of queues (service centers). Queues $1, \ldots, q$ are assumed to be of the single-server type and queues $q + 1, \ldots, N$ are assumed to be of the infinite-sever (IS) type. As usual, for the single-server queues, the service discipline may be first-come first-served (FCFS), last-come first-served preemptive-resume (LCFSPR) or processor sharing (PS). In the case of FCFS, the service times of all job classes at a queue are assumed to be exponential with the same mean.
- $R_{ri,sj}$ = routing matrix entry, probability that a class r job completing service at queue i will next proceed to queue j as a class s job for $1 \leq i, j \leq N$, $1 \leq r, s \leq M$ (i.e., class hopping is allowed). The pair (r, i) is referred to as a stage in the network.
- K_j = number of jobs in the j^{th} closed chain, $1 \leq j \leq p$, which is fixed.
- $\mathbf{K} = (K_1, \ldots, K_p)$, the population vector, specified as part of the model data.
- n_{ri} = number of jobs of class r in queue i, $1 \leq r \leq M$, $1 \leq i \leq N$.
- n_i = number of jobs in queue i, i.e., $n_i = \sum_{r=1}^{M} n_{ri}$. $1 \leq i \leq N$.
- $\mathbf{n} = (n_{ri})$, $1 \leq r \leq M$, $1 \leq i \leq N$, the job vector, the queue lengths, the state variable.
- C_j = set of stages in the j^{th} closed chain. Clearly, $\sum_{(r,i) \in C_j} n_{ri} = K_j$, $1 \leq j \leq p$.
- $q_{ji} = \sum_{r:(r,i) \in C_j} n_{ri}$, number of jobs from chain j at queue i.

- $s(\mathbf{K})$ = state space of allowable job vectors or queue lengths (including those in service), i.e.,

$$S(\mathbf{K}) = \left\{ \mathbf{n} : n_{ri} \in \mathbf{Z}^{+} \quad \text{and} \quad \sum_{(r,i)\in C_j} n_{ri} = K_j, \quad 1 \le j \le p \right\}, \tag{5.164}$$

where $\mathbf{Z}^{+}$ is the set of nonnegative integers.

- e_{ri} = visit ratio, i.e., solution of the traffic rate equation

$$\sum_{(r,i)\in C_k} e_{ri} R_{ri,sj} = e_{sj} \quad \text{for all} \quad (s,j) \in C_k \quad \text{and} \quad 1 \le k \le p \, . \tag{5.165}$$

For each chain there is one degree of freedom in (5.165). Hence, for each chain j, the visit ratios $\{e_{ri} : (r,i) \in C_j\}$ are specified up to a constant multiplier.

- t_{ri} = the mean service time for class r at queue i.
- $\rho'_{ri} = t_{ri} e_{ri}$, $1 \le r \le M$, $1 \le i \le N$, the relative traffic intensities.
- $\rho_{j0} = \sum_{i=q+1}^{N} \sum_{(r,i)\in C_j} \rho'_{ri}$ and $\rho_{ji} = \sum_{(r,i)\in C_j} \rho'_{ri}$ for $i = 1, 2, \ldots, q$, the aggregate relative traffic intensities.

For this model, the steady-state distribution is given by (5.158) and the partition function is given by (5.159), where

$$f(\mathbf{n}) = \left[\prod_{i=1}^{q} n_i! \prod_{r=1}^{M} \frac{\rho'^{n_{ri}}_{ri}}{n_{ri}!} \right] \left[\prod_{i=q+1}^{N} \prod_{r=1}^{M} \frac{\rho'^{n_{ri}}_{ri}}{n_{ri}!} \right] . \tag{5.166}$$

The generating function $\hat{g}(\mathbf{z})$ is given by (5.161), using (5.159) and (5.166). By changing the order of summation, it can be seen that $\hat{g}(\mathbf{z})$ can be expressed remarkably simply as

$$\hat{g}(\mathbf{z}) = \frac{\exp\left(\sum_{j=1}^{p} \rho_{j0} z_j\right)}{\prod_{i=1}^{q} \left(1 - \sum_{j=1}^{p} \rho_{ji} z_j\right)} . \tag{5.167}$$

In general, there may be multiplicity in the denominator factors of (5.167) if two or more queues are identical with respect to visits by customers of all classes. In such a situation (5.167) becomes

$$\hat{g}(\mathbf{z}) = \frac{\exp\left(\sum_{j=1}^{p} \rho_{j0} z_j\right)}{\prod_{i=1}^{q'} \left(1 - \sum_{j=1}^{p} \rho_{ji} z_j\right)^{m_i}}, \tag{5.168}$$

where

$$\sum_{i=1}^{q'} m_i = q \ . \tag{5.169}$$

Here (5.168) is the preferred form, not (5.167); i.e., the key parameters are p and q'. The inversion algorithm simplifies by having different queues with identical single-server parameters.

Given the normalization constant $g(\mathbf{K})$ in (5.159) and (5.166), we can directly compute the steady-state probability mass function $p(\mathbf{n})$ in (5.158). Moreover, several important performance measures can be computed directly from ratios of normalization constants. For example, the throughput of class r jobs at queue i is

$$\theta_{ri} = e_{ri} \frac{g(\mathbf{K} - \mathbf{1}_j)}{g(\mathbf{K})} \quad \text{for} \quad (r, i) \in C_j, \tag{5.170}$$

where $\mathbf{1}_j$ is the vector with a 1 in the j^{th} place and 0's elsewhere.

From (5.170) we see that we will often need to compute normalization constants $g(\mathbf{K})$ for several closely related vector arguments $\mathbf{K}$. When the population vector $\mathbf{K}$ is large, it is possible to calculate such closely related normalization constants efficiently by exploiting shared computation; see [20]

The means $E[n_{ri}]$ and $E[q_{ji}]$ and higher moments $E[n_{ri}^k]$ and $E[q_{ji}^k]$ can also be computed directly from the normalization constants, but the standard formulas involve more than two normalization constant values. Choudhury, Leung and Whitt [20] developed an improved algorithm for means and higher moments via generating functions, which we now describe.

