

## A numerical approach to cyclic-service queueing models

Authors	Blanc,J.P.C.
Published in	Queueing Systems
Publication Date	1990
Link	<a href="https://research.tilburguniversity.edu/en/publications/35f6ffa3-84dc-49fd-969e-085200dfde29">https://research.tilburguniversity.edu/en/publications/35f6ffa3-84dc-49fd-969e-085200dfde29</a>
Citation	Blanc, J P C 1990, 'A numerical approach to cyclic-service queueing models', Queueing Systems, vol. 6, no. 1, pp. 173-188.
Download Date	2026-04-13 08:30:06
Rights	<p>General rights</p> <p>Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.</p> <ul style="list-style-type: none"> <li>- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.</li> <li>- You may not further distribute the material or use it for any profit-making activity or commercial gain</li> <li>- You may freely distribute the URL identifying the publication in the public portal"</li> </ul> <p>Take down policy</p> <p>If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.</p>

## A NUMERICAL APPROACH TO CYCLIC-SERVICE QUEUEING MODELS

J.P.C. BLANC

*Tilburg University, Faculty of Economics, P.O. Box 90153, 5000 LE Tilburg, The Netherlands*

Received 4 April 1988; revised 11 August 1989

### Abstract

An iterative numerical technique for the evaluation of queue length distributions is applied to multi-queue systems with one server and cyclic service discipline with Bernoulli schedules. The technique is based on power-series expansions of the state probabilities as functions of the load of the system. The convergence of the series is accelerated by applying a modified form of the epsilon algorithm. Attention is paid to economic use of memory space.

**Keywords:** Power-series algorithm, traffic intensity, waiting time, epsilon algorithm, memory space;

### 1. Introduction

Queueing systems with more than one waiting line are very hard to analyse when the joint queue length distribution is not of some kind of product form. In Hooghiemstra et al. [8] and Blanc [1–3] a numerical technique has been developed for the evaluation of performance measures for such multi-queue systems. The technique is based on power-series expansions of the state probabilities as functions of one parameter (the traffic intensity) of the systems. The coefficients of these power series can be recursively calculated for a large class of multi-queue models. The coefficients of the power-series expansions of the moments of the queue length distributions follow directly from those of the state probabilities. In most instances a bilinear transformation ensures convergence of the power series over the whole range of traffic intensities for which the system is stable. We have introduced in Blanc [2,3] extrapolations of the coefficients of the power series in order to accelerate the convergence of the series. One of these extrapolations will be combined with the epsilon algorithm (cf. Brezinski [6], Wynn [13]) in the present paper. The advantages of the present technique are that quantities are calculated iteratively, that it is relatively easy to compute additional terms of the power series in order to increase accuracy, that algorithms for accelerating the convergence of sequences can be applied, and that, once the coefficients of the

power series have been obtained, it requires little effort to compute performance measures for various values of the traffic intensity (compare with numerical techniques based on truncation of the state space and solution of large sets of balance equations). The main drawback is the large amount of memory space necessary to store the coefficients of the power series of the state probabilities. The available memory space mainly limits the size of the models which can be handled. Therefore, attention will be paid to optimize the use of the available memory space.

The power-series algorithm will be applied to a multi-queue model with one server and cyclic service discipline with Bernoulli schedules. This kind of model is often used to study distributed computer systems with a single communication channel and a cyclic access scheme. Several authors have derived general relations or have proposed approximations for the mean waiting times in such systems (cf. Boxma and Groenendijk [4], Boxma and Meister [5], Cooper [7], Kühn [9], Servi [10], Takagi [11], Watson [12]). Our approach provides exact data for moderate-sized systems, which are of interest in itself for studying the interaction between queues, and which may be helpful in finding and validating approximations for large scale systems.

The organisation of the paper is as follows. The multi-queue model with cyclic service will be described in section 2. Section 3 is devoted to the derivation of the scheme for calculating the coefficients of the power series. A modified form of the epsilon algorithm is introduced in section 4. Section 5 contains remarks on the implementation of the power-series algorithm, section 6 some numerical examples for the multi-queue model. Possible extensions of the model and the algorithm will be discussed in section 7. Section 8 contains some technical details of the power-series algorithm.

## **2. The multi-queue model**

The system consists of  $s$  queues. Jobs arrive at queue  $j$  according to a Poisson process with rate  $\lambda_j$ ,  $j = 1, \dots, s$ . The single server inspects the queues in cyclic order, i.e. queue 1, 2, ..., and after queue  $s$  again queue 1, etc. When the server finds queue  $j$  non-empty, he serves the first arrived job in this queue. After completion of the service of a job at queue  $j$  the server starts another service at this queue with probability  $q_j$  when this queue is not empty; otherwise the server switches to the next queue ( $j = 1, \dots, s$ ). The times for switching from one queue to the next will be neglected in the present study. Service times at queue  $j$  are assumed to be identically, exponentially distributed with mean  $1/\mu_j$ ,  $j = 1, \dots, s$ . Each queue may contain an unbounded number of jobs. See section 7 for a discussion on the relaxation of some of these model assumptions. Note that the server visits queue  $j$  according to a Bernoulli schedule with parameter  $q_j$ ; this includes exhaustive service ( $q_j = 1$ ) and one-limited service ( $q_j = 0$ ),  $j = 1, \dots, s$ .

