NUMERICAL SOLUTION OF PIECEWISE-STATIONARY $M_t/G_t/1$ QUEUES

GAGAN L. CHOUDHURY

AT&T Laboratories, Holmdel, New Jersey

DAVID M. LUCANTONI

Isoquantic Technologies, Wayside, New Jersey

WARD WHITT

AT&T Laboratories, Murray Hill, New Jersey

(Received June 1993; revision received February 1995; accepted June 1995)

We develop an algorithm for computing the (exact) cumulative distribution function of the time-dependent workload in a piecewise-stationary $M_t/G_t/1$ queue with a work-conserving service discipline and general service-time distributions, where service times are determined at arrival instants. The t subscripts indicate that the arrival rate and the general service-time distribution may change with time, but we allow changes only at finitely many time points. The algorithm is based on numerical transform inversion, using the classical Takács double-transform of the transient workload in an $M/G/1$ queue recursively over the successive stationary intervals. In particular, we apply our recently developed Fourier-series-based inversion algorithms for two-dimensional transforms and nested one-dimensional transforms. We also do additional work to greatly speed up the computation while tightly controlling the error. As a consequence, the computation time grows only quadratically with the number of intervals. The algorithm is effective for ten or fewer intervals, where the intervals may have unlimited and possibly unequal lengths, typically running in at most a few minutes and maintaining high accuracy. We have also demonstrated that the algorithm can solve a 21-interval example with 7-to-10-digit accuracy in about half an hour. Models with only a few intervals are useful to study overload control strategies.

It has long been recognized that many queueing systems are most appropriately modeled by nonstationary queueing models, in which the arrival and service rates are functions of time; e.g., see Palm (1943), Koopman (1972), Green et al. (1991), and Chapter 6 of Hall (1991). In response to this need, there is a growing literature on methods for calculating time-dependent performance measures in nonstationary queueing models. The mainstay is clearly computer simulation, which is very appealing because of its flexibility and, more and more, also because of its ease of use. Nonstationary queueing models can be analyzed effectively by computer simulation by using multiple independent replications, but many replications are required to obtain good estimates of time-dependent probability distributions, especially if extremely small tail probabilities (e.g., 10^{-9}) are required, as in applications to high-speed communication networks. (We discuss simulation further in Section 9.)

Numerical solutions based on analytical expressions of time-dependent performance measures are an alternative to simulation. The mainstay here has no doubt been the numerical solution of time-dependent continuous-time Markov chains (CTMCs) by numerically solving a system of time-dependent ordinary differential equations (ODEs). The ODE approach was successfully used by Koopman (1972), Taaffe and Ong (1987), Ong and Taaffe (1989), Green et al. (1991), Zhang and Coyle (1991), Davis et al. (1995), and no doubt many others. The ODE approach applies naturally to the queue-length process in Markovian $M_t/M_t/s/r$ models and generalizations such as $Ph_t/Ph_t/s/r$ models involving time-dependent phase-type distributions. It is also possible to apply iterative techniques to discrete-time Markov chains (DTMCs), either directly or for CTMCs after applying uniformization (or randomization); e.g., see Gross and Miller (1984). New time-domain methods are also being developed; e.g., see Logothetics (1994). An important way to treat larger and more complicated models numerically is to exploit approximations; e.g., see Asmussen and Rolski (1994), Duda (1986), Ong and Taaffe (1989), Taaffe and Ong (1987), and references in these sources.

We suggest a different analytical approach: numerical transform inversion. The general idea is that we can obtain a two-dimensional transform of a desired time-dependent performance measure by taking the transform with respect to time as well as the performance state variable. Familiar examples are the two-dimensional transforms of the transient performance measures of the $M/G/1$ queue in Takács (1962). In Choudhury et al. (1994a) we developed numerical inversion algorithms to invert such multidimensional transforms, and showed that these algorithms are effective by applying them to compute transient performance measures in the $M/G/1$ queue. In Lucantoni et al. (1994) we derived corresponding transient performance measures for the more general *BMAP/G/1* queue (with a Batch Markovian Arrival Process) and again applied two-dimensional

Subject classifications: Queues, nonstationary: numerical solution of single-server models. Queues, algorithms. numerical transform inversion for nonstationary queues. Queues, transient results: two-dimensional numerical transform inversion.
Area of review: STOCHASTIC PROCESSES AND THEIR APPLICATIONS.

0030-364X/97/4503-0451 $05.00
© 1997 INFORMS

transform inversion to calculate the time-dependent performance measures in examples.

Our purpose here is to go beyond our previous transform inversion work computing time-dependent distributions of *stationary* models to computing time-dependent distributions of *time-dependent* models. In particular, we calculate the distribution of the workload (virtual waiting time), denoted by $W(t)$, at an arbitrary time t in a piecewise-stationary $M_t/G_t/1$ queue. This model has a single server, unlimited waiting room, a work-conserving service discipline (e.g., first-in first-out, last-in first-out, processor sharing, etc.), a Poisson arrival process, and i.i.d. service times with a general distribution that are independent of the arrival process. The t subscripts indicate that the arrival rate and the service-time distribution are allowed to depend on time, but the piecewise stationarity implies that changes can occur only at finitely many time points. The successive intervals can have unlimited and unequal lengths, and the traffic intensities can exceed one on some intervals. (In the context of the $M_t/M_t/1$ queue, piecewise-stationary models have also recently been considered by van den Berg and Groenendijk 1991 and Kuitenbrouwer 1992, but our methods are quite different.)

A customer arriving at time t has a service-time distribution that depends on t, but otherwise does not depend on the arrival process or the service times of other customers. (The service times are determined upon arrival.) The $M_t/G_t/1$ model also includes the $M_t^{X_t}/G_t/1$ model with a time-dependent batch-Poisson arrival process, because the workload process is the same as in an associated $M_t/G_t/1$ model with service times equal to the sum of all the service times in the batches.

The workload at time t is the remaining service time of the customer in service, if any, plus the sum of all the remaining service times of other customers in the system at time t. In the case of the first-in first-out (FIFO) discipline, the workload is also the virtual waiting time, the time that a hypothetical arrival at time t would have to wait before beginning service, i.e., the time required to complete service of all the customers in the system at time t, ignoring new arrivals after time t in non-FIFO disciplines. The sample path of the workload process decreases at rate 1 whenever there is work in the system and has jumps up at each arrival epoch equal to the service time of that arriving customer. In the $M_t/G_t/1$ model, the workload process is a nonstationary Markov process on the nonnegative half line. A difficulty for some methods is that the state space is the nonnegative real line, which is unbounded and uncountably infinite.

The $M_t/G_t/1$ model with periodic arrival rate has received quite a bit of attention (see Lemoine 1989, Rolski 1987 and 1989, Asmussen and Rolski 1994, and references in these sources), but not much work has been done on numerical methods. Our algorithm can be applied to periodic systems by iterating on the initial distribution, but we do not consider that here.

An alternative ODE-based numerical algorithm for calculating a time-dependent workload distribution in $M_t/Ph_t/1$ (and more general Markovian) models with finite waiting room was developed by Ong and Taaffe (1989). Their algorithm is based on computing the first passage time to the origin after time t for the Markovian queue-length process, assuming no new arrival after the time of interest. It is important to recognize that the Ong-Taaffe algorithm is for a different model, where the service-time distribution is determined when service begins and is in process, and not at arrival instants. It does not seem easy to treat our model with time-dependent service times determined upon arrival in their framework. The Ong-Taaffe algorithm should be effective for many models when the service-time distributions either do not change or are determined when service is in process, but it does not apply when the service-time distribution is not phase type, and the required computation will grow as the order of the phase-type distribution grows and the number of waiting spaces grows. In contrast, our algorithm is less sensitive to the form of the service-time distribution and has unlimited waiting room. Also, the accuracy of our algorithm is usually up to seven to ten digits, which is more than what is achievable by their fast approximation procedure.

Our main idea is that the classical two-dimensional transform expression for the transient workload distribution in an $M/G/1$ queue (see Section 3 of Chapter 1 in Takács 1962) can be used recursively, coupled with numerical transform inversion, to calculate the workload distribution in the piecewise-constant $M_t/G_t/1$ model. However, the $M_t/G_t/1$ model is substantially more complicated than the stationary $M/G/1$ model treated in Choudhury et al. (1994a). First, computation of the two-dimensional transform requires n recursive steps, where n is the number of previous piecewise-constant intervals. Furthermore, each recursive step requires the one-dimensional inversion of the transform of a complex quantity resulting in a nested set of inversions. A straightforward application of the recursive procedure has two difficulties: first, the computational effort grows exponentially with the number of intervals (a significant contribution of this paper is to show that an effective algorithm can be developed where the computational effort grows only quadratically with the number of intervals); second, the numerical error also grows rapidly in the nested inversion procedure. To address that problem, we use a special error-control procedure reported in Choudhury et al. (1994a). We first developed that error-control procedure for solving the $M_t/G_t/1$ model considered here.

To illustrate we consider a concrete example. We consider a time period of length 70 divided into 7 intervals, each of length 10. We let the service-time distribution be gamma with mean 1 and squared coefficient of variation (SCV, variance divided by the square of the mean) 4 in all intervals. We let the Poisson arrival process have rates 0.6, 0.9, 1.2, 1.5, 1.1, 0.8, and 0.5, respectively, in the seven intervals. Note that the arrival rate first increases and then decreases, achieving a maximum in the middle interval.