Given the steady-state probability mass function, we can calculate moments. Without loss of generality, let $(r, i) \in C_1$. We start with a standard expression for the probability mass function of q_{1i}, the number of chain 1 customers at queue i, namely,

$$P(q_{1i} = k) = \frac{\rho_{1i}^k (g(\mathbf{K} - k\mathbf{1}_1) - \rho_{1i} g(\mathbf{K} - (k+1)\mathbf{1}_1))}{g(\mathbf{K})}, \tag{5.171}$$

see (3.257) on p. 147 of Lavenberg [61]. (A similar expression holds for the mass function of n_{ri}. It involves ρ'_{ri} instead of ρ_{1i}.)

From the telescoping property of (5.171), we can write the tail probabilities as

$$P(q_{1i} \geq k) = \frac{\rho_{1i}^k g(\mathbf{K} - k\mathbf{1}_1)}{g(\mathbf{K})} \ . \tag{5.172}$$

From (5.172) we obtain the standard formula for the mean,

$$E[q_{1i}] = \sum_{k=1}^{\infty} P(q_{1i} \geq k) = \sum_{k=1}^{K_1} \rho_{1i}^k \frac{g(\mathbf{K} - k\mathbf{1}_1)}{g(\mathbf{K})}; \tag{5.173}$$

e.g., see (3.258) on p. 147 of Lavenberg [61]. Unfortunately, formula (5.173) is not too convenient, because it requires $K_1 + 1$ normalization function calculations and thus $K_1 + 1$ numerical inversions. We now show how this mean can be calculated by *two* inversions.

For this purpose, we rewrite (5.173) as

$$E[q_{1i}] = \frac{\rho_{1i}^{K_1} h(\mathbf{K})}{g(\mathbf{K})} - 1, \tag{5.174}$$

where

$$h(\mathbf{K}) = \sum_{k=0}^{K_1} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \ , \tag{5.175}$$

with K_2 defined after (5.162). Let $\hat{h}_1(z_1)$ be the generating function of $h(\mathbf{K})$ with respect to K_1. Then

$$\begin{aligned} \hat{h}_1(z_1) &= \sum_{m=0}^{\infty} z_1^m h(m, \mathbf{K}_2) = \sum_{m=0}^{\infty} z_1^m \sum_{k=0}^{m} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \\ &= \sum_{k=0}^{\infty} \rho_{1i}^{-k} g(k, \mathbf{K}_2) \sum_{m=k}^{\infty} z_1^m = \frac{g^{(1)}(z_1/\rho_{1i}, \mathbf{K}_2)}{1 - z_1}, \end{aligned} \tag{5.176}$$

where $g^{(1)}(\mathbf{z}_1, \mathbf{K}_2)$ is the partial generating function in (5.162). Now, if $\hat{h}(\mathbf{z})$ represents the full generating function of $h(\mathbf{K})$, then from (5.176) it is clear that

$$\hat{h}(\mathbf{z}) = \frac{\hat{g}(z_1/\rho_{1i}, z_2, \ldots, z_p)}{1 - z_1} \ . \tag{5.177}$$

Since $\hat{h}(\mathbf{z})$ is of the same form as $\hat{g}(\mathbf{z})$ it may be inverted by the established inversion procedure. Hence, we can obtain the mean $E[q_{1i}]$ using two inversions from (5.174). We invert $\hat{g}(\mathbf{z})$ and $\hat{h}(\mathbf{z})$, respectively, to obtain $g(\mathbf{K})$ and $h(\mathbf{K})$.

By the same approach, we can also calculate higher moments, see [20].

1.5.2 Dimension Reduction by Decomposition

In general, the inversion of a p-dimensional generating function $\hat{g}(\mathbf{z})$ represents a p-dimensional inversion, whether it is done directly or by our proposed recursive technique. This presents a major problem because the computational complexity of the algorithm is exponential in the dimension. Fortunately, however, it is often possible to reduce the dimension significantly by exploiting special structure. To see the key idea, note that if $\hat{g}(\mathbf{z})$ can be written as a product of factors, where no two factors have common variables, then the inversion of $\hat{g}(\mathbf{z})$ can be carried out by inverting the factors separately and the dimension of the inversion is thus reduced. For example, if $\hat{g}(z_1, z_2, z_3) = \hat{g}_1(z_1)\hat{g}_2(z_2, z_3)$, then $\hat{g}$ can be treated as one two-dimensional problem plus one one-dimensional problem, which is essentially a two-dimensional problem, instead of a three-dimensional problem.

We call the direct factorization of the generating function $\hat{g}$ an *ideal decomposition.* It obviously provides reduction of computational complexity, but we do not really expect to be able to exploit it, because it essentially amounts to having two or more completely separate models, which we would not have with proper model construction. We would treat them separately to begin with.

Even though ideal decomposition will virtually *never* occur, key model elements (e.g., closed chains) are often only *weakly coupled*, so that we can still exploit a certain degree of decomposition to reduce the inversion dimensionality, often dramatically. The idea is to look for *conditional decomposition.* The possibility of conditional decomposition stems from the fact that when we perform the $(j-1)^{\text{st}}$ inversion in (5.163), the outer variables $z_1, \ldots, z_{j-1}$ are fixed. Hence, for the $(j-1)^{\text{st}}$ inversion it suffices to look for decomposition in the generating functions regarded as a function of the remaining $p-j+1$ variables. For example, if $\hat{g}(z_1, z_2, z_3) = \hat{g}_1(z_1, z_2)\hat{g}_2(z_1, z_3)$, then for each fixed z_1, the transform $\hat{g}$ as a function of (z_2, z_3) factors into the product of two functions of a single variable. Hence $\hat{g}$ can be treated as two two-dimensional problems instead of one three-dimensional problems.