First, the condition for ergodicity of the system will be considered (cf. Kühn [9]). The sum of the arrival processes at the various queues is a Poisson process with rate  $\Lambda := \sum_{j=1}^s \lambda_j$ . The service rate of an arbitrary job is  $\mu_j$  with probability  $\lambda_j/\Lambda$ ,  $j = 1, \dots, s$ . Hence, the load or traffic intensity  $\rho$  of the system is in a natural way defined by

$$\rho := \Lambda \sum_{j=1}^s \frac{1}{\mu_j} \frac{\lambda_j}{\Lambda} = \sum_{j=1}^s \frac{\lambda_j}{\mu_j}, \tag{2.1}$$

and a necessary and sufficient condition for ergodicity of the system is

$$\rho < 1. \tag{2.2}$$

Because the traffic intensity  $\rho$  will be used as a variable in power-series expansions, the arrival rates will be written as

$$a_j \rho = \lambda_j, \quad j = 1, \dots, s. \tag{2.3}$$

It will be assumed that the system is in steady state and hence (2.2) will hold. Let  $N_j$  denote the number of jobs in queue  $j$  (waiting or being served),  $j = 1, \dots, s$ . The supplementary variable  $H$ , indicating the queue to which the server attends, is introduced in order to transform the queue length process into a Markov process (which can be described as a multidimensional quasi birth–death process). Let  $\mathbf{n} = (n_1, \dots, n_s)$  be a vector with non-negative integer entries. Note that when the system is empty (in state  $\mathbf{0}$ ) the value of  $H$  is not determined. Therefore, the probability that the system is empty at load  $\rho$  will be denoted by  $p(\rho; \mathbf{0})$ . For  $\mathbf{n} \neq \mathbf{0}$  the state probabilities are defined as follows: for  $h = 1, \dots, s$ ,  $0 \leq \rho < 1$ ,

$$p(\rho; \mathbf{n}, h) := \Pr\{N_j = n_j, j = 1, \dots, s, H = h; \text{ at load } \rho\}. \tag{2.4}$$

Let  $I\{E\}$  stand for the indicator function of the event  $E$ , and let  $\mathbf{e}_j$  be a vector with zero entries except an entry of one at the  $j$ th position ( $j = 1, \dots, s$ ). When the server attends queue  $h$ , a state  $\mathbf{n}$  with  $n_h = 1$  could only have been entered through an arrival at queue  $h$  if all queues were empty ( $h = 1, \dots, s$ ). Note further that the server may reach queue  $h$  from any queue  $h - j$  ( $j = 1, \dots, s$ ) on condition that all intermediate queues  $h - j + 1, \dots, h - 1$  are empty ( $h = 1, \dots, s$ ; read here and below queue  $i + s$  for queue  $i$  whenever  $i < 1$ ). With this in mind, the balance equations for the state probabilities (2.4) are readily verified to be, for  $h = 1, \dots, s$ ,  $n_h > 0$ ,

$$\begin{aligned} & \left[ \rho \sum_{j=1}^s a_j + \mu_h \right] p(\rho; \mathbf{n}, h) \\ &= a_h \rho p(\rho; \mathbf{0}) I\{\mathbf{n} = \mathbf{e}_h\} \\ &+ \rho \sum_{j=1}^s a_j p(\rho; \mathbf{n} - \mathbf{e}_j, h) I\{n_j > 0; n_j > 1 \text{ if } j = h\} \end{aligned}$$

$$\begin{aligned}
& + [q_h + (1 - q_h)I\{n_i = 0 \forall i, i \neq h\}] \mu_h p(\rho; \mathbf{n} + \mathbf{e}_h, h) \\
& + \sum_{j=1}^{s-1} [1 - q_{h-j} + q_{h-j}I\{n_{h-j} = 0\}] I\{n_i = 0, i = h - j + 1, \dots, h - 1\} \\
& \times \mu_{h-j} p(\rho; \mathbf{n} + \mathbf{e}_{h-j}, h - j); \tag{2.5}
\end{aligned}$$

$$\rho \sum_{j=1}^s a_j p(\rho; \mathbf{0}) = \sum_{j=1}^s \mu_j p(\rho; \mathbf{e}_j, j). \tag{2.6}$$

### 3. The power-series algorithm

The power-series algorithm will be discussed briefly in this section. The reader is referred to Blanc [2] and Hooghiemstra et al. [8] for more details and a motivation of the method. First, introduce the bilinear mapping of the interval  $[0, 1]$  onto itself:

$$\rho = \rho(\theta) = \frac{\theta}{1 + G - G\theta} \left( \theta = \frac{(1 + G)\rho}{1 + G\rho} \right), \quad G \geq 0. \tag{3.1}$$

Then, introduce the following power-series expansions, for  $h = 1, \dots, s$ ,

$$\begin{aligned}
p(\rho(\theta); \mathbf{n}, h) &= \theta^{n_1 + \dots + n_s} \sum_{k=0}^{\infty} \theta^k b(k; \mathbf{n}, h), \quad \mathbf{n} \neq \mathbf{0}, \\
p(\rho(\theta); \mathbf{0}) &= \sum_{k=0}^{\infty} \theta^k b(k; \mathbf{0}). \tag{3.2}
\end{aligned}$$