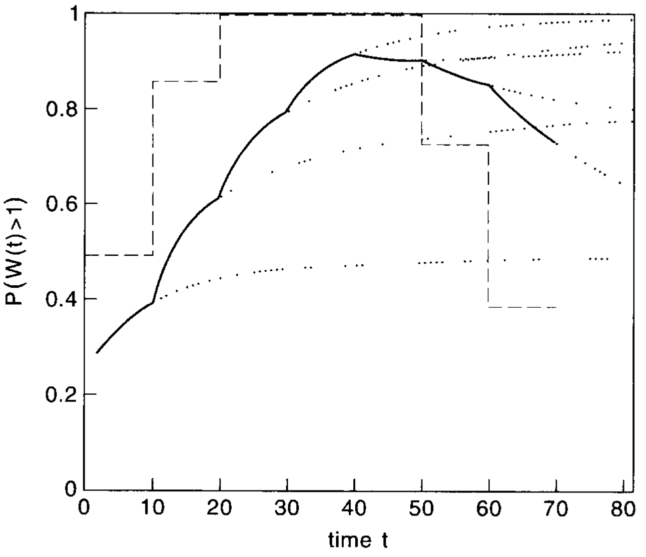

Figure 1. The time-dependent complementary workload distribution $P(W(t) > 1)$ for the first seven-interval example with gamma service times having mean 1 and SVC 4, arrival rates 0.6, 0.9, 1.2, 1.5, 1.1, 0.8, 0.5, and initial workload $W(0) = 1$. Here each interval has length 10. The dotted lines represent the probabilities if the arrival rates did not change at the end of each interval. The dashed lines represent the steady-state values associated with that interval.

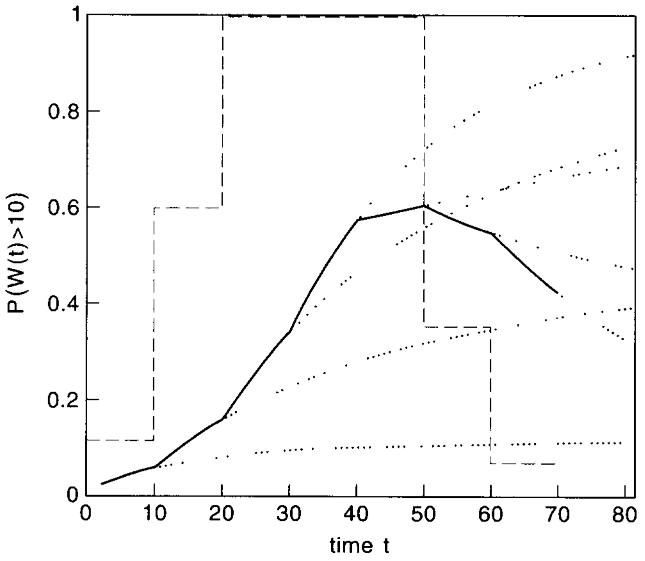

Figure 2. The time-dependent complementary workload distribution $P(W(t) > 10)$ for the first seven-interval example with gamma service times having mean 1 and SVC 4, arrival rates 0.6, 0.9, 1.2, 1.5, 1.1, 0.8, 0.5, and initial workload $W(0) = 1$. Here each interval has length 10. The dotted lines represent the probabilities if the arrival rates did not change at the end of each interval. The dashed lines represent the steady-state values associated with that interval.

Also note that the queue is instantaneously unstable (the arrival rate exceeds the service rate) during the third, fourth and fifth intervals.

In Figures 1 and 2 we plot the complementary workload distributions $P(W(t) > x)$ for the cases $x = 1$ and $x = 10$. These values are obtained by calculating the distributions at time values $2k$ for $k = 1, 2, \ldots, 35$. In addition to showing the transient behavior by a solid line, we also show by dotted lines how the system would have behaved if the arrival rates were not changed at the end of any particular interval. The dotted lines are obtained by another application of our algorithm, again using time values $2k$. The dotted lines asymptotically approach the steady state behavior associated with each interval. We also display by dashed lines in each interval the constant limiting values that would prevail if the rate in that interval held indefinitely in the past. The dashed lines are obtained by applying the EULER numerical inversion algorithm in Abate and Whitt (1992a) with the Pollaczek-Khintchine Laplace transform of the steady-state workload distribution.

From Figures 1 and 2, it is evident that the actual time-dependent workload distribution is very different from both the one-interval steady-state view and the one-interval transient view. This shows that the time-dependent analysis provided by an algorithm such as ours can be very important in the performance analysis of systems with time-varying arrival rates.

Of course, when the traffic intensities are less than one and the intervals are very long, the steady-state view becomes more appropriate. Our algorithm provides a means for studying the phenomenon. To illustrate, we reconsider the seven-interval example above with each interval being 100 instead of 10. Then we compute the time-dependent workload distribution at times $20k$ for $1 \leq k \leq 35$. This modified example is no harder to solve by our algorithm, but it would require simulation runs ten times as long.

Figures 3 and 4 display the numerically computed time-dependent workload tail probabilities $P(W(t) > x)$ for $x = 10$ and 100 in this longer interval example. With the longer intervals, the steady-state view is somewhat more appropriate in the first two intervals, where the traffic intensities are less than one, but still not very good. For example, the actual value of $P(W(t) > 100)$ is order-of-magnitude less than the steady-state value throughout the second interval. This is evident in Figure 4 since the tail probability is plotted in log scale.

In Figure 4, the steady-state view is also far off the mark in the last two intervals, where the traffic intensities are also less than one. The last two intervals are strongly affected by the preceding period of length 300 where the traffic intensities are greater than one. For this longer interval example, considerable insight can be gained by considering the deterministic fluid approximation, in which work arrives deterministically and continuously at a rate equal to the instantaneous traffic intensity. The fluid model

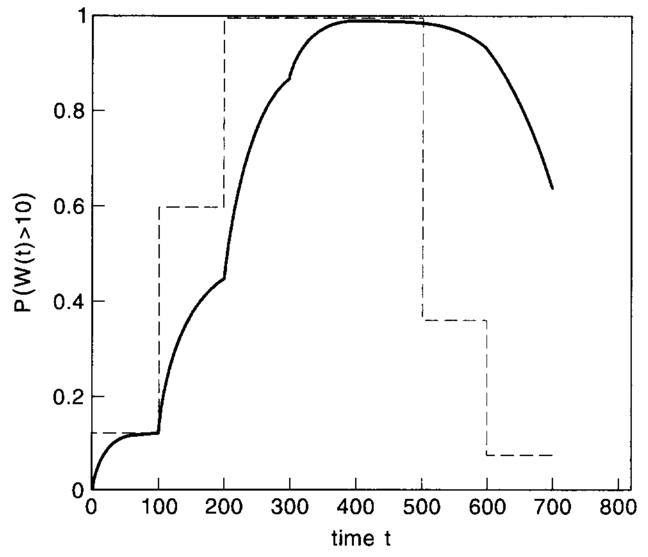

Figure 3. The time-dependent complementary workload distribution $P(W(t) > 10)$ for the second seven-interval example with gamma service times having mean 1 and SVC 4, arrival rates 0.6, 0.9, 1.2, 1.5, 1.1, 0.8, 0.5, and initial workload $W(0) = 1$. Here each interval has length 100. The dashed lines represent the steady-state values associated with that interval.

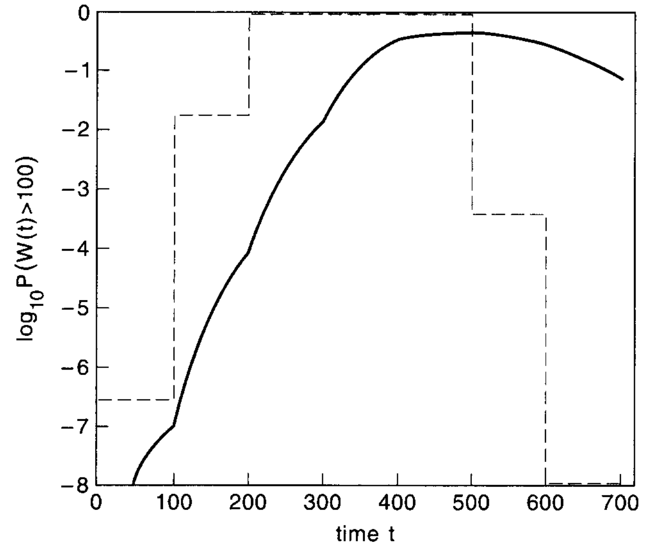

Figure 4. The time-dependent complementary workload distribution $P(W(t) > 100)$ for the second seven-interval example with gamma service times having mean 1 and SVC 4, arrival rates 0.6, 0.9, 1.2, 1.5, 1.1, 0.8, 0.5, and initial workload $W(0) = 1$. Here each interval has length 100. The dashed lines represent the steady-state values associated with that interval.

predicts workloads of 80, 60, and 10 at times 500, 600, and 700. However, Figures 3 and 4 show that the fluid model is not very accurate, either. (This conclusion still holds when the intervals are increased to 1000, which again is no more difficult to compute, using the same number of time points, i.e., time points at $200k$ for $1 \leq k \leq 35$.)