More generally, we select d variables that we are committed to invert. We then look at the generating function with these d variables fixed and see if the remaining function of $p-d$ variables can be factored. Indeed, we write the function of the remaining $p-d$ variables as a product of factors, where no two factors have any variables in common. The maximum dimension of the additional inversion required beyond the designated d variables is equal to the maximum number of the $p-d$ remaining variables appearing in one of the factors, say m. The overall inversion can then be regarded as being of dimension $d+m$. The idea, then, is to select an appropriate d variables, so that the resulting dimension $d+m$ is small.

This dimension reduction can be done whenever a multidimensional transform can be written as a product of factors. From (5.168), we see this structure always occurs with closed queueing networks. For closed queueing networks there is a factor for each queue in the network, and the variable z_j appears in the factor for queue i if and only if chain j visits queue i. Thus, conditional decomposition tends to occur when chains tend to visit relatively few queues. This property is called sparseness of routing chains in Lam and Lien [60]. As noted by Lam and Lien, this sparseness property is likely to be present in large communication networks and distributed systems.

To carry out this dimension reduction, we exploit the representation of the generating function $\hat{g}(\mathbf{z})$ as a product of separate factors, i.e.,

$$\hat{g}(\mathbf{z}) = \prod_{i=1}^{m} \hat{g}_i(\hat{\mathbf{z}}_i) \tag{5.178}$$

where $m \geq 2$ and $\hat{\mathbf{z}}_i$ is a subset of $\{z_1, z_2, \ldots, z_p\}$. We assume that each $\hat{g}_i(\hat{\mathbf{z}}_i)$ cannot be further factorized into multiple factors, unless at least one of the latter is a function of all variables in the set $\hat{\mathbf{z}}_i$.

We now represent the conditional decomposition problem as a graph problem. We construct a graph, called an *interdependence graph*, to represent the interdependence of the variables z_k in the factors. We let each variable z_k be represented by a node in the graph. For each factor $\hat{g}_i(\hat{\mathbf{z}}_i)$ in (5.178), form a fully connected subgraph Γ_i by connecting all nodes (variables) in the set $\hat{\mathbf{z}}_i$. Then let $\Gamma = \bigcup_{i=1}^{m} \Gamma_i$.

Now for any subset D of Γ, we identify the *maximal connected subsets* $S_i(D)$ of $\Gamma - D$; i.e., $S_i(D)$ is connected for each i, $S_i(D) \cap S_j(D) = \emptyset$ when $i \neq j$ and $\bigcup_i S_i(D) = \Gamma - D$. Let $|A|$ be the cardinality of the set A. Then the dimension of the inversion resulting from the selected subset D is

$$\text{inversion dimension } = |D| + \max_i\{|S_i(D)|\} \ . \tag{5.179}$$

It is natural to consider the problem of minimizing the overall dimension of the inversion. This is achieved by finding the subset D to achieve the following minimum:

$$\text{minimal inversion dimension } = \operatorname*{Min}_{D \subseteq \Gamma} \{|D| + \max_i\{|S_i(D)|\} \ . \tag{5.180}$$

In general, it seems difficult to develop an effective algorithm to solve this graph optimization problem; it seems to be an interesting research problem. However, for the small-to-moderate number of variables that we typically encounter, we can solve (5.180) by inspection or by enumeration of the subsets of Γ in increasing order of cardinality. Since our overall algorithm is likely to have difficulty if the reduced dimension is not relatively small (e.g., ≤ 10), it is not necessary to consider large sets D in (5.180). This dimension reduction is illustrated in the example in Section 1.5.4.

Even though it is not at first obvious, it turns out that the approach to dimension reduction developed by Choudhury, Leung and Whitt [20] is essentially equivalent to the tree algorithm of Lam and Lien [60] used with the convolution algorithm. The connection can be seen by noting that convolution of normalization constants corresponds to multiplication of the generating functions. Dimension reduction may be easier to understand with generating functions, because multiplication is a more elementary operation than convolution.

1.5.3 Scaling

In this subsection we discuss scaling to control the aliasing error. This additional step is needed here because, unlike probabilities, the normalization constants are not bounded.

To make the motivation clear, we specify the inversion in (5.163) in more detail. Letting $g_j(K_j) = g^{(j-1)}(\mathbf{z}_{j-1}, \mathbf{K}_j)$ and $\hat{g}_j(z_j) = g^{(j)}(\mathbf{z}_j, \mathbf{K}_{j+1})$, we can apply Theorem 8 to express the j^{th} step of the inversion in (5.163) as

$$g_j(K_j) = g_j^a(K_j) - e_j \ ,$$

where

$$g_j^a(K_j) = \frac{1}{2\ell_j K_j r_j^{K_j}} \sum_{k_1=1}^{l_j} e^{-\frac{\pi i k_1}{l_j}} \sum_{k_1=0}^{l_j-1} e^{-\frac{\pi i k_1}{l_j}} \sum_{k=-K_j}^{K_j-1} (-1)^k \hat{g}_j \left(r_j e^{\frac{\pi i (k_1 + l_j k)}{l_j K_j}} \right), \tag{5.181}$$

l_j is a positive integer, r_j is a suitably small positive real number and e_j represents the *aliasing error*, which is given by

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2nl_j K_j) r_j^{2nl_j K_j} \ . \tag{5.182}$$