Replace  $\rho$  by  $\theta$  in the balance equations (2.5) according to (3.1), and substitute the power-series (3.2) into these equations. Equating the coefficients of corresponding powers of  $\theta$  in the resulting equations leads to the following iterative scheme for computing the coefficients of the power-series (3.2): for  $h = 1, \dots, s$ ,  $n_h > 0$ , for  $k = 0, 1, 2, \dots$ ,

$$\begin{aligned}
& (1 + G)\mu_h b(k; \mathbf{n}, h) \\
& = \left[ G\mu_h - \sum_{j=1}^s a_j \right] I\{k > 0\} b(k - 1; \mathbf{n}, h) + a_h b(k; \mathbf{0}) I\{\mathbf{n} = \mathbf{e}_h\} \\
& + \sum_{j=1}^s a_j b(k; \mathbf{n} - \mathbf{e}_j, h) I\{n_j > 0; n_j > 1 \text{ if } j = h\} \\
& + [q_h + (1 - q_h)I\{n_i = 0 \forall i, i \neq h\}] \mu_h [(1 + G)b(k - 1; \mathbf{n} + \mathbf{e}_h, h) \\
& \times I\{k > 0\} - Gb(k - 2; \mathbf{n} + \mathbf{e}_h, h) I\{k > 1\}] \\
& + \sum_{j=1}^{s-1} [1 - q_{h-j} + q_{h-j}I\{n_{h-j} = 0\}]
\end{aligned}$$

$$\begin{aligned} & \times I\{n_i=0, i = h-j+1, \dots, h-1\} \mu_{h-j} \\ & \times [(1+G)b(k-1; \mathbf{n} + \mathbf{e}_{h-j}, h-j)I\{k > 0\} \\ & - Gb(k-2; \mathbf{n} + \mathbf{e}_{h-j}, h-j)I\{k > 1\}]. \end{aligned} \tag{3.3}$$

To determine the coefficients of  $p(\rho(\theta); \mathbf{0})$  the law of total probability is used instead of (2.6) to complete the recursive scheme, because the term with  $b(k; \mathbf{0})$  vanishes in (2.6). Substituting (3.1) and (3.2) into the law of total probability gives:

$$\begin{aligned} b(0; \mathbf{0}) &= 1, \\ b(k; \mathbf{0}) &= - \sum_{0 < n_1 + \dots + n_s \leq k} \dots \sum_{h=1}^s b(k - n_1 - \dots - n_s; \mathbf{n}, h), \quad k = 1, 2, \dots \end{aligned} \tag{3.4}$$

There are several ways to compute the coefficients  $b(k; \mathbf{n}, h)$  recursively from (3.3) and (3.4). One convenient way is the following. Calculate all coefficients  $b(k; \mathbf{n}, h)$  with  $k + n_1 + \dots + n_s = m$  before those with  $k + n_1 + \dots + n_s = m + 1$  ( $m = 0, 1, 2, \dots$ ), and on each hyperplane  $k + n_1 + \dots + n_s = m$ ,  $m$  fixed, calculate all coefficients  $b(k; \mathbf{n}, h)$  with  $k = j$  before those with  $k = j + 1$ ,  $j = 0, 1, \dots, m - 1$  ( $m = 0, 1, \dots$ ). See section 8 for more details. Once the coefficients of the power-series expansions of the state probabilities have been determined, those of the moments of the queue length distribution can be obtained as well. Write

$$E\{N_j^\nu\} = \sum_{k=1}^{\infty} \theta^k f_\nu(k; j), \quad j = 1, \dots, s, \nu = 1, 2, \dots \tag{3.5}$$

It follows readily from (3.5) and (3.2) that for  $j = 1, \dots, s, \nu = 1, 2, \dots, k = 1, 2, \dots$ ,

$$f_\nu(k; j) = \sum_{0 \leq n_1 + \dots + n_s \leq k} \dots \sum_{h=1}^s n_j^\nu b(k - n_1 - \dots - n_s; \mathbf{n}, h). \tag{3.6}$$

It is more convenient for obtaining moments of the queue length distribution to compute first their coefficients via (3.6) and then to use (3.5) than to compute first the state probabilities via (3.2) and then the moments directly from the state probabilities. In the first way, algorithms for accelerating the convergence can be applied to partial sums of the series (3.5); see section 4. Moreover, the second way will be more laborious when one is not interested in the (many!) state probabilities themselves.

This section is concluded with a discussion of the stationary waiting time ( $W_h$ ) distribution of jobs arriving at queue  $h$  ( $h = 1, \dots, s$ ). The number of jobs at queue  $h$  left behind by a job departing from that queue is equal to the number of jobs that arrived at queue  $h$  during the sojourn time of the departing job. Because

arrivals occur according to a Poisson process, this implies (cf. Takagi [11]) for  $h = 1, \dots, s$ ,  $|z| \leq 1$ ,

$$E\{z^{N_h}\} = \frac{1}{1 + (1-z)\lambda_h/\mu_h} E\{e^{-\lambda_h(1-z)W_h}\}. \quad (3.7)$$

The moments of the waiting time distributions can be obtained from the moments of the marginal queue length distributions through relation (3.7). Let  $W$  be the waiting time of an arbitrary job, irrespectively of the queue at which it arrives, in steady state. Then, with (3.7) and (2.1),

$$E\{W\} = \sum_{h=1}^s \frac{\lambda_h}{\Lambda} E\{W_h\} = \left[ E\left\{ \sum_{h=1}^s N_h \right\} - \rho \right] / \Lambda. \quad (3.8)$$

Finally, we note that the expected values of the waiting times for jobs in the various queues of a cyclic service system satisfy the following conservation law (cf. Boxma and Groenendijk [4], Watson [12]):

$$\sum_{h=1}^s \frac{a_h}{\mu_h} E\{W_h\} = \frac{\rho}{1-\rho} \sum_{h=1}^s a_h/\mu_h^2. \quad (3.9)$$