For our numerical algorithm, major issues are precision and computation time. We obtained the five numerical values in the last (most difficult) interval for two values of x with about (7–10)-digit precision on a SUN workstation (SPARCSTATION 10) using FORTRAN with standard double precision in three minutes of running time. (Throughout the paper, when we speak of precision, we refer to absolute precision; i.e., seven-digit precision means to 10^{-7}.) As discussed in Section 8, we calculate values in the last interval of a 21-interval example with similar precision in 26 minutes.

As indicated in Abate and Whitt (1992a), transform inversion gets difficult with a purely deterministic (D) service-time distribution; i.e., we get a high computational requirement to achieve reasonable precision. However, in Choudhury and Whitt (1997) we show that highly accurate and fast computation is possible in this case as well through an effective approximation scheme.

From an engineering point of view, the workload probabilities have two clear applications. One application is describing high percentiles of delays. For this purpose, we are typically interested in tail probabilities in the range 10^{-1} to 10^{-3}. For such probabilities, we can clearly do with less than 10^{-7} precision and thus may pursue some approximation procedure. Even simulation may be appropriate.

The second application is obtaining buffer overflow probabilities in communication networks in order to do proper buffer sizing, admission control, etc. For this application, the output channel typically has constant service rate and the variability in the service time comes from the variability of the message size. In this case, the workload gives the buffer occupancy, so that the workload tail probability approximately gives the buffer overflow probability. (The approximation is due to the infinite buffer assumption in the model.) For this application, workload tail probabilities are typically required in the range 10^{-6} to 10^{-10}, for which high precision is really needed. Fortunately, as the workload tail probabilities get smaller, the computational errors tend to get smaller too. Hence, when we say we get (7–10)-digit precision, we mean that we get seven-digit precision when the tail probability is near 1, and ten-digit precision when the tail probability is less than 10^{-3}. We usually can get 11-digit precision or more for tail probabilities of order 10^{-9} and 10^{-10}. Moreover, in this case, simulation tends to be inadequate; see Section 9.

Here is how the rest of this paper is organized. In Section 1 we present the two-dimensional Laplace transform of the time-dependent workload distribution and give an overview of our algorithm. In Section 2 we briefly describe the two-dimensional numerical transform inversion algorithm (in Choudhury et al. 1994a) that we use. We also describe the one-dimensional numerical transform inversion algorithm we use in the recursive algorithm; it is a modification of the EULER algorithm in Abate and Whitt (1992a) to cover the case in which the inverse of the transform is in general complex-valued instead of real-valued.

As indicated above, a straightforward application of this algorithm leads to computational difficulties, both in computation time and numerical precision. The numerical precision problem and the way to resolve it are explained in Sections 2 and 3. The computation time problem is explained and quantified in Section 4. In Section 5 we indicate how to speed up the computation by storing and reusing intermediate computational results. The modifications make the computational complexity grow approximately linearly or quadratically with the number of subintervals instead of exponentially.

In Section 6 we summarize the algorithm; those only interested in implementing the algorithm can go directly to Section 6 and refer to earlier sections only as needed. In Section 7 we describe a degenerate case that can cause numerical difficulties and indicate how to cope with it. In Section 8 we describe our numerical experience. There we indicate how we verify that we have produced an accurate computation. In Section 9 we make a comparison with simulation, indicating when our algorithm should be preferred. Finally, in Section 10 we briefly discuss ways to exploit approximations to speed up the computation.

We end this introduction by pointing out that our approach can also be applied to other processes such as queue-length processes and to more general models. We hope to discuss such extensions in future papers. We can treat more general models by applying the transient results for the *BMAP/G/*1 queue in Lucantoni et al. (1994). We can also calculate moments of the time-dependent distributions, as in Choudhury and Lucantoni (1996). In Choudhury and Whitt (1996) we have developed another recursive algorithm for computing and inverting transforms of performance measures arising in polling models. That procedure also applies to time-dependent models. For overviews of our recent work with numerical transform inversion, see Choudhury et al. (1994b) and Choudhury and Whitt (1995).

1. AN OVERVIEW OF THE ALGORITHM

Let $W(t)$ be the workload at time t, let $W(t, x)$ be its cdf (cumulative distribution function), and let $W^c(t, x)$ be its complementary cdf, i.e.,

$$W(t, x) = P(W(t) \leq x) \quad \text{and} \quad W^c(t, x) = 1 - W(t, x).$$

Let $\hat{w}(t, s)$ be the Laplace-Stieltjes transform of $W(t, x)$ with respect to the *space variable* x and let $\bar{w}(\xi, x)$ be the Laplace transform of $W(t, x)$ with respect to the *time variable* t, respectively; i.e.,

$$\hat{w}(t, s) = \int_0^\infty e^{-sx}\, d_x\, W(t, x), \quad \text{and} \tag{1}$$

$$\bar{w}(\xi, x) = \int_0^\infty e^{-\xi t} W(t, x)\, dt, \tag{2}$$

where s and ξ are complex variables with $\text{Re}(s) > 0$ and $\text{Re}(\xi) > 0$. Let $\tilde{w}(\xi, s)$ be the Laplace transform of $\hat{w}(t, s)$, i.e.,

$$\tilde{w}(\xi, s) = \int_0^\infty e^{-\xi t} \hat{w}(t, s)\, dt, \tag{3}$$

where $\text{Re}(\xi) > 0$.

Let $\hat{w}^c(t, s)$ be the Laplace transform of $W^c(t, x)$ with respect to the variable x and let $\tilde{w}^c(\xi, s)$ be the Laplace transform of $\hat{w}^c(t, s)$ with respect to the variable t, i.e.,

$$\hat{w}^c(t, s) = \int_0^\infty e^{-sx} W^c(t, x)\, dx \quad \text{and}$$

$$\tilde{w}^c(\xi, s) = \int_0^\infty e^{-\xi t} \hat{w}^c(t, s)\, dt. \tag{4}$$

(Note that $\hat{w}^c(t, s)$ in (4) is the Laplace transform, whereas $\hat{w}(t, s)$ in (1) is the Laplace-Stieltjes transform.) It can be shown that

$$\tilde{w}^c(\xi, s) = \frac{1}{s\xi} - \frac{\tilde{w}(\xi, s)}{s}. \tag{5}$$

The form of Equation (5) suggests a possible numerical problem for very small s or ξ, but no difficulty occurs since the inversion algorithm never has to compute the transform at very small s or ξ. As will be shown later, the inversion summation is done along contours parallel to the imaginary axes of the complex s and ξ planes with $\text{Re}(\xi) = A_1/2tl_1$ and $\text{Re}(s) = A_2/2xl_2$, where t is the time point, x is the state variable and A_1, A_2, l_1, l_2 are positive parameters of the inversion. Thus, $\text{Re}(\xi)$ and $\text{Re}(s)$ may be close to zero only if t or x are very large. However, it can be shown that, if t or x are very large, then we can always increase A_1 or A_2 as well, thereby ensuring that $\text{Re}(\xi)$ or $\text{Re}(s)$ are never too close to zero.

We intend to compute $W^c(t, x)$ by numerically inverting the double transform $\tilde{w}^c(\xi, s)$. However, since $\tilde{w}^c(\xi, s)$ is directly related to $\tilde{w}(\xi, s)$, in the rest of the paper we show how to get the latter. We compute $W^c(t, x)$ instead of $W(t, x)$ because we are often interested in small tail probabilities (say 10^{-6} to 10^{-9}) and in that range it is possible to compute $W^c(t, x)$ significantly more accurately, as explained in Section 2.

To proceed further, we now assume that the arrival rate and the service-time cdf change at only finitely many points. In addition, suppose that the time of interest is inside the nth stationary interval and t represents the time since the beginning of the nth interval. Let the length of the jth stationary interval be t_j, $1 \leq j \leq n-1$. Hence, the time of interest is $t_1 + \ldots + t_{n-1} + t$. Let λ_j be the *arrival rate* and let H_j be the *service-time cdf* in the jth interval. Let $\hat{h}_j$ be the Laplace-Stieltjes transform of H_j, i.e.,

$$\hat{h}_j(s) = \int_0^\infty e^{-sx}\, dH_j\,(x), \quad 1 \leq j \leq n.$$

Moreover, let the workload cdf and its transforms have subscripts to indicate the stationary interval they are associated with. Within each interval let time be measured from the beginning of that interval. With this convention, we want to compute $W_n(t, x)$, where it is understood that $W_j(0, x) = W_{j-1}(t_{j-1}, x)$, $2 \leq j \leq n$, and $W_1(0, x)$ is the initial workload cdf, which is specified as part of the model data.