Note that, for $j = 1$, $g_1(K_1) = g(\mathbf{K})$ is real, so that $\hat{g}_1(\overline{z}_1) = \overline{\hat{g}_1(z_1)}$. This enables us to cut the computation in (5.181) by about one half. For $j = 1$, we replace (5.181) by

$$\begin{aligned} g_1^a(K_1) \quad = \quad & \frac{1}{2\ell_1 K_1 r_1^{K_1}} \Bigg[\hat{g}_1(r_1) - (-1)^{K_1} \hat{g}_1(-r_1) + 2 \sum_{k_1=1}^{l_1} e^{\frac{-\pi i k_1}{l_1}} \\ & \times \sum_{k=0}^{K_1-1} \hat{g}_1 \left(r_1 e^{\pi i (k_1 + l_1 k)/l_1 K_1} \right) \Bigg] \ . \end{aligned} \tag{5.183}$$

To control the aliasing error (5.182), we choose

$$r_j = 10^{-\frac{\gamma_j}{2\ell_j K_j}} \ . \tag{5.184}$$

Inserting (5.184) into (5.182), we get

$$e_j = \sum_{n=1}^{\infty} g_j(K_j + 2nl_j K_j) 10^{-\gamma_j n} \ . \tag{5.185}$$

This choice of r_j enables us to more easily control the aliasing error e_j using the parameter γ_j. For instance, if g_j were bounded above by 1, as is the case with probabilities, then the aliasing error would be bounded above by $10^{-\gamma_j}/(1 - 10^{-\gamma_j}) \approx 10^{-\gamma_j}$.

As is clear from (5.185), a bigger γ_j decreases the aliasing error. However, since $r_j^{-K_j} = 10^{\gamma_j/2\ell j}$, the factor $r_j^{-K_j}$ in (5.181) increases sharply with γ_j and thus can cause roundoff error problems. Since the parameter l_j does not appear in the aliasing error term (5.185), it can be used to control the growth of $r_j^{-K_j}$ without altering the aliasing error. As indicated in Section 1.2.2, bigger values of l_j yield less roundoff error, but more computation because the number of terms in (5.181) is proportional to l_j.

Since the normalization constants can be arbitrarily large, the aliasing error e_j in (5.185) can also be arbitrarily large. Thus, in order to control errors, we scale the generating function in each step by defining a *scaled generating function* as

$$\hat{g}_j^*(z_j) = \alpha_{0j} \hat{g}_j(\alpha_j z_j), \tag{5.186}$$

where α_{0j} and α_j are positive real numbers. We invert this scaled generating function after choosing α_{0j} and α_j so that the errors are suitably controlled. Let $\overline{g}_j(K_j)$ represent the inverse function of $\hat{g}_j^*(z_j)$. The desired inverse function $g_j(K_j)$ may then be recovered from $\overline{g}_j(K_j)$ by

$$g_j(K_j) = \alpha_{0j}^{-1}\alpha_j^{-K_j}\overline{g}_j(K_j) \ . \tag{5.187}$$

A way to choose the parameters $\alpha_0, \alpha_1, \ldots, \alpha_p$ especially designed for CQNs was developed in Choudhury, Leung and Whitt [20]. Here we present a more general approach from Choudhury and Whitt [29] that can also be used in other contexts. For a p-dimensional inversion, our general strategy is to choose $p+1$ parameters $\alpha_0, \alpha_1, \ldots, \alpha_p$ so that

$$g_\alpha(\mathbf{K}) = \alpha_0 \prod_{j=1}^{p} \alpha_j^{K_j} g(\mathbf{K}) \tag{5.188}$$

is approximately a probability mass function (pmf) with $\mathbf{K}$ being approximately its mean vector. Then we should be able to control the discretization error in computing $g_\alpha(\mathbf{K})$ by acting as if it is bounded in $\mathbf{K}$. If we succeed in calculating $g_\alpha(\mathbf{K})$ with small discretization error, by (5.188), we succeed in calculating $g(\mathbf{K})$ with small relative discretization error.

The scaling algorithm is based on the following theorem.

Theorem 13 *Suppose that $\hat{g}(\mathbf{z})$ is the p-dimensional generating function of a nonnegative function $g(\mathbf{K})$. For any vector $(m_1, \ldots, m_p)$ of positive numbers, if the set of p equations*

$$z_i \frac{\partial}{\partial z_i} \log \hat{g}(z_1, \ldots, z_p)|_{z_j=\alpha_j \text{ for all } j} = m_i \ , \quad 1 \le i \le p \ , \tag{5.189}$$

has a solution $(\alpha_1, \ldots, \alpha_p)$ with $\hat{g}(\alpha_1, \ldots, \alpha_p) < \infty$, then $g_\alpha(\mathbf{K})$ in (5.188) with

$$\alpha_0 = \frac{1}{\hat{g}(\alpha_1, \ldots, \alpha_p)}$$

is a pmf with mean vector $(m_1, \ldots, m_p)$.

Proof. Note that

$$\begin{aligned} z_i \frac{\partial}{\partial z_i} \log \hat{g}(\mathbf{z})|_{\mathbf{z}=\boldsymbol{\alpha}} &= \frac{z_i \frac{\partial}{\partial z_i} \hat{g}(\mathbf{z})|_{\mathbf{z}=\boldsymbol{\alpha}}}{\hat{g}(\boldsymbol{\alpha})} \\ &= = \frac{\sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} K_i \prod_{j=1}^{p} \alpha_j^{K_j} g(\mathbf{K})}{\sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} \prod_{j=1}^{p} \alpha_j^{K_j} g(\mathbf{K})} = m_i \ . \end{aligned}$$

It remains to solve the p equations (5.189). Choudhury and Whitt [29] suggest using an iterative procedure, fixing all parameters except α_0 and α_i

for some i and the solving the single equation in (5.189) for i. Solving a single equation is relatively easy because the left side of (5.189) is strictly increasing in α_i. (See Theorem 7.1 of [29].) To speed up convergence, after a few initial steps, a Newton-Raphson root finding algorithm can be used.