By Little's formula and (2.1), relation (3.9) is equivalent to the following relation for the mean queue lengths:

$$\sum_{h=1}^s \frac{1}{\mu_h} E\{N_h\} = \frac{\rho}{1-\rho} \sum_{h=1}^s a_h/\mu_h^2. \quad (3.10)$$

Related to the conservation law is the fact that for the present models

$$p(\rho; \mathbf{0}) = 1 - \rho, \quad \text{or } p(\rho(\theta); \mathbf{0}) = \frac{(1+G)(1-\theta)}{1+G-G\theta}, \quad (3.11)$$

cf. (3.1), so that definition (3.2) implies (see also (3.4)):

$$b(k; \mathbf{0}) = -\frac{1}{1+G} \left( \frac{G}{1+G} \right)^{k-1}, \quad k = 1, 2, \dots \quad (3.12)$$

Relations (3.10) and (3.12) provide useful checks on the correctness and the accuracy of the computations. In the special case that all mean service times are equal (i.e.  $\mu_h = \mu$ ,  $h = 1, \dots, s$ ) then (3.9) and (3.10) lead, with (3.8) and (2.1), to

$$E\left\{ \sum_{h=1}^s N_h \right\} = \mu E\{W\} = \frac{\rho}{1-\rho}, \quad (3.13)$$

the well-known results for the  $M/M/1$  system. Note that relations (3.8)–(3.13) hold for any set of Bernoulli parameters  $\{q_j, j = 1, \dots, s\}$ .

#### 4. Application of the epsilon algorithm

The epsilon algorithm aims to accelerate the convergence of slowly convergent sequences or to determine a value for divergent sequences (cf. Wynn [13],

Brezinski [6]). The epsilon algorithm consists of the following triangular recursive scheme: for  $m = 0, 1, \dots$ ,  $\kappa = 0, 1, \dots$ ,

$$\epsilon_{\kappa+1}^{(m)} = \epsilon_{\kappa-1}^{(m+1)} + [\epsilon_{\kappa}^{(m+1)} - \epsilon_{\kappa}^{(m)}]^{-1}, \tag{4.1}$$

with initial values, for  $m = 0, 1, \dots$ ,

$$\epsilon_{-1}^{(m)} = 0, \quad \epsilon_0^{(m)} = S_m; \tag{4.2}$$

here  $S_m$ ,  $m = 0, 1, \dots$ , is the partial sum of a series. Only the even sequences  $\{\epsilon_{2\kappa}^{(m)}, m = 0, 1, \dots\}$  will be sequences which may converge faster to a limit than  $\{S_m, m = 0, 1, \dots\}$ ,  $\kappa = 1, 2, \dots$ . The odd sequences  $\{\epsilon_{2\kappa+1}^{(m)}, m = 0, 1, \dots\}$  are just intermediate steps in the calculation scheme,  $\kappa = 0, 1, \dots$ . When  $S_m$  is the partial sum of a power-series, say

$$S_m = S_m(\theta) = \sum_{i=0}^m c_i \theta^i, \quad m = 0, 1, \dots, \tag{4.3}$$

then the epsilon algorithm transforms this sequence of polynomials into sequences of quotients of two polynomials. More precisely,  $\epsilon_{2\kappa}^{(m-2\kappa)}$  will be a quotient of a polynomial of degree  $m - \kappa$  over a polynomial of degree  $\kappa$ , and

$$|S_m - \epsilon_{2\kappa}^{(m-2\kappa)}| = O(\theta^{m+1}), \quad \theta \rightarrow 0, \quad \kappa = 1, 2, \dots, m = 2\kappa, 2\kappa + 1, \dots; \tag{4.4}$$

see Wynn [13]. Because many queueing systems have the property that the  $\nu$ th moments of the queue length distribution are of order  $(1 - \rho)^{-\nu}$  as  $\rho \uparrow 1$ ,  $\nu = 1, 2, \dots$ , we propose to modify the initial values for the epsilon algorithm as follows when this algorithm is applied to accelerate the convergence of power-series for moments, cf. (3.5). Before applying the epsilon algorithm, we first extrapolate the coefficients of the power series to take into account the pole at  $\theta = 1$ . This extrapolation has been introduced in Blanc [1,2]. It means that we take for first order moments

$$\epsilon_0^{(m)} = S_m + c_m \frac{\theta^{m+1}}{1 - \theta}, \quad m = 1, 2, \dots, \tag{4.5}$$

and for second order moments

$$\epsilon_0^{(m)} = S_m + \left[ c_m + \frac{c_m - c_{m-1}}{1 - \theta} \right] \frac{\theta^{m+1}}{1 - \theta}, \quad m = 1, 2, \dots, \tag{4.6}$$

instead of the second relation of (4.2); here  $S_m$  is of the form (4.3) and  $c_m$ ,  $m = 1, 2, \dots$ , stand for coefficients of a series as defined in (3.5). It is our experience that the use of (4.5) and (4.6) instead of (4.2) leads to considerably faster convergence (cf. Blanc [2]), and this property is preserved in higher order sequences  $\{\epsilon_{2\kappa}^{(m)}, m = 1, 2, \dots\}$ ,  $\kappa = 1, 2, \dots$ , produced by the epsilon algorithm. For instance, when relation (4.5) is used as an initial sequence, then

$$\epsilon_2^{(m-2)} = S_m + c_m \frac{\theta^{m+1}}{1 - \theta} + \frac{\sigma \theta^{m+1} (c_m - c_{m-1})}{(1 - \theta)(1 - \sigma \theta)}, \tag{4.7}$$

with

$$\sigma = \frac{c_m - c_{m-1}}{c_{m-1} - c_{m-2}}, \quad m = 2, 3, \dots \quad (4.8)$$