As mentioned in the introduction, the main idea is to use the classical double transform for the transient workload distribution in the stationary *M/G/*1 model recursively over the successive intervals. For the jth interval, the double transform $\tilde{w}_j(\xi, s)$ can be expressed as

$$\tilde{w}_j(\xi, s) = \frac{\hat{w}_{j-1}(t_{j-1}, s) - s\tilde{w}_j(\xi, 0)}{\xi - s + \lambda_j - \lambda_j \hat{h}_j(s)}, \tag{6}$$

where $t_0 = 0$ and $\hat{w}_0(t_0, s) = \hat{w}_1(0, s)$ is the Laplace-Stieltjes transform of the *initial workload* cdf $W(0, x) \equiv W_1(0, x)$ and $\tilde{w}_j(\xi, 0)$ is the Laplace transform of the *emptiness function* in the jth interval, $W_j(t, 0)$. Our result in (6) is obtained from (15) on page 53 plus (9) on page 52 of Takács (1962), which is (39) of Lucantoni et al. (1994). Takács gives greater emphasis to the single transform $\hat{w}(t, s)$ in (1) in his (8) on page 51, but it is easier to compute using the double transform in (6) since we have an efficient double-transform inversion algorithm. (See Abate and Whitt 1994 for more references and discussion on the transient behavior of the *M/G/*1 queue.)

The emptiness transform $\tilde{w}_j(\xi, 0)$ in (6) can be expressed as

$$\tilde{w}_j(\xi, 0) = \frac{\hat{w}_{j-1}(t_{j-1}, \xi + \lambda_j - \lambda_j \hat{g}_j(\xi))}{\xi + \lambda_j - \lambda_j \hat{g}_j(\xi)}, \tag{7}$$

where $\hat{g}_j(s)$ is the Laplace-Stieltjes transform of the busy-period cdf, say G_j, associated with the jth stationary interval, i.e.,

$$\hat{g}_j(s) = \int_0^\infty e^{-sx}\, dG_j(x), \tag{8}$$

which satisfies the functional equation

$$\hat{g}_j(s) = \hat{h}_j(s + \lambda_j - \lambda_j \hat{g}_j(s)); \tag{9}$$

e.g., see (10) on page 52 of Takács (1962), Abate and Whitt (1992b), and (15) and (24) of Lucantoni et al. (1994).

For any desired integer j and complex s with $\mathrm{Re}(s) > 0$, we calculate the busy-period transform $\hat{g}_j(s)$ by iterating (9). As indicated in Abate and Whitt (1992b), when $\rho_j < 1$, where ρ_j is the traffic intensity associated with the jth interval, this iterative scheme always converges to the unique solution of (9) for complex s with $\mathrm{Re}(s) > 0$. Moreover, upper and lower bounds can be obtained by starting the iteration with 1 and 0 for $\hat{g}_n(s)$ in the right side of (9). When $\rho \geq 1$, the iteration still converges, but it has to start with 0 to get the correct solution. In our numerical examples, we stop the iteration when successive iterates differ in absolute value by no more than 10^{-13}.

Thus, we obtain the desired complementary cdf value $W_n^c(t, x)$ by applying two-dimensional numerical transform inversion to the double transform $\tilde{w}_n^c(\xi, s)$ in (5), which is obtained from $\tilde{w}_j(\xi, s)$ in (6) by setting $j = n$. This requires computation of the emptiness function from (7) and the busy-period transform from (9), in each case setting $j = n$. The computation requires the transform values $\hat{w}_{n-1}(t_{n-1}, s)$ and $\hat{w}_{n-1}(t_{n-1}, \xi + \lambda_n - \lambda_n \hat{g}_n(\xi))$, which we obtain by applying one-dimensional numerical transform inversion with respect to the time variable ξ to the double transform $\tilde{w}_{n-1}(\xi, s)$, which we denote by

$$\hat{w}_{n-1}(t_{n-1}, s) = L_\xi^{-1}(\tilde{w}_{n-1}(\xi, s)). \tag{10}$$

This last step requires that we go back to (6), but this time with $j = n - 1$, and thereby starting a recursive procedure.

In summary, we perform one two-dimensional numerical inversion of $\tilde{w}_n^c(\xi, s)$ which requires calculating $\tilde{w}_n(\xi, s)$ for various (ξ, s) pairs. To calculate $\tilde{w}_n(\xi, s)$ we perform $(n - 1)$ stages of one-dimensional numerical inversions with respect to the time variable ξ of the form (10). For these one-dimensional inversions, the second variable is fixed at either s or $\xi + \lambda_j - \lambda_j \hat{g}_j(\xi)$, $1 \leq j \leq n - 1$. In Sections 4 and 5 we show that it is important to exploit this special structure.

2. TRANSFORM INVERSION ALGORITHMS

We saw in the last section that we need to invert the double transform $\tilde{w}_n^c(\xi, s)$ and, in the recursive procedure, we need to invert $\tilde{w}_j(\xi, s)$ for $j = n - 1, n - 2, \ldots 1$ only with respect to the ξ variable. We briefly state the inversion algorithms.

Let $f(t_1, t_2)$ be a real-valued function of two nonnegative real variables t_1 and t_2 with a proper double Laplace transform

$$\tilde{f}(s_1, s_2) = \int_0^\infty \int_0^\infty f(t_1, t_2) e^{-(s_1 t_1 + s_2 t_2)}\, dt_1\, dt_2. \tag{11}$$

Simplifying Equation (2.11) in Choudhury et al. (1994a), we get the double-transform inversion formula

$$f(t_1, t_2) = \frac{\exp(A_1/2l_1)}{2l_1 t_1} \Bigg[\hat{f}\Big(\frac{A_1}{2l_1 t_1}, t_2\Big) + 2 \sum_{j=1}^{l_1} \sum_{j_1=0}^{\infty} (-1)^{j_1} \cdot \mathrm{Re}\Big[\exp\Big(-\frac{ij\pi}{l_1}\Big) \hat{f}\Big(\frac{A_1}{2l_1 t_1} - \frac{ij\pi}{l_1 t_1} - \frac{ij_1 \pi}{t_1}, t_2\Big)\Big]\Bigg] - e_d, \tag{12}$$

where $i = \sqrt{-1}$,

$$\hat{f}(s_1, t_2) = \frac{\exp(A_2/2l_2)}{2l_2 t_2}\left[\tilde{f}\left(s_1, \frac{A_2}{2l_2t_2}\right) + \sum_{k=1}^{l_2}\sum_{k_1=0}^{\infty}(-1)^{k_1}\exp\left(-\frac{ik\pi}{l_2}\right) \cdot \tilde{f}\left(s_1, \frac{A_2}{2l_2t_2} - \frac{ik\pi}{l_2t_2} - \frac{ik_1\pi}{t_2}\right) + \sum_{k=1}^{l_2}\sum_{k_1=0}^{\infty}(-1)^{k_1}\exp\left(\frac{ik\pi}{l_2}\right) \cdot \tilde{f}\left(s_1, \frac{A_2}{2l_2t_2} + \frac{ik\pi}{l_2t_2} + \frac{ik_1\pi}{t_2}\right)\right], \tag{13}$$

and e_d represents the aliasing error in the inversion. Since $f(t_1, t_2)$ represents a probability in our case, it can be shown that e_d is bounded by

$$|e_d| \le \frac{(e^{-A_1} + e^{-A_2} - e^{-(A_1+A_2)})}{(1-e^{-A_1})(1-e^{-A_2})} \simeq (e^{-A_1} + e^{-A_2}). \tag{14}$$

The parameters A_1, A_2, l_1, l_2 are for error control and will be explained in Section 3.

Next, let $f(t)$ be a complex-valued function with a proper Laplace transform

$$\hat{f}(s) = \int_0^\infty f(t)e^{-st}\,dt. \tag{15}$$

Modifying the Euler algorithm in Abate and Whitt (1992a), we get the inversion formula as

$$f(t) = \frac{\exp(A/2l)}{2lt}\left[\hat{f}\left(\frac{A}{2lt}\right) + \sum_{k=1}^{l}\sum_{k_1=0}^{\infty}(-1)^{k_1}\exp\left(-\frac{ik\pi}{l}\right) \cdot \hat{f}\left(\frac{A}{2lt} - \frac{ik\pi}{lt} - \frac{ik_1\pi}{t}\right) + \sum_{k=1}^{l}\sum_{k_1=0}^{\infty}(-1)^{k_1}\exp\left(\frac{ik\pi}{l}\right) \cdot \hat{f}\left(\frac{A}{2lt} + \frac{ik\pi}{lt} + \frac{ik_1\pi}{t}\right)\right] - e_d. \tag{16}$$

The parameters A and l in (16) are for error control and will be explained in Section 3. In (16), e_d is the aliasing error, which is bounded by

$$|e_d| \le \frac{ce^{-A}}{1-e^{-A}} \simeq ce^{-A}, \tag{17}$$

where c is a constant such that

$$|f(t)| \le c \quad \text{for all } t. \tag{18}$$

In the current context

$$f(t) = \hat{w}_j(t, s) = \int_0^\infty e^{-sx}\,d_x\,W_{n-1}(t, x), \tag{19}$$

so that

$$|f(t)| \le \int_0^\infty |e^{-sx}|\,d_x\,W_{n-1}(t,x) \le \int_0^\infty d_x\,W_{n-1}(t,x) \equiv 1 \quad \text{for } \mathrm{Re}(s) > 0. \tag{20}$$

Therefore, we get the error bound in (17) with $c = 1$.