To illustrate, we consider a CQN with two chains. Equation (5.167) then yields the generating function

$$\hat{g}(z_1, z_2) = \frac{\exp(\rho_{10} z_1 + \rho_{20} z_2)}{(1 - \rho_{11} z_1 - \rho_{21} z_2)(1 - \rho_{12} z_1 - \rho_{22} z_2)} . \tag{5.190}$$

We work with the scaled generating function

$$\hat{g}_\alpha(z_1, z_2) = \alpha_0 \hat{g}(\alpha_1 z_1, \alpha_2 z_2) . \tag{5.191}$$

The scaling parameters α_1 and α_2 are obtained from the two equations

$$\begin{aligned} n_i &= z_i \frac{\partial}{\partial z_i} \log \hat{g}(z_1, z_2)|_{z_1=\alpha_1, z_2=\alpha_2} \\ &= \alpha_i \left[\rho_{i0} + \sum_{j=1}^{q} \frac{\rho_{ij}}{1 - \sum_{k=1}^{2} \rho_{kj} \alpha_k} \right] \quad \text{for} \quad i = 1, 2 . \end{aligned} \tag{5.192}$$

We must solve the pair of nonlinear equations in (5.191). As suggested above, we can fix α_2 and search for the value α_1 that satisfies the equation for $i = 1$. Next, fixing α_1 at the value obtained, we search for the value α_2 that satisfies the equation for $i = 2$. We do this repeatedly until convergence is achieved based on some prescribed error criterion. We observed that this procedure indeed converges, but the rate of convergence becomes slow as n_1 and n_2 increases. By contrast, the two-dimensional Newton-Raphson method (see Press et al. [71], Chapter 9) converges very fast (less than 10 steps), provided that we start not too far from the root. So we initially use the search procedure a few times and then the Newton-Raphson method.

Here is a concrete with generating function (5.190). It corresponds to a CQN with two single-server queues, one infinite-server queue and two chains. Given (5.190), we need not specify all parameters. The relevant parameters are:

$$\begin{aligned} \rho_{1,0} &= 1 , \quad \rho_{2,0} = 1 , \quad \rho_{1,1} = 1 , \quad \rho_{2,1} = 2 , \\ \rho_{1,2} &= 2 , \quad \rho_{2,2} = 3 . \end{aligned}$$

Numerical results for several values of the chain populations n_1 and n_2 are displayed in Table 5.1. The accurate computation in the last case would be challenging by alternative algorithms. For that case, we use Euler summation in each dimension and it took only seconds. Accuracy was checked by performing two independent computations with two sets of inversion parameters.

Table 5.10 Numerical results for the normalization constant g_{n_1,n_2} in a closed queueing network with two chains.

n_1	n_2	g_{n_1,n_2}	α_1	α_2	α_0
3	2	0.243883E+04	0.240070	0.104428	0.806627E-01
30	20	0.627741E+33	0.294397	0.130311	0.589928E-02
300	200	0.973460E+331	0.299451	0.133032	0.564701E-03
3000	2000	0.235196E+3318	0.299945	0.133303	0.562179E-04

1.5.4 A Challenging Example

We conclude this section and this chapter by giving a challenging numerical example. We calculate the normalization constant $g(\mathbf{K})$ given by (5.159) and using (5.166) for specified population vectors $\mathbf{K}$ from the generating function $\hat{g}(\mathbf{z})$ in (5.168). Thus the parameters are the number of chains, p, the number of distinct single-server queues, q', the multiplicities m_i, the aggregate relative traffic intensities ρ_{ji}, $1 \le j \le p$, $0 \le i \le q'$, and the desired population vector $\mathbf{K}$.

From (5.168), note that the normalization constant $g(\mathbf{K})$ only depends on these parameters p, q', m_i, ρ_{ji} and $\mathbf{K}$. Hence, we do not fully specify the models below. In particular, we do not give the routing probabilities $R_{ri,sj}$ or the mean service times t_{ri}. Thus, there are many detailed models consistent with our partial model specifications. One possible routing matrix consistent with the data that we provide is a cyclic routing matrix, all of whose entries are 0's and 1's, which yields visit ratios $e_{ri} = 1$ for all stages (r, i) from (5.165). If we consider this case, then $t_{ri} = \rho'_{ri}$ and the throughputs θ_{ri} in (5.170) coincide with the normalization constant ratios $g(\mathbf{K} - \mathbf{1}_j)/g(\mathbf{K})$. We also display these ratios along with the values of $g(\mathbf{K})$ in our numerical results below. We note that the throughputs for any more detailed model can be found by solving (5.165) for the visit ratios e_{ri} and then applying (5.170).

The particular example that we consider has 11 closed chains and 1000 queues. Specifically, $p = 11$, $q = 1000$, $q' = 10$ and $m_i = 100$, $1 \le i \le 10$. A crucial step is dimension reduction, which reduces the effective dimension from 11 to 2. We consider three cases. First, let the chain populations be $K_j = 200$ for $2 \le j \le 11$. We obtain four subcases by considering four different values for K_1. Our numerical results are given below in Table 5.2.

The accuracy was checked by performing the calculations twice, once with $l_1 = 1$ and once with $l_1 = 2$.

The results in Table 5.2 were obtained in less than one minute by exploiting Euler summation with 51 terms. This example would seem to be out of the range of many other algorithms. For example, convolution would require a

Table 5.11 Numerical results for Case 1 of the 11-chain example.

chain populations K_j for $2 \leq j \leq 11$	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K}-\mathbf{1}_1)/g(\mathbf{K})$
200	20	1.232036e278	4.582983e-3
200	200	2.941740e579	4.281094e-2
200	2000	3.399948e2037	2.585489e-1
200	20,000	9.07177e8575	4.846930e-1

storage of size $200^{10} \times 2 \times 10^4 = 2.5 \times 10^{27}$ for the last case of Table 5.2. The last case would appear to be difficult even for the tree algorithm [60].