For comparison, if relation (4.2) were used as an initial sequence, then

$$\epsilon_2^{(m-2)} = S_m + \frac{\tilde{\sigma}\theta^{m+1}c_m}{(1 - \tilde{\sigma}\theta)}, \quad \tilde{\sigma} = \frac{c_m}{c_{m-1}}, \quad m = 2, 3, \dots \quad (4.9)$$

It will be clear that (4.7) provides a better approximation of  $S_\infty$  than (4.9) when  $S_\infty$  indeed possesses a pole at  $\theta = 1$ . We notice that  $\epsilon_2^{(m-2)}$  as given in (4.7) is identical to the approximation proposed in Blanc [3, formula 4.19]. From the theory of the epsilon algorithm (cf. Brezinski [6], Wynn [13]) it follows that if  $S_\infty$  is a rational function of  $\theta$  with, as denominator, a polynomial of degree  $r + \nu$ ,  $r = 0, 1, \dots$ , which contains a factor  $(1 - \theta)^\nu$ , then

$$\epsilon_{2r}^{(m)} = S_\infty, \quad \text{for } m \geq m_0, \quad (4.10)$$

when (4.5) or (4.6) is used as an initial value for  $\nu = 1$ , or  $\nu = 2$  respectively; the constant  $m_0$  depends on the degrees of the numerator and of the denominator of  $S_\infty$ . This result holds for  $\theta$  smaller as well as larger than the radius of convergence of the series  $S_\infty(\theta)$ , cf. (4.3). Therefore, if the moments of the queue length distribution are rational functions of  $\rho$ , it would not be necessary to use the transformation (3.1). However, experience learns that it is still advisable to use the mapping (3.1) in such a case, because the convergence of the series may be slower and the power-series algorithm may be numerically instable when  $G$  is too small. The latter seems to occur when some state probabilities possess more singularities than the moments, as functions of  $\rho$ . To obtain a good value of  $G$  a test run of the power-series algorithm with  $G = 0$  is needed in order to estimate the radius of convergence of the various power series.

The performance of the modified epsilon algorithm (cf. (4.5), (4.1)) is illustrated in table 1 on the basis of an asymmetrical two-queue system with alternating service discipline (i.e.  $q_1 = q_2 = 0$ ). The arrival rates are  $\lambda_1 = 0.64$ ,  $\lambda_2 = 0.32$ , and the service rates are  $\mu_1 = 1$ ,  $\mu_2 = 2$  (hence  $\rho = 0.8$ ). We have chosen  $G = 2$ . The exact values indicated in this table have been obtained by means of calculations with  $m$  up to 250. It should be noted that the rate of convergence of the sequences  $\{\epsilon_{2\kappa}^{(m)}, m = 1, 2, \dots\}$  does not increase monotonously with increasing  $\kappa$  (see the columns with  $2\kappa = 6$  in table 1). This may be caused by pairs of complex conjugate singularities of the mean queue lengths as a function of  $\rho$ . In general, it is quite unpredictable which sequence produced by the epsilon algorithm will converge most rapidly. It may depend on the value of  $G$ . When the model is more symmetrical or when the traffic intensity is lower, the performance of the epsilon algorithm will be better than in the case of table 1 (and vice versa). More research is needed to discover how the power-series

Table 1  
Performance of the modified epsilon algorithm

$m$	$E\{N_1\}$ (exact: 3.27159431)					
	$\epsilon_0^{(m)}$	$\epsilon_2^{(m-2)}$	$\epsilon_4^{(m-4)}$	$\epsilon_6^{(m-6)}$	$\epsilon_8^{(m-8)}$	$\epsilon_{16}^{(m-16)}$
20	3.2545	3.2702	3.2742	3.2712	3.2721	3.271366
24	3.2639	3.2704	3.2726	3.2711	3.2716	3.273994
28	3.2683	3.2707	3.2720	3.2713	3.2715	3.271606
32	3.2702	3.2710	3.2717	3.2714	3.2716	3.271599
36	3.2711	3.2712	3.2716	3.2715	3.1716	3.271596
40	3.2714	3.2714	3.2716	3.2714	3.2716	3.271595
$m$	$E\{N_2\}$ (exact: 0.65681138)					
	$\epsilon_0^{(m)}$	$\epsilon_2^{(m-2)}$	$\epsilon_4^{(m-4)}$	$\epsilon_6^{(m-6)}$	$\epsilon_8^{(m-8)}$	$\epsilon_{16}^{(m-16)}$
20	0.69101	0.65968	0.65162	0.65762	0.65583	0.657269
24	0.67225	0.65924	0.65484	0.65784	0.65682	0.652011
28	0.66345	0.65863	0.65605	0.65749	0.65694	0.656788
32	0.65953	0.65803	0.65653	0.65720	0.65689	0.656801
36	0.65788	0.65753	0.65672	0.65707	0.65685	0.656807
40	0.65722	0.65718	0.65679	0.65721	0.65683	0.656810

algorithm can be combined most effectively with the epsilon algorithm or any other algorithm for accelerating the convergence of sequences (cf. Brezinski [6]).