Equations (12), (13) and (16) contain infinite sums of the form $s = \sum_{k=0}^{\infty}(-1)^k a_k$ where a_k is real or complex. Such a sum may be efficiently approximated by the Euler sum where

$$E(m_1, n_1) \equiv \sum_{k=0}^{m_1}\binom{m_1}{k}2^{-m_1}S_{n_1+k}, \quad \text{where} \tag{21}$$

$$S_j \equiv \sum_{k=0}^{j}(-1)^k a_k. \tag{22}$$

The total number of terms considered is $K \equiv m_1 + n_1$. Using (21) we can accurately estimate each of the infinite sums appearing in (12), (13), and (16).

3. ERROR CONTROL AND PARAMETER CHOICE

The parameters A_1, A_2, l_1, l_2 in (12), (13), A, l in (16), and m_1, n_1 in (21) are for error control. We briefly explain their role and prescribe their numerical values for good accuracy. See Abate and Whitt (1992a) and Choudhury et al. (1994a) for further explanation. First, unless the service-time distribution has discontinuities, the choice $n_1 = 39$, $m_1 = 11$ and hence $K = 50$ typically computes each infinite sum pretty accurately (with errors below 10^{-11}). This would not be true with deterministic or discrete service-time distributions, and much bigger K would be needed for good accuracy. However, in Choudhury and Whitt (1997) we describe a smoothing procedure that effectively eliminates the discreteness problem.

Next, the aliasing error bounds given by (14) and (17) may be controlled by choosing A_1, A_2, and A large. However, this increases the round-off error. The round-off error may be reduced by increasing the parameters l_1, l_2, l. In fact we introduced the parameters l_1, l_2, l for the express purpose of round-off error control in the $M_t/G_t/1$ model, but have reported it since then in Choudhury et al. (1994a). As mentioned in Remark 5.8 of Abate and Whitt (1992a), the one-dimensional Euler algorithm with $l = 1$ would retain only 2/3 of the precision. Thus, after $(n - 1)$ stages of recursion and a final double transform inversion, roughly about $(2/3)^{n+1}$ of the precision will be retained in the final answer. With $l_1 = l_2 = l > 1$, we roughly retain ρ^{n+1} of the precision in the final answer, where ρ may be made pretty close to 1. Choosing suitably large l_1, l_2, l thereby allows accurate computation even with large n.

There is a trade-off between error control and computation time, since whenever l_1, l_2, or l is increased the computation time increases proportionally, and increasing A_1, A_2 or A requires a corresponding increase in l_1, l_2, or l. Also, tighter error control is needed as the number of

Table I
Parameter Values Required as a Function of the Number of Intervals in Order to Achieve (7–10)-digit Accuracy

Number of Intervals	Parameter Values	
	$A_1 = A_2 = A$	$l_1 = l_2 = l$
1–2	22	2
3–5	24	3
6–8	26	4
9–12	26	5
13–16	26	6
17–21	26	7

Table II
Estimated Computational Complexity of the Direct Inversion Algorithm Without Enhancements

Computations	Estimated number
$\tilde{w}_n(\xi, s)$	$N_1 \cdot N_2 \cdot 2K^2l^2$
$\tilde{w}_j(\xi, s), j \neq n$	$\frac{N_1N_2}{8} \cdot (4Kl)^{n+1}$
$\hat{w}_j(t_j, s), j \neq n$	$\frac{N_1N_2}{4} \cdot (4Kl)^{n+1}$
iterations for $\hat{g}_j(\xi)$, $1 \leq j \leq n$	$\frac{N_1N_2}{8} (4Kl)^{n+1}I$

intervals increases since more intervals means more nested inversion and each inversion introduces error.

We have observed (based on a wide range of service-time distributions) what parameter values are required to get seven- to ten-digit accuracy. First, as stated earlier, we need $n_1 = 39$ and $m_1 = 11$. The remaining parameters depend on the number of intervals (starting from the beginning until the current one and are given in Table I. (There is no dependence on future intervals.)

Table I shows only one parameter value for all parameters. In fact, it can be advantageous to use different parameter values for the different levels of the recursion. We can get by with less stringent error control in the outer levels of the inversion procedure. Specifically, we get a speed-up by a factor of about 2 to 3 in the seven-interval and 21-interval examples, without affecting accuracy, using this procedure. See Section 8 for the actual parameter values used.

4. INITIAL DIFFICULTIES WITH THE ALGORITHM

A straightforward implementation of the algorithm leads to serious numerical difficulties for a large number n of stationary intervals, both in terms of the computation time and the precision of the final answer. We first give an intuitive explanation for these difficulties and then give quantitative details.

We not only need to do a two-dimensional transform inversion, but as part of computing the double transform we need to perform $(n - 1)$ stages of single transform inversion of the form (10). Moreover, the single transform inversions are nested, i.e., the jth single transform inversion depends on the $(j - 1)$st single transform inversion for $j = 2, 3, \ldots, n - 1$. Therefore, at first glance, it appears that the problem is as difficult as an $(n + 1)$-dimensional transform inversion problem, where the computational requirement grows geometrically with n and precision is quickly lost. The difficulty is further compounded by the fact that the busy-period-transform values $g_j(\xi)$ are not directly available and have to be computed iteratively. The difficulty with precision has already been discussed in Section 3. Now we quantify approximately the difficulties with computation time. For simplicity, we assume that the same values of $A_1 = A_2 = A$ and $l_1 = l_2 = l$ for each inversion.

Suppose that we want to compute $W_n(t, x)$ for $t = \tau_1, \tau_2, \ldots, \tau_{N_1}$ and for $x = x_1, x_2, \ldots, x_{N_2}$ (for $N_1 \cdot N_2$ different values) by inverting the double transform $\tilde{w}_n(\xi, s)$ in (6). Note that computation of $W_n(t, x)$ requires about $2K^2l^2$ computations of $\tilde{w}_n(\xi, s)$, where K is the number of terms from the infinite series required for Euler summation, i.e., $K = m_1 + n_1$ in (21). Each computation of $\tilde{w}_n(\xi, s)$ requires one computation of $\hat{g}_n(\xi)$ and two computations of $\hat{w}_{n-1}(t_{n-1}, s_1)$, one at $s_1 = s$ and another at $s_1 = \xi + \lambda_n - \lambda_n\hat{g}_n(\xi)$ (see (6) and (7)).

Each computation of the busy-period transform $\hat{g}_n(\xi)$ requires a number of iterations on the busy-period functional equation (9). In general, the number of iterations for $\hat{g}_j(\xi)$ depends on ξ and other system parameters for interval j. However, for the purpose of characterizing computational complexity, we assume I to be the number of iterations required for all cases. Therefore, each computation of $\tilde{w}_n(\xi, s)$ requires I iterations of $\hat{g}_n(\xi)$.

Each computation of $\hat{w}_{n-1}(t_{n-1}, s)$ using (10) requires about $2Kl$ computations of $\tilde{w}_{n-1}(\xi, s)$. For $j = n - 1, n - 2, \ldots, 1$, each computation of $\tilde{w}_j(\xi, s)$ requires I iterations of $\hat{g}_j(\xi)$ and two computations of $\hat{w}_{j-1}(t_{j-1}, s_1)$ at $s_1 = s$ and at $s_1 = \xi + \lambda_j - \lambda_j\hat{g}_j(\xi)$. For $j = n - 1, n - 2, \ldots, 2$, each computation of $\hat{w}_{j-1}(t_{j-1}, s)$ requires about $2Kl$ computations of $\tilde{w}_{j-1}(\xi, s)$.

From the above, we get computational requirements as depicted in Table II. The first row of Table II represents the *basic computational requirements* for two-dimensional transform inversion given that the double transform is directly computable. The last three rows of Table II represent *additional computations due to the recursive scheme* for obtaining the double transform.

Note that the basic computations are independent of n and are proportional to K^2l^2. In contrast, the additional computations grow geometrically with n and become orders of magnitude bigger than the basic computations as n grows. Both the basic and the additional computations are proportional to the product of N_1 and N_2 and hence, as N_1 and N_2 increase, both types of computation increase in the same proportion.

Table III
Estimated Numerical Values for the Number of Computations as a Function of the Number n of Stationary Intervals in the Unrefined Algorithm when $N_1 = N_2 = 10$ and $K = I = 50$

Computation	Number of Stationary Intervals $n = 2$	$n = 5$	$n = 10$
$\tilde{w}_n(\xi, s)$	2.0×10^6	4.5×10^6	1.3×10^6
$\tilde{w}_j(\xi, s), j \neq n$	8.0×10^8	5.8×10^{17}	1.3×10^{34}
$\hat{w}_j(t_j, s), j \neq n$	1.6×10^9	1.2×10^{18}	2.5×10^{34}
iterations for $\hat{g}_j(\xi)$, $1 \leq j \leq n$	4.0×10^{10}	2.9×10^{19}	6.4×10^{35}

The quantity I is typically between a few tens and a few hundreds. Therefore, we anticipate that *the most time-consuming parts of the algorithm are the iterations for $\hat{g}_j(\xi)$.*

Table III depicts typical numerical values for the number of computations with the unrefined algorithm as a function of the number n of intervals when $N_1 = N_2 = 10$, $K = I = 50$ and the values of l are taken from Table I. Our experience indicates that this is a representative case. From Table III, it is evident that the computations are already pretty high for $n = 2$ and are practically out of the question for $n \geq 5$.