Some asymptotic methods require that there be an IS queue and that each chain visit this queue or that *all* chain populations be large. To show that our algorithm does not have such limitations, we consider two modifications of Case 1. Case 2 has classes 1 and 2 with small populations, while the other class populations remain large. In particular, we let $K_2 = 2$ and $K_3 = 5$. Numerical results for Case 2 appear below in Table 5.3.

Table 5.12 Numerical results for Case 2 of the 11-chain example.

chain populations K_j for $2 \leq j \leq 11$	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K}-\mathbf{1}_1)/g(\mathbf{K})$
in all cases:	2	3.842031e407	5.128582e-4
$K_2 = 2$,	20	1.484823e454	5.087018e-3
$K_3 = 5$ and	200	6.003231e747	4.706783e-2
$K_j = 200, 4 \leq j \leq 11$	2000	5.442693e2154	2.705391e-1
	20,000	2.494765e8617	4.852817e-1

Case 3 is a modification of Case 2 in which we remove all the IS nodes, i.e., we set $\rho_{j0} = 0$ for all j. Numerical results for Case 3 appear below in Table 5.4. As for Case 1, Cases 2 and 3 required about a minute on the SUN SPARC-2 workstation.

Before closing this section, we point out that if the model is such that we cannot take advantage of any of available speed-up techniques (namely, dimension reduction, fast summation of large sums and large queue multiplicities), then the inversion algorithm will be slower than the convolution algorithm, as indicated in [20]. Similarly, if dimension reduction is possible but none of the

Table 5.13 Numerical results for Case 3 of the 11-chain example.

chain populations K_j for $2 \leq j \leq 11$	K_1	normalization constant $g(\mathbf{K})$	ratio $g(\mathbf{K} - \mathbf{1}_1)/g(\mathbf{K})$
in all cases:	2	9.959619e313	4.762073e-4
$K_2 = 2$,	20	1.447107e361	4.728591e-3
$K_3 = 5$ and	200	1.222889e660	4.417444e-2
$K_j = 200, 4 \leq j \leq 11$	2000	2.948943e2096	2.645993e-1
	20,000	4.210541e8588	4.851015e-1

other speed-ups, then the inversion algorithm will be slower than the tree convolution algorithm, which also does dimension reduction in the time domain.

To illustrate, Lam and Lien [60] analyze an example with 64 queues and 32 chains, requiring about 3×10^7 operations. Choudhury, Leung and Whitt [20] analyzed this model and observed that the effective dimension can be reduced from 32 to 9. However, all chain populations are between 1 and 5 and so the speed-up technique based on Euler summation does not apply. Also there are no multiplicities. Choudhury et al. estimated that the operation count for this example would be about 10^{12}, so that the inversion algorithm is considerably slower than the tree convolution algorithm, even though the inversion algorithm is faster than the pure convolution algorithm, which has an operation count of about 10^{23}. It appears that the inversion algorithm nicely complements the tree algorithm, because the tree algorithm will be faster if the effective dimension after dimension reduction remains large but all chain populations are small. In contrast, the inversion algorithm will be faster if the effective dimension after dimension reduction is small (typically 5 or less) but some of the chain populations are large.

References

[1] J. Abate, G. L. Choudhury, D. M. Lucantoni, and W. Whitt. Asymptotic analysis of tail probabilities based on the computation of moments. *Ann. Appl. Prob.*, 5:983–1007, 1995.

[2] J. Abate, G. L. Choudhury, and W. Whitt. Calculation of the GI/G/1 waiting-time distribution and its cumulants from Pollaczek's formulas. *Archiv für Elektronik und Übertragungstechnik*, 47:311–321, 1993.

[3] J. Abate, G. L. Choudhury, and W. Whitt. Waiting-time tail probabilities in queues with long-tail service-time distributions. *Queueing Systems*, 16:311–338, 1994.

[4] J. Abate, G. L. Choudhury, and W. Whitt. Exponential approximations for tail probabilities in queues, I: waiting times. *Oper. Res.*, 43:885–901, 1995.

[5] J. Abate, G. L. Choudhury, and W. Whitt. On the Laguerre method for numerically inverting Laplace transforms. *INFORMS Journal on Computing*, 8:413–427, 1996.

[6] J. Abate, G. L. Choudhury, and W. Whitt. Numerical inversion of multidimensional Laplace transforms by the Laguerre method. *Performance Evaluation*, 31:229–243, 1998.

[7] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10:5–88, 1992.

[8] J. Abate and W. Whitt. Numerical inversion of probability generating functions. *Oper. Res. Letters*, 12:245–251, 1992.

[9] J. Abate and W. Whitt. Solving probability transform functional equations for numerical inversion. *Oper. Res. Letters*, 12:275–281, 1992.

[10] J. Abate and W. Whitt. Numerical inversion of Laplace transforms of probability distributions. *ORSA J. on Computing*, 7:36–43, 1995.

[11] J. Abate and W. Whitt. Calculating transient characteristics of the Erlang loss model by numerical transform inversion. *Stochastic Models*, 14:663–680, 1998.

[12] S. Asmussen. *Applied Probability and Queues*. Wiley, New York, 1987.

[13] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase type distributions via the EM algorithm. *Scand. J. Statist.*, 23:419–441, 1996.

[14] V. E. Beneš. The covariance function of a simple trunk group with applications to traffic measurements. *Bell System Tech. J.*, 40:117–148, 1961.

[15] V. E. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, 1965.

[16] A. Bertozzi and J. McKenna. Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review*, 35:239–268, 1993.

[17] A. A. Borovkov. *Asymptotics Methods in Queueing Theory*. Wiley, New York, 1984.