### 5. Implementation

The main restriction in applying the power-series algorithm is the required amount of memory space. Therefore, this section is devoted to ideas for an efficient implementation of the power-series algorithm. One way to limit the required amount of memory space is the reduction of the number of coefficients  $b(k; \mathbf{n}, h)$  which have to be calculated (cf. (3.2), (3.3), (3.4)) by applying algorithms for accelerating the convergence of sequences such as the epsilon algorithm discussed in section 4. Other ways may be found in preventing that part of the available memory positions remains unused and in reusing the memory positions which are occupied by coefficients  $b(k; \mathbf{n}, h)$  that will no longer be needed in later computations. These topics will be addressed below.

Suppose that the coefficients of the power-series expansions of the state probabilities and the moments of the queue length distribution have to be computed up to the  $M$ th power of  $\theta$  for a particular model. This implies that the coefficients  $b(k; \mathbf{n}, h)$  must be calculated for all  $k$  and  $\mathbf{n}$  with  $k + n_1 + \dots + n_s = m$ ,  $m = 0, 1, \dots, M$  and for  $h = 1, \dots, s$  (cf. (3.2), (3.5), (3.6)), i.e.

$$s \binom{M + s + 1}{s + 1} \tag{5.1}$$

of those coefficients are needed. When these coefficients would be stored in rectangular arrays, then

$$s(M+1)^{s+1} \quad (5.2)$$

memory positions would be required. Hence, there is a considerable reduction in storage requirement when a two-dimensional array of size (5.1) is used to store the coefficients  $b(k; \mathbf{n}, h)$ . In order to be able to locate the coefficients, the following mapping of the lattice points  $(k, \mathbf{n})$ ,  $k + n_1 + \dots + n_s \leq M$ , onto the set of integers  $0, 1, \dots, \binom{M+s+1}{s+1} - 1$ , can be used (cf. Blanc [2]):

$$C(k; \mathbf{n}) = \sum_{j=0}^s \binom{k+j+\sum_{i=1}^j n_i}{j+1}. \quad (5.3)$$

The drawback of this procedure is that it costs quite some computation time to determine the locations of the  $3s+2$  coefficients which are in general involved in each step of the iteration (3.3) by using (5.3) directly. Therefore, we give a more efficient algorithm for determining the locations of these  $3s+2$  coefficients simultaneously in section 8.

A further reduction of the storage requirement can be obtained by the following considerations. In many circumstances one is not interested in all the individual state probabilities (3.2), but only in some aggregated measures of performance such as the first and second order moments of the queue length distribution and a few characteristic probabilities. In this case it will be more efficient to store the coefficients of the power-series expansions of this limited number of performance measures in separate arrays. The coefficients  $b(k; \mathbf{n}, h)$  can then be deleted from memory as soon as they are no longer needed in later steps of the iteration (3.3), and we can use the following mapping to locate these coefficients:

$$C_M(k; \mathbf{n}) = C(k; \mathbf{n}) \bmod D_M. \quad (5.4)$$

Here  $D_M$  is the maximal distance which occurs between the value  $C(k; \mathbf{n})$  and any of the values  $C(k-1; \mathbf{n})$ ,  $C(k; \mathbf{n} + \mathbf{e}_h)$ ,  $C(k-1; \mathbf{n} + \mathbf{e}_h)$ ,  $C(k-2; \mathbf{n} + \mathbf{e}_h)$ ,  $h = 1, \dots, s$  (cf. (3.3)), over all points  $(k; \mathbf{n})$  with  $k + n_1 + \dots + n_s \leq M$ . It is readily verified that this implies (see also section 8) that

$$D_M = \max \{ C(k; \mathbf{n}) - C(k-2; \mathbf{n} + \mathbf{e}_s); k + n_1 + \dots + n_s \leq M \}, \quad (5.5)$$

if the coefficients  $b(k; \mathbf{n}, h)$ ,  $h = 1, \dots, s$ , are computed in order of increasing value of  $C(k; \mathbf{n})$  (cf. (5.3)). It turns out that the maximum in (5.5) is attained at the point  $(M; \mathbf{0})$ , so that

$$\begin{aligned} D_M &= \binom{M+s+1}{s+1} - 2 \binom{M+s-1}{s+1} + \binom{M+s-2}{s+1} \\ &= \binom{M+s}{s} + \binom{M+s-2}{s-1}. \end{aligned} \quad (5.6)$$

Table 2  
Maximum number of terms  $M$  at a storage capacity of  $10^6$  coefficients

# queues	2	3	4	5	6
Rectangular	78	23	11	6	4
Triangular (5.3)	142	50	28	19	14
With modulus (5.4)	997	123	46	26	18

This approach requires  $sD_M$  memory positions to store the coefficients  $b(k; \mathbf{n}, h)$ . Beside these coefficients also those of the aggregated performance measures have to be stored. However, in order to apply the epsilon algorithm to the coefficients of these measures, they must be determined also when the modulus operator in (5.4) would not be used. To illustrate the gain which is obtained by applying (5.4) and (5.6), we show in table 2 the maximum number  $M$  of terms of the power series (3.5) which can be computed when respectively rectangular arrays (cf. (5.2)), the mapping (5.3) (cf. (5.1)), or the mapping (5.4) are used and when there is storage capacity for  $10^6$  coefficients  $b(k; \mathbf{n}, h)$ .

## 6. Examples

In this section numerical data for cyclic-service systems which have been obtained with the aid of the power-series algorithm will be presented. The value of  $G$  in the mapping (3.1), the number of terms  $M$  of the power-series (cf. (3.5)), and the number of steps  $\kappa$  in the epsilon algorithm (cf. (4.1)), which were needed to obtain these data, depend on various properties of the models. Generally, these quantities increase with increasing traffic intensity, with increasing number of queues, with increasing asymmetry between the parameters of the various queues, and with decreasing value of the Bernoulli parameters  $q_j$ ,  $j = 1, \dots, s$ .