5. SPEEDING UP THE COMPUTATION

We speed up the computation by making two basic observations. First, not all computations of $\hat{w}_j(t_j, s)$ and $\hat{g}_j(s)$ are at distinct values of s. So, if we identify all the distinct values, compute $\hat{w}_j(t_j, s)$ and $\hat{g}_j(s)$ only at those values, store these computations and use these stored values as needed, then a substantial saving in overall computation results. Furthermore, storage efficiency may also be improved by discarding quantities whenever they are no longer needed.

Second, busy periods and workloads are real valued, so that the following relations hold:

$$\hat{g}_j(\bar{s}) = \overline{\hat{g}_j(s)} \quad \text{and} \quad \hat{w}_j(t_j, \bar{s}) = \overline{\hat{w}_j(t_j, s)}, \tag{23}$$

where $\bar{s}$ represents the complex conjugate of s. Therefore, wherever $\hat{g}_j(s)$ and $\hat{w}_j(t_j, s)$ are needed at complex conjugate values of s, only one of them need be computed and stored. We remark that these ways to streamline the algorithm are similar to the efficiencies gained through the fast Fourier transform even though the actual algorithms are quite different; see Abate and Whitt (1992a, p. 21) and Rabiner and Gold (1975, p. 51 and 357).

We now describe the computational requirements with these changes. Our conclusions are summarized in Table IV. Suppose, as in Section 4, that we intend to compute $W_n(t, x)$ for $t = \tau_1, \tau_2, \ldots, \tau_{N_1}$, and $x = x_1, x_2, \ldots, x_{N_2}$; i.e., N_1N_2 values in all. Then $\tilde{w}_n(\xi, s)$ is needed at

$$\xi = \frac{A}{2\tau_p l} - \frac{i\pi j}{\tau_p l}, \quad s = \frac{A}{2x_m l} - \frac{i\pi k}{x_m l}, \tag{24}$$

for $p = 1, 2, \ldots N_1$; $j = 0, 1, \ldots, Kl - 1$; $m = 1, 2, \ldots, N_2$; and $k = 0, \pm 1, \pm 2, \ldots, \pm(Kl - 1)$. As before, these are about $N_1 \cdot N_2 \cdot 2K^2l^2$ distinct values. Next, $\hat{g}_n(\xi)$ is needed at ξ in (24) for $p = 1, 2, \ldots, N_1$; and $j = 0, 1, \ldots, (Kl - 1)$. These are $N_1 \cdot Kl$ distinct values requiring N_1KlI iterations. Also, $\hat{w}_{n-1}(t_{n-1}, s)$ is needed at s in (24) for $m = 1, 2, \ldots, N_2$; $k = 0, 1, \ldots, (Kl - 1)$; and at $s = \xi + \lambda_n - \lambda_n\hat{g}_n(\xi)$ for ξ in (24) with $p = 1, 2, \ldots, N_1$; $j = 0, 1, \ldots, (Kl - 1)$. Hence, there are $(N_1 + N_2)Kl$ distinct values.

It can be shown that for $j = n - 1, n - 2, \ldots, 1$, $\tilde{w}_j(\xi, s)$ is needed at

$$\xi = \frac{A}{2t_j l} - \frac{i\pi k}{t_j l}, \quad s = \frac{A}{2x_m l} - \frac{i\pi k_1}{x_m l}, \tag{25}$$

for $k = 0, \pm 1, \ldots, \pm(Kl - 1)$; $m = 1, 2, \ldots N_2$; and $k_1 = 0, 1, \ldots, (Kl - 1)$. In addition, $\tilde{w}_j(\xi, s)$ is needed at the same ξ as in (25) and at

$$s = \xi_q + \lambda_{n-q} - \lambda_{n-q}\hat{g}_{n-q}(\xi_q), \tag{26}$$

for $q = 0, 1, \ldots, (n - 1 - j)$, where

$$\xi_0 = \frac{A}{2\tau_p l} - \frac{i\pi j}{\tau_p l}, \tag{27}$$

for $p = 1, 2, \ldots, N_1$; $j = 0, 1, \ldots, (Kl - 1)$ and, for $q > 0$,

$$\xi_q = \frac{A}{2t_{n-q} l} - \frac{i\pi k_q}{t_{n-q} l}, \tag{28}$$

Table IV
Estimated Computational Requirements for the More Efficient Implementation of the Inversion Algorithm

Computation	Estimated Number
$\tilde{w}_n(\xi, s)$	$(N_1N_2)2K^2l^2$
$\tilde{w}_j(\xi, s), j \neq n$	$\left[(N_1 + N_2)(n - 1) + \frac{(n-2)(n-1)}{2}\right]2K^2l^2$
$\hat{w}_j(t_j, s), j \neq n$	$\left[(N_1 + N_2)n + \frac{(n-1)n}{2}\right]Kl$
$\hat{g}_j(\xi)$, $1 \leq j \leq n$	$(N_1 + n - 1)Kl$
iterations for $\hat{g}_j(\xi)$, $1 \leq j \leq n$	$(N_1 + n - 1)KlI$
total storage requirement	$(3N_1 + 2N_2 + 3(n - 1))Kl$

for $k_q = 0, 1, \ldots, (Kl - 1)$. These are about $2Kl \cdot (N_2 \cdot Kl + N_1 \cdot Kl + (n - 1 - j) \cdot Kl)$ distinct values.

Also, for $j = n - 1, n - 2, \ldots, 1, \hat{g}_j(\xi)$ is needed at ξ in (25) for $k = 0, 1, \ldots, (Kl - 1)$. These are Kl distinct values corresponding to KlI distinct iterations. Finally, for $j = n - 2, n - 3, \ldots, 1, 0, \hat{w}_j(t_j, s)$ is needed at s in (25) for $m = 1, 2, \ldots, N_2$; $k_1 = 0, 1, \ldots, (Kl - 1)$ and at s in (26) where $q = 0, 1, \ldots, (n - 1 - j)$, ξ_0 in (27) for $p_1 = 1, 2, \ldots, N_1$; $j = 0, 1, \ldots, (Kl - 1)$; and, for $q > 0$, ξ_q in (28) for $k_l = 0, 1, \ldots, (Kl - 1)$. These are $(N_1 + N_2 + n - j)Kl$ distinct values.

The reduced computation is accompanied by the requirement of some additional storage. We need to store only the quantities $\hat{g}_j(\xi)$ and $\hat{w}_j(t_j, s)$. The total storage for $\hat{g}_j(\xi)$ for $j = 1, 2, \ldots n$ is about $[N_1 + (n - 1)Kl$. The storage requirement for $\hat{w}_j(t_j, s)$ is $(N_1 + N_2 + n - 1 - j)Kl$. Since $\hat{w}_{j-1}(t_{j-1}, s)$ may be discarded after computing $\hat{w}_j(t_j, s)$, at any time we need to store $\hat{w}_j(t_j, s)$ for only two successive values of j. This gives a maximum storage of about $2(N_1 + N_2 + (n - 1)Kl$. So the total storage for $\hat{g}_j(\xi)$ and $\hat{w}_j(t_j, s)$ is $[3(N_1 + n - 1) + 2N_2]Kl$. The computation and storage requirements are summarized in Table IV.

In summary, the first row of Table IV represents the basic computation requirement for two-dimensional transform inversion, which is no different from the unrefined algorithm in the first row of Table II. From Table IV, see that some of the additional computations in rows 2–5 grow linearly and others grow quadratically with n. This is to be contrasted with the exponential growth in Table II.

Note that basic computations are proportional to the product of N_1 and N_2, whereas the additional computations are proportional to the sum $(N_1 + N_2)$ (rows 2 and 3) or just N_1 (row 4). Therefore, as N_1 and N_2 increase, the ratio of additional computations to basic computations decreases.

It is significant that the number of iterations needed for the busy-period transforms $g_j(\xi)$ is drastically reduced compared to Table II. Therefore, unless N_2 is small and I is too large, the computational requirement for $g_j(\xi)$ remains small compared to the overall computation requirement. Finally, the total storage requirement given in Table IV is not much and increases linearly with n.

We conclude this section by giving the computational complexity of the refined algorithm for the numerical example in Table III. The new numerical values are given in Table V. These numbers clearly indicate that, with the more efficient algorithm described in this section, the amount of computation and storage remains manageable even with $n = 10$.

Table V
Estimated Numerical Values for the Number of Computations and the Storage Requirement as a Function of the Number n of Stationary Intervals in the Refined Algorithm when $N_1 = N_2 = 10$ and $K = I = 50$

	Number of Stationary Intervals		
Computation	$n = 2$	$n = 5$	$n = 10$
$\tilde{w}_n(\xi, s)$	2.0×10^6	4.5×10^6	1.3×10^6
$\tilde{w}_j(\xi, s), j \neq n$	4.0×10^5	3.9×10^6	2.8×10^7
$\hat{w}_j(t_j, s), j \neq n$	4.2×10^3	1.7×10^4	6.0×10^4
iterations for $\hat{g}_j(\xi)$, $1 \leqslant j \leqslant n$	5.6×10^4	1.1×10^5	2.4×10^5
total storage requirement	5300	9300	19,250

6. THE ALGORITHM

In this section we provide a step-by-step description of how to implement the algorithm. For simplicity we assume all the inversion parameters are the same, but it is easy to make them different.