[18] M. L. Chaudhry, M. Agarwal, and J. G. C. Templeton. Exact and approximate numerical solutions of steady-state distributions arising in the GI/G/1 queue. *Queueing Systems*, 10:105–152, 1992.

[19] G. L. Choudhury, K. K. Leung, and W. Whitt. An algorithm to compute blocking probabilities in multi-rate multi-class multi-resource loss models. *Adv. Appl. Prob.*, 27:1104–1143, 1995.

[20] G. L. Choudhury, K. K. Leung, and W. Whitt. Calculating normalization constants of closed queueing networks by numerically inverting their generating functions. *J. ACM*, 42:935–970, 1995.

[21] G. L. Choudhury, K. K. Leung, and W. Whitt. Efficiently providing multiple grades of service with protection against overloads in shared resources. *AT&T Tech. J.*, 74:50–63, 1995.

[22] G. L. Choudhury, K. K. Leung, and W. Whitt. An inversion algorithm to compute blocking probabilities in loss networks with state-dependent rates. *IEEE/ACM Trans. Networking*, 3:585–601, 1995.

[23] G. L. Choudhury and D. M. Lucantoni. Numerical computation of the moments of a probability distribution from its transform. *Oper. Res.*, 44:368–381, 1996.

[24] G. L. Choudhury, D. M. Lucantoni, and W. Whitt. Multidimensional transform inversion with applications to the transient M/G/1 queue. *Ann. Appl. Prob.*, 4:719–740, 1994.

[25] G. L. Choudhury, D. M. Lucantoni, and W. Whitt. Squeezing the most out of ATM. *IEEE Trans. Commun.*, 44:203–217, 1996.

[26] G. L. Choudhury, D. M. Lucantoni, and W. Whitt. Numerical solution of $\mathrm{M_t/G_t/1}$ queues. *Oper. Res.*, 45:451–463, 1997.

[27] G. L. Choudhury and W. Whitt. Q^2: A new performance analysis tool exploiting numerical transform inversion. In *Proc. Third Int. Workshop on Modeling, Analysis and Simul. of Computer and Telecomm. Systems*, pages 411–415, Durham, NC, 1995. (MASCOTS '95).

[28] G. L. Choudhury and W. Whitt. Computing distributions and moments in polling models by numerical transform inversion. *Performance Evaluation*, 25:267–292, 1996.

[29] G. L. Choudhury and W. Whitt. Probabilistic scaling for the numerical inversion of non-probability transforms. *INFORMS J. Computing*, 9:175–184, 1997.

[30] J. W. Cohen. *The Single Server Queue.* North-Holland, Amsterdam, second edition, 1982.

[31] A. E. Conway and N. D. Georganas. *Queueing Networks – Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation.* MIT Press, Cambridge, MA, 1989.

[32] D. A. Darling and J. F. Siegert. The first passage problem for a continuous Markov process. *Ann. Math. Statist.*, 24:624–639, 1953.

[33] B. Davies and B. L. Martin. Numerical inversion of the Laplace transform: A survey and comparison of methods. *J. Comp. Phys.*, 33:1–32, 1979.

[34] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration.* Academic Press, New York, second edition, 1984.

[35] G. Doetsch. *Guide to Applications of Laplace Transforms.* Van Nostrand, London, 1961.

[36] G. Doetsch. *Introduction to the Theory and Application of the Laplace Transformation.* Springer-Verlag, New York, 1974.

[37] H. Dubner and J. Abate. Numerical inversion of Laplace transforms by relating them to the finite Fourier cosine transform. *J. ACM*, 15:115–123, 1968.

[38] N. G. Duffield and W. Whitt. A source traffic model and its transient analysis for network control. *Stochastic Models*, 14:51–78, 1998.

[39] F. Durbin. Numerical inversion of Laplace transforms: an efficient improvement to Dubner and Abate's method. *Comput. J.*, 17:371–376, 1974.

[40] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31:245–279, 1998.

[41] W. Feller. *An Introduction to Probability Theory and its Applications*, volume II. Wiley, New York, second edition, 1971.

[42] D. P. Gaver. Observing stochastic processes and approximate transform inversion. *Operations Research*, 14:444–459, 1966.

[43] D. P. Gaver and P. A. Jacobs. Waiting times when service times are stable laws: tamed and wild. In J. G. Shanthikumar and U. Sumita (eds.), editors, *Recent Contributions in Applied Probability and Stochastic Processes, Festschrift for Julian Keilson.* Kluwer, Boston, 1998.

[44] W. C. Giffin. *Transform Techniques for Probability Modeling.* Academic Press, New York, 1975.

[45] T. Hosono. Numerical inversion of Laplace transform. *J. Inst. Elec. Eng. Jpn.*, pages A54–A64, 1979. 494 (In Japanese).

[46] T. Hosono. Numerical inversion of Laplace transform and some applications to wave optics. *Radio Sci.*, 16:1015–1019, 1981.

[47] T. Hosono. *Fast Inversion of Laplace Transform by BASIC.* Kyoritsu Publishers, Japan, 1984. (In Japanese).

[48] D. L. Jagerman. Some properties of the Erlang loss function. *Bell System Tech. J.*, 53:525–551, 1974.

[49] D. L. Jagerman. An inversion technique for the Laplace transform with applications. *Bell System Tech. J.*, 57:669–710, 1978.

[50] D. L. Jagerman. An inversion technique for the Laplace transform. *Bell Sys. Tech. J.*, 61:1995–2002, 1982.

[51] R. Johnsonbaugh. Summing an alternating series. *Amer. Math. Monthly*, 86:637–648, 1979.

[52] E. P. C. Kao. *An Introduction to Stochastic Processes.* Duxbury Press, New York, 1997.

[53] J. Keilson. *Markov Chain Models – Rarity and Exponentiality.* Springer-Verlag, New York, 1979.