Table 3 shows the way in which the expected values and standard deviations of the waiting times depend on the values of the Bernoulli parameters  $q_1$  and  $q_2$ , for a two-queue system with  $\mu_1 = \mu_2 = 1$  and  $\lambda_1 = \lambda_2 = 0.45$  (i.e.  $\rho = 0.9$ ). For com-

Table 3  
Dependency of the waiting time distributions on the parameters of the Bernoulli schedules ( $\rho = 0.9$ )

$q_1$	$q_2$	$E\{W_1\}$	$E\{W_2\}$	$\sigma\{W_1\}$	$\sigma\{W_2\}$
0.0	0.0	9.000	9.000	11.187	11.187
0.5	0.5	9.000	9.000	11.547	11.547
1.0	1.0	9.000	9.000	12.362	12.362
0.0	0.5	14.697	3.303	18.220	3.851
0.5	1.0	15.694	2.306	19.145	2.378
0.0	1.0	16.364	1.636	19.390	1.809

Table 4

Standard deviations of the waiting times for symmetrical systems ( $\mu_j = 1, j = 1, \dots, s$ )

$\rho$	$s = 2$		$s = 3$		$s = 4$	
	exh.	1 - lim.	exh.	1 - lim.	exh.	1 - lim.
0.10	0.491	0.490	0.493	0.492	0.494	0.493
0.30	1.071	1.056	1.081	1.070	1.086	1.078
0.50	1.901	1.835	1.925	1.881	1.935	1.906
0.70	3.693	3.460	3.730	3.598	3.743	3.680
0.80	5.876	5.414	5.919	5.680	5.933	5.846
0.90	12.36	11.19	12.41	11.87	12.42	12.31
0.95	25.29	22.67	25.33	24.20	25.35	25.21

parison, the standard deviation of the waiting time in an  $M/M/1$  system with  $\rho = 0.9$ ,  $\mu = 1$  and FIFO service discipline is 9.950.

In table 4 the standard deviation of the waiting time distribution has been listed for symmetrical systems with either exhaustive service ( $q_j = 1, j = 1, \dots, s$ ) or 1-limited service ( $q_j = 0, j = 1, \dots, s$ ). The mean waiting times follow directly from (3.13) for symmetrical systems, and do not depend on the Bernoulli schedule.

Table 5 shows the influence of a relatively heavily loaded queue on the mean waiting times at queues which are four times more lightly loaded, for various service schedules. The parameters of the system are, in the case of  $s = 3$  (4) queues:  $\mu_1 = 1, \mu_j = 2, j = 2, 3(,4)$ ;  $a_1 = 2a_j, j = 2, 3(,4)$ ;  $q_j = q_2, j = 3(,4)$ ; and  $\rho = 0.8$ . Note that the differences in mean waiting times of the lightly loaded queues are not negligible, although their arrival and service rates are the same. It should be noted that the mean waiting times can be computed more efficiently by solving the set of equations proposed by Cooper [7] in the case of exhaustive service at all queues ( $q_1 = q_2 = 1$ ).

Table 6 is concerned with the order in which the server attends the queues. The system consists of 4 queues A, B, C, D, with parameters  $a_A = a_B = 0.16, a_C = a_D = 0.64, \mu_A = \mu_C = 1, \mu_B = \mu_D = 4, q_j = 0$  for  $j = A, B, C, D$ , and  $\rho = 0.9$ . For

Table 5

The influence of one relatively heavily loaded queue ( $\rho = 0.8$ )

$s = 3$	$q_1$	$q_2$	$E\{W_1\}$	$E\{W_2\}$	$E\{W_3\}$	$E\{W\}$	
	0	0	4.170	1.644	1.677	2.915	
	1	0	1.515	6.936	7.004	4.242	
	1	1	2.453	4.869	5.319	3.773	
$s = 4$	$q_1$	$q_2$	$E\{W_1\}$	$E\{W_2\}$	$E\{W_3\}$	$E\{W_4\}$	$E\{W\}$
	0	0	4.190	1.719	1.745	1.774	2.724
	1	0	1.334	5.499	5.554	5.610	3.866
	1	1	2.344	3.979	4.183	4.460	3.462

Table 6  
The effect of the order in which the server attends the queues

Order	$E\{W_A\}$	$E\{W_B\}$	$E\{W_C\}$	$E\{W_D\}$	$E\{W\}$
A B C D	1.654	1.549	9.547	7.585	7.173
A D C B	1.654	<b>1.485</b>	9.544	7.611	7.176
A C B D	1.670	1.486	<b>9.543</b>	7.597	7.172
A D B C	1.639	1.548	9.547	7.597	7.177
A B D C	<b>1.638</b>	1.531	9.545	7.613	7.181
A C D B	1.671	1.499	9.547	<b>7.583</b>	<b>7.169</b>

each queue the minimum mean waiting time has been printed in bold type in the table. It can be seen that for each queue A, B and D separately it is best to follow after queue C and it is worst to precede queue C, the most heavily loaded queue. This difference is relatively largest at queue B (4.3%).