Step 1. Identify the input parameters; i.e., determine the number of intervals n; the number of time values in the last interval at which the workload distribution is needed ($t = \tau_1, \tau_2, \ldots \tau_{N_1}$); the desired workload values $x_1, x_2, \ldots x_{N_2}$; the arrival rate λ_j, and service-time transform $\hat{h}_j(s)$ in each interval; and the Laplace-Stieltjes transform of the initial workload CDF, $\hat{w}_0(t_0, s)$.

Step 2. Identify the parameters $n_1, m_1, K, A_1, A_2, A, l_1, l_2, l$ of the inversion algorithm, as in Section 3.

Step 3. Compute and store $\hat{g}_j(\xi)$ for $j = 1, 2, \ldots n$ using the busy-period functional equation recursion (9) at all required distinct values of ξ. For $j = n$, the required ξ values are as in (24) with $p = 1, 2, \ldots N_1$ and $j = 0, 1, \ldots, (Kl - 1)$. For $j < n$, the required ξ values are as in (25) with $k = 0, 1, \ldots (Kl - 1)$.

Step 4. Compute $\hat{w}_j(t_j, s)$ at all required distinct values of s starting with $j = 0$ and proceeding as $j = 0, 1, \ldots, n - 1$, using formulas (10) and (6). The distinct s values are as in (25) for $m = 1, 2, \ldots, N_2$; $k_1 = 0, 1, \ldots (Kl - 1)$ and also as in (26) with $q = 0, 1, \ldots (n - 1 - j)$, ξ_0 in (27) with $p = 1, 2, \ldots N_1$; $j = 0, 1, \ldots (Kl - 1)$; and, for $q > 0$, ξ_q in (28) for $k_q = 0, 1, \ldots (Kl - 1)$. Some of the distinct s values depend on the $\hat{g}_j(\xi)$ values which are already computed and stored. For $j = 0$, $\hat{w}_0(t_0, s)$ is the same as the initial workload transform and is part of input specification. For $j > 0$, $\hat{w}_j(t_j, s)$ is obtained by single transform inversion of $\tilde{w}_j(\xi, s)$ using Equation (16). Computation of the transform value depends on the already stored values of $\hat{g}_j(\xi)$ and $\hat{w}_{j-1}(t_{j-1}, s)$ or their complex conjugate. Once $\hat{w}_j(t_j, s)$ is computed at all required s values, $\hat{w}_{j-1}(t_{j-1}, s)$ may be deleted, freeing up storage space.

Step 5. Compute the complementary CDF of the workload, $W_n^c(t, x)$ at all required values by applying two-dimensional transform inversion formula $\tilde{w}_n^c(\xi, s)$ in (12) and (13). The transform value is obtained using (5), (6), and (7). The computation requires the already stored values of $\hat{g}_n(\xi)$ and $\hat{w}_{n-1}(t_{n-1}, s)$ or their complex conjugates.

7. A DEGENERATE CASE CAUSING NUMERICAL PROBLEMS

In Equation (6) with $n = j$, suppose ξ and s are related as follows:

$$\xi = s - \lambda_j + \lambda_j \hat{h}_j(s). \tag{29}$$

Using (9) it can be shown that the above also implies

$$s = \xi + \lambda_j - \lambda_j \hat{g}_j(\xi). \tag{30}$$

Inserting (29) and (30) into (6), we see that both the numerator and denominator are zero, causing indeterminacy and numerical problems.

In general, it is unlikely for (29) and (30) to hold, but there is an interesting special case in which (29) and (30) hold with certainty. Section 5 shows all the distinct argument pairs for which $\tilde{w}_j(\xi, s)$ needs to be evaluated and from that it becomes clear that (29) and (30) hold if either: (a) $t_{n-q} = t_j$, $\lambda_{n-q} = \lambda_j$, $\hat{h}_{n-q}(\cdot) = \hat{h}_j(\cdot)$ and the inversion parameters A_t and l_t used in intervals $n - q$ and j are the same for some q in the range $1, \ldots, (n - 1 - j)$; or (b) $\tau_p = t_j$, $\lambda_n = \lambda_j$, $\hat{h}_n(\cdot) = \hat{h}_j(\cdot)$ for some p and j and the inversion parameters A_t and l_t used in intervals n and j are the same. Roughly speaking, the degeneracy occurs if two separate intervals become indistinguishable from each other.

While it is important to be aware of the degeneracy condition, it is very easy to avoid it. First, making the inversion parameters A_t or l_t slightly different in different intervals will most likely avoid it; e.g., instead of setting $A = 24$ for all intervals i, set it to $23.5 + (i/n)$ on interval i, where n is the total number of intervals. Second, we can simply check to see if there is degeneracy. All the required ξ and s values are computed in the initial part of the algorithm. At this stage it is possible to check explicitly to see if the condition (29) is satisfied or almost satisfied. Since all computations are done in double precision, (29) has to be satisfied very closely to cause any problem. We need to check only on a small subset of all possible (ξ, s) pairs since the inversion formula only considers values of s and ξ with constant real parts. If the outcome of checking is positive, then one of the inversion parameters can be slightly changed and the computation can be redone. Since this checking can be done in a very small fraction of the overall computation time, there is not any significant impact on the overall computation time.

8. NUMERICAL EXPERIENCE

We have checked the algorithm using several numerical examples with several values of the number of intervals, n, and several variations of arrival rates and service time distributions. One concern we had was to make sure that the workload distributions being computed are correct. A simple sanity check is to compare with the steady-state workload distribution, computed by a separate algorithm, when the last stationary interval is quite long. Our algorithm passed this test. For the steady-state distributions, we applied the algorithms in Abate and Whitt (1992a).

Table VI
A Comparison of Numerical Results for the 7-interval and 21-interval Cases

Time t Since the Origin	Workload x	Tail Probability $P(W(t) > x)$	
		7 intervals	21 intervals
66	1	0.769456620	0.769456602
70	1	0.727309000	0.727309017
66	10	0.47234728205	0.47234728210
70	10	0.42607030889	0.42607030883

The one-dimensional and two-dimensional inversion algorithms also have a built-in accuracy check in that the same computation may be done with many different values of A_t and l_t parameters. If the computations are inaccurate, then it is extremely unlikely that the results will match. We verified that the results did match up to seven-to-ten decimal places. Also, by dividing any piecewise stationary interval into more than one intervals, we can produce a new equivalent model with different inversion calculations but the same workload distributions. We verified accuracy by this method as well.

One set of examples we considered had up to $n = 10$ intervals and $K = 50$. In each interval, the arrival process was Poisson, the service-time distribution was gamma with SCV (squared coefficient of variation) in the range 1/16 to 16, and the server utilization was in the range 0.1 to 0.9. We always obtained (7–10)-digit accuracy using the values of l in Table I.

In the introduction we gave some numerical results for a concrete example with seven stationary intervals. There are eight layers of inversion (the six recursive one-dimensional inversions and one final two-dimensional inversion). We used $\{l_t\} = (2, 3, 3, 3, 4, 4, 4, 4)$ and $\{A_t\} = (22, 24, 24, 24, 26, 26, 26, 26)$ going from outermost to innermost layers to obtain (7–10) digit precision. The runtime was three minutes to compute five points in the last interval for two values of x.

Next, to really stress the algorithm, we also considered a 21-interval example, obtained by dividing each of the seven subintervals into three subintervals. We let the lengths of these subintervals be 2, 3, and 5 in each case to avoid the degeneracy discussed in Section 7. The arrival rate and service-time distribution in each subinterval is the same as before, so that the time-dependent workload distribution is unchanged. We used

$$\{l_t\} = (4,5,5,5,5,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7),$$

and $A_t = 26$ for all i.

In Table VI we display the workload tail probabilities $P(W(t) > x)$ for $x = 1$ and 10 for two time points in the last interval $t = 66$ and $t = 70$. (Here t is the time since the origin.) At $x = 1$, we obtain agreement to seven digits, while at $x = 10$ we obtain agreement to nine and ten digits. The run time was 26 minutes. Notice that three times as many intervals required approximately nine times as much computation time, so that the computation indeed grows approximately as the square of the number of intervals.

Even though we were successful in this pretty large example, we believe that our algorithm should primarily be regarded as being for ten or fewer intervals. Then the algorithm runs in at most a few minutes and the errors are tightly controlled. Many practical situations satisfy this requirement of ten or fewer intervals; e.g., overload controls.

9. COMPARISON WITH SIMULATION

It is appropriate to compare our numerical inversion algorithm to simulation, because simulation is a readily available alternative algorithm for calculating the tail probabilities $P(W(t) > x)$. With simulation, we would perform n independent replications of the workload process over the time interval $[0, t]$ and estimate the tail probability $P(W(t) > x)$ by the proportion of the n sample paths for which $W(t) > x$; i.e., if i indexes the replication and $1\{W_i(t) > x\} = 1$ if $W_i(t) > x$ and 0 otherwise, then our estimate is

$$P(W(t) > x) \approx n^{-1} \sum_{i=1}^{n} 1\{W_i(t) > x\}. \tag{31}$$

In some sense, simulation and the inversion algorithm are not directly comparable, because the required computations depend on different features of the model. The simulation run time tends to be linear in the number of events, and thus in model time t, but is independent of the number of subintervals in a piecewise-constant representation. Indeed, simulation does not require the piecewise-constant structure. In contrast, the inversion algorithm is essentially independent of model time, but quadratic in the number of intervals. Thus, the two algorithms tend to be complementary, with the inversion being superior for a model with only few intervals.