[54] J. Keilson and H. F. Ross. Passage time distributions for Gaussian Markov (Ornstein-Uhlenbech) statistical processes. *Selected Tables in Mathematical Statistics*, 3:233–327, 1975.

[55] J. F. C. Kingman. The single server queue in heavy traffic. *Proc. Camb. Phil. Soc.*, 57:902–904, 1961.

[56] L. Kleinrock. *Queueing Systems, Volume I: Theory.* Wiley, New York, 1975.

[57] C. Knessl. On the transient behavior of the M/M/m/m loss model. *Stochastic Models*, 6:749–776, 1990.

[58] H. Kobayashi. *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Addison-Wesley, Reading, MA, 1978.

[59] Y. K. Kwok and D. Barthez. An algorithm for the numerical inversion of the Laplace transform. *Inverse Problems*, 5:1089–1095, 1989.

[60] S. S. Lam and Y. L. Lien. A tree convolution algorithm for the solution of queueing networks. *Commun. ACM*, 26:203–215, 1983.

[61] S. S. Lavenberg, editor. *Computer Performance Modeling Handbook.* Academic Press, Orlando, FL, 1983.

[62] D. M. Lucantoni, G. L. Choudhury, and W. Whitt. The transient BMAP/G/1 queue. *Stochastic Models*, 10:145–182, 1994.

[63] D. Mitra and A. Weiss. The transient behavior in Erlang's model for large trunk groups and various traffic conditions. In *Teletraffic Science for new cost-Effective Systems, Networks and Services*, pages 1367–1374, Amsterdam, 1989. Elsevier-Science.

[64] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models.* The Johns Hopkins University Press, Baltimore, 1981.

[65] M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications.* Marcel Dekker, New York, 1989.

[66] C. A. O'Cinneide. Euler summation for Fourier series and Laplace transform inversion. *Stochastic Models*, 13:315–337, 1997.

[67] B. Van Der Pol and H. Bremmer. *Operational Calculus.* Cambridge Press, reprinted Chelsea Press, New York, 1987.

[68] F. Pollaczek. Fonctions caractéristiques de certaines répartitions définies au money de la notion d'ordre. Application à la théorie de attentes. *C. R. Acad. Sci. Paris*, 234:2334–2336, 1952.

[69] F. Pollaczek. Concerning an analytic method for the treatment of queueing problems. In W. L. Smith and W. E. Wilkinson, editors, *Proceedings of the Symposium on Congestion Theory*, pages 1–25 and 34–42, Chapel Hill, 1965. The University of North Carolina Press.

[70] A. D. Poularikas. *The Transforms and Applications Handbook.* CRC Press, Boca Raton, FL, 1996.

[71] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes, FORTRAN Version.* Cambridge University Press, Cambridge, England, 1988.

[72] F. Reif. *Fundamentals of Statistical and Thermal Physics.* McGraw-Hill, New York, 1965.

[73] M. Reiser and H. Kobayashi. Queueing networks with multiple closed chains: theory and computational algorithms. *IBM J. Res. Dev.*, 19:283–294, 1975.

[74] J. Riordan. *Stochastic Service Systems.* Wiley, New York, 1962.

[75] R. M. Simon, M. T. Stroot, and G. H. Weiss. Numerical inversion of Laplace transforms with applications to percentage labeled experiments. *Comput. Biomed. Res.*, 6:596–607, 1972.

[76] W. L. Smith. On the distribution of queueing times. *Proc. Camb. Phil. Soc.*, 49:449–461, 1953.

[77] R. Srikant and W. Whitt. Simulation run lengths to estimate blocking probabilities. *ACM J. TOMACS*, 6:7–52, 1996.

[78] L. Takács. *Introduction to the Theory of Queues.* Oxford University Press, New York, 1962.

[79] G. P. Tolstov. *Fourier Series.* Dover, New York, 1976.

[80] W. T. Weeks. Numerical inversion of Laplace transforms using Laguerre functions. *J. ACM*, 13:419–426, 1966.

[81] W. Whitt. Heavy-traffic approximations for service systems with blocking. *AT&T Bell Lab. Tech. J.*, 63:689–708, 1984.

[82] J. Wimp. *Sequence Transformations and Their Applications.* Academic Press, New York, 1981.

Joseph Abate received the B.S. degree from the City College of New York in 1961, and the Ph.D. degree in Mathematical Physics from New York University in 1967. From 1966 to 1970 he was employed by Computer Applications Corporation where he specialized in evaluating the performance of real-time computer systems. In 1971, he joined AT&T Bell Laboratories and retired in 1990. He has since served as a consultant to AT&T and Lucent Technologies. For most of his career, he worked on the design and analysis of transaction processing systems used in support of telephone company operations. For many years he has had a great passion for the use of Laplace transforms in queueing problems.

Gagan L. Choudhury received the B. Tech. degree in Radio Physics and Electronics from the University of Calcutta, India in 1979 and the MS and Ph.D. degrees in Electrical Engineering from the State University of New York (SUNY) at Stony Brook in 1981 and 1982, respectively. Currently he is a Technical Manager at the Teletraffic Theory and System Performance Department in AT&T Laboratories, Holmdel, New Jersey, USA. His main research interest is in the development of multi-dimensional numerical transform inversion algorithms and their application to the performance analysis of telecommunication and computer systems.

Ward Whitt received the A.B. degree in Mathematics from Dartmouth College, Hanover, NH, USA, in 1964 and the Ph.D. degree in Operations Research from Cornell University, Ithaca, NY, USA, in 1969. He was on the faculty of Stanford University and Yale University before joining AT&T Laboratories in 1977. He is currently a member of the Network Mathematics Research Department in AT&T Labs-Research in Florham Park, NJ, USA. His research has focused on probability theory, queueing models, performance analysis and numerical transform inversion.