### 7. Comments

The power-series algorithm has been applied in this paper to a single server, multi-queue system with cyclic service discipline, Bernoulli schedules, infinite buffers, Poisson arrival streams, exponential service time distributions and negligible switching times. It can also be applied to several variations and extensions of this model. Service disciplines like random allocation or priority for the longest queue can be treated in the same way as cyclic service. Disciplines like gated service or  $K$ -limited service (at most  $K$  jobs are served at each visit of the server to a queue with this discipline) pose some problems, because supplementary variables with rather large ranges of values are needed to transform the queue length process into a Markov process. The power-series algorithm can, in principle, also be applied to models with finite buffers. The main difference with infinite buffer systems is that steady state occurs at any traffic intensity  $\rho$ ,  $\rho > 0$ . Therefore, we propose to use the conformal mapping

$$\theta = \frac{\rho}{C + \rho} \left( \rho = \frac{C\theta}{1 - \theta} \right), \quad C \geq 0, \tag{7.1}$$

of  $[0, \infty)$  onto  $[0, 1)$ , instead of the conformal mapping (3.1). Further, the modification of the epsilon algorithm as in (4.5) and (4.6) should not be applied since the moments of the queue length distribution are finite for all values of the traffic intensity when buffers are finite. Application of the power-series algorithm to finite buffer systems is a subject for further research. Note, however, that it may be more efficient to solve the set of balance equations directly when buffer sizes are small. Exponential distributions in the model may be replaced by phase-type distributions (cf. Blanc [3]). This requires the introduction of one

supplementary variable for each non-Poissonian arrival process and one supplementary variable when one or more service time distributions are non-exponential. Non-zero switching times can also be incorporated in the model and the algorithm, but they must have phase-type distributions. Then a two-valued variable should be added indicating whether the server is serving a job or moving from one queue to the next. In this case the variable  $H$  indicating the position of the server is also defined when all queues are empty. The distribution of  $H$  when the system is empty cannot be recursively calculated, because the server may turn an arbitrary number of cycles around the queues when they are all empty. Sets of  $s$  linear equations have to be solved to compute the coefficients of the power-series expansions of the probabilities that the system is empty and the server is moving between two adjacent queues. The coefficients of all other state probabilities can be calculated in a recursive manner.

## 8. Technical details of the algorithm

In section 5 a mapping of the lattice points  $(k, \mathbf{n})$  to the set of integers has been discussed (cf. (5.3)). In this section we give an efficient procedure for determining the values of this mapping in an integrated way for all lattice points which occur at one step of the power-series algorithm (cf. (3.3)). This procedure is based on the following properties of the mapping  $C(k; \mathbf{n})$  that are straightforwardly verifiable:

$$\begin{aligned} C(k; \mathbf{n} - \mathbf{e}_j) &= C(k; \mathbf{n} - \mathbf{e}_{j+1}) - v(j), \\ C(k-1; \mathbf{n} + \mathbf{e}_{j+1}) &= C(k-1; \mathbf{n} + \mathbf{e}_j) - v(j), \end{aligned} \quad (8.1)$$

with

$$v(j) := \binom{k+j-1 + \sum_{i=1}^j n_i}{j}, \quad (8.2)$$

for  $j = 0, 1, \dots, s$ ; here and below, both  $(k; \mathbf{n} - \mathbf{e}_{s+1})$  and  $(k-1; \mathbf{n} + \mathbf{e}_0)$  stand for  $(k; \mathbf{n})$ , while both  $(k; \mathbf{n} - \mathbf{e}_0)$  and  $(k-1; \mathbf{n} + \mathbf{e}_{s+1})$  are equivalent to  $(k-1; \mathbf{n})$ . Further, the iteration (3.3), (3.4) will proceed along points  $(k; \mathbf{n})$  according to increasing values of  $C(k; \mathbf{n})$ . This order will be indicated later (cf. (8.4)). The predecessor of the point  $(k; \mathbf{n})$  in this order is denoted by  $(k^*, \mathbf{n}^*)$ . The procedure to locate the points which are needed in the iteration step (3.3) then reads:

$C(k; \mathbf{n}) = C(k^*; \mathbf{n}^*) + 1;$   
 For  $j := 0$  to  $s$  calculate  $v(j);$   
 For  $j := s$  downto  $1$  do  $C(k; \mathbf{n} - \mathbf{e}_j) := C(k; \mathbf{n} - \mathbf{e}_{j+1}) - v(j);$

If  $k \geq 1$  then for  $j := 0$  to  $s$  do  $C(k - 1; \mathbf{n} + \mathbf{e}_{j+1}) := C(k - 1; \mathbf{n} + \mathbf{e}_j) - v(j)$ ;  
 If  $k \geq 2$  and  $G > 0$  then for  $j := 0$  to  $s - 1$  do

$$C(k - 2; \mathbf{n} + \mathbf{e}_{j+1}) := C(k - 2; \mathbf{n} + \mathbf{e}_j) - v(j) \frac{k - 1 + \sum_{i=1}^j n_i}{k + j - 1 + \sum_{i=1}^j n_i}. \quad (8.3)$$

Finally, we present a procedure for calculating the coefficients of the power-series expansions of the state probabilities and the moments of the queue length distribution according to (3.3), (3.4) and (3.6) up to the  $M$ th power of  $\theta$ , along points with increasing values of  $C(k; \mathbf{n})$ .

```

b(0; 0) := 1;
for  $\nu := 1$  to 2,  $j := 1$  to  $s$ ,  $m := 1$  to  $M$  do  $f_\nu(m; j) := 0$ ;
sum1 := 0;
for  $m := 1$  to  $M$ 
for  $n_s := m$  downto 0
for  $n_{s-1} := m - n_s$  downto 0
:
:
for  $n_1 := m - n_s - \dots - n_2$  downto 0 do
[ $k := m - n_s - \dots - n_1$ ;
if  $k = m$  then {  $b(m; \mathbf{