The performance of simulation also depends critically on the size of the probability we want to estimate and the statistical precision we want to achieve. It is easy to analyze the performance of the simulation estimator, because for any fixed t and x the estimator is just the sample mean estimator for the probability $p \equiv P(W(t) > x)$ in n Bernoulli trials. The expected value of this estimator is p and the standard deviation is $\sqrt{p(1-p)/n}$. Since we typically are interested in relatively small tail probabilities, it seems natural to use a criterion of *relative standard error* (RSE), which we define as the ratio of the standard deviation to the mean. Clearly, the relative standard error is

$$\text{RSE} = \sqrt{(1-p)/pn} \approx 1/\sqrt{pn}, \tag{32}$$

with the approximation holding for p suitably small. Hence, if we desire an RSE of ϵ, the required number of replications in the simulation is approximately

$$n = 1/p\epsilon^2. \tag{33}$$

If the desired probability p and the target RSE ϵ are not small and the run time for one replication is not long, then simulation will be effective. For example, if $p = \epsilon = 0.1$, then only $n = 1000$ replications are required, which is often feasible. The 1000 replications can well be competitive with the inversion for a problem that is difficult for the inversion, but for a piecewise-constant model with only a few intervals (e.g., one to three), the inversion will require only seconds and thus tend to dominate even with this low precision requirement. Simulation is clearly prohibitive at the high accuracies of 10^{-10} easily achievable by the inversion algorithm when there are only a few intervals. For instance, for $p = 10^{-4}$ and $\epsilon = 10^{-8}$, $n = 10^{20}$. Practical examples might well have $p = \epsilon = 10^{-2}$, for which $n = 10^6$. As the target probability and RSE get smaller, the inversion algorithm begins to dominate simulation. Moreover, it is often possible to greatly speed up the inversion algorithm when there are many subintervals and we require only moderate accuracy by judiciously introducing approximations, as we now indicate.

10. EXPLOITING APPROXIMATIONS TO SPEED UP THE COMPUTATION

Since the inversion computation becomes difficult when there are many intervals, it is natural to consider ways to effectively reduce the number of intervals by making various approximations. The idea is quite simple: If we want to calculate a tail probability $P(W(t) > x)$ at time t, then we might approximately determine the distribution at some time t_0 with $0 < t_0 < t$ and consider a new problem on the time interval $[t_0, t]$ using the transform of the approximating distribution at time t_0 as the initial distribution. If t_0 is not too close to t, then we should still attain good accuracy at time t based on less accuracy at t_0.

One way to do this is to use the inversion algorithm over $[0, t_0]$ to calculate the cdf $P(W(t_0) \le x)$ for several values of x at time t_0. (A similar procedure might also be used at other time points before time t_0; i.e., we need not use only one internal point.) We use these calculated values $P(W(t_0) \le x_i)$ to determine an approximate distribution and transform at t_0.

Another general way to accomplish this goal is to exploit simulation. As discussed in the last section, simulation can be quite effective in obtaining a rough estimate of the distribution at t_0. We can use the empirical transform, i.e., the transform of the empirical distribution at time t_0. We can then use the inversion algorithm to obtain a more precise estimate of the probability distribution at the later time t.

In addition to the two methods just described, there may occur natural *decoupling points* where we know the distribution at least approximately. Two kinds of decoupling points come to mind: (1) steady-state and (2) light traffic. First, if there is a very long interval for which the model in that interval is stable, then we can approximate the distribution at the end of the interval by the steady-state distribution associated with that interval. Second, if there is a long period of very light traffic, then we can approximate the distribution at the end of that interval by a unit point mass at 0.

REFERENCES

ABATE, J. AND W. WHITT. 1992a. The Fourier-Series Method for Inverting Transforms of Probability Distributions. *Queueing Systems* **10**, 5–88.

ABATE, J. AND W. WHITT. 1992b. Solving Probability Transform Functional Equations for Numerical Inversion. *OR Lett.* **12**, 275–281.

ABATE, J. AND W. WHITT. 1994. Transient Behavior of the *M/G/*1 Workload Process. *Opns. Res.* **42**, 750–764.

ASMUSSEN, S. AND T. ROLSKI. 1994. Risk Theory in a Periodic Environment: the Cramér-Lundberg Approximation and Lundberg's Inequality. *Math. O. R.* **19**, 410–433.

CHOUDHURY, G. L. AND D. M. LUCANTONI. 1996. Numerical Computation of the Moments of a Probability Distribution from its Transforms. *Opns. Res.*, **44**, 368–381.

CHOUDHURY, G. L., D. M. LUCANTONI, AND W. WHITT. 1994a. Multi-Dimensional Transform Inversion with Applications to the Transient *M/G/*1 Queue. *Ann. Appl. Prob.* **4**, 719–740.

CHOUDHURY, G. L., D. M. LUCANTONI, AND W. WHITT. 1994b. Numerical Transform Inversion to Analyze Teletraffic Models. *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks, Proceedings of the 14th Int. Teletraffic Congress.* J. Labetoulle and J. W. Roberts (eds.), Elsevier, Amsterdam, 1b, 1043–1052.

CHOUDHURY, G. L. AND W. WHITT. 1995. Q^2: A New Performance Analysis Tool Exploiting Numerical Transform Inversion. *Proc. IEEE MASCOTS '95*, Durham, NC, January., 411–415.

CHOUDHURY, G. L. AND W. WHITT. 1996. Computing Distributions and Moments in Polling Models by Numerical Transform Inversion. *Performance Evaluation* **25**, 267–292.

CHOUDHURY, G. L. AND W. WHITT. 1997. Non-Probability Approximations for Probability Distributions to Aid Numerical Transform Inversion. in preparation.

DAVIS, J., W. A. MASSEY AND W. WHITT. 1995. Sensitivity to the Service-Time Distribution in the Nonstationary Erlang Loss Model. *Mgmt. Sci.*, **41**, 1107–1116.

DUDA, A. 1986. Diffusion Approximation for the Time-Dependent Queueing Systems. *IEEE J. Sel. Areas Commun.* SAC-4, 905–918.

GREEN, L., P. KOLESAR, AND A. SVORONOS. 1991. Some Effects of Nonstationarity on Multiserver Markovian Queueing Systems. *Opns. Res.* **39**, 502–511.

GROSS, D. AND D. MILLER. 1984. The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes. *Opns. Res.* **32**, 362–379.

KOOPMAN, B. O. 1972. Air Terminal Queues Under Time-Dependent Conditions. *Opns. Res.* **20**, 1089–1114.

KUITENBROUWER, G. J. 1992. *An M/M/1 Queue with Piecewise Constant Intensities and the M(t)/M(t)/1 Queue*. University of Twente, The Netherlands.

LEMOINE, A. J. 1989. Waiting Time and Workload in Queues with Periodic Poisson Input. *J. Appl. Prob.* **26**, 390–397.

LOGOTHETIS, D. 1994. Transient Analysis of Communication Networks. Ph.D. Thesis, Department of Electrical Engineering, Duke University.

LUCANTONI, D. M., G. L. CHOUDHURY, AND W. WHITT. 1994. The Transient *BMAP/G/*1 Queue. *Stochastic Models*, **10**, 145–182.

ONG, K. L. AND M. R. TAAFFE. 1989. Nonstationary Queues with Interrupted Poisson Arrivals and Unreliable/Repairable Servers. *Queueing Systems*, **4**, 27–46.

PALM, C. 1943. Intensity Variations in Telephone Traffic. *Ericsson Technics* **44**, 1–189. (English translation by North-Holland, Amsterdam, 1988).

RABINER, L. R. AND B. GOLD. 1975. *Theory and Application of Digital Signal Processing*. Prentice Hall, Englewood Cliffs, NJ.

ROLSKI, T. 1987. Approximation of Periodic Queues. *Adv. Appl. Prob.* **19**, 691–707.

ROLSKI, T. 1989. Queues with Nonstationary Inputs. *Queueing Systems*, **5**, 113–130.

TAAFFE, M. R. AND K. L. ONG. 1987. Approximating *Ph*(*t*)/*M*(*t*)/*S*/*C* Queueing systems. *Ann. Opns. Res.* **8**, 103–116.

TAKÁCS, L. 1962. *Introduction to the Theory of Queues*. Oxford University Press, New York.

VAN DEN BERG, J. L. AND W. GROENENDIJK. 1991. Transient Analysis of an *M/M/*1 Queue with Regularly Changing Arrival and Service Intensities. In *Teletraffic and Datatraffic in a Period of Change. Proceedings of ITC 13*, A. Jensen and V. B. Iversen (eds), North-Holland, Amsterdam.

ZHANG, J. AND E. J. COYLE. 1991. The Transient Solution of the Time-Dependent *M/M/*1 Queue. *IEEE Trans. Inf. Theory*, **37**, 1690–1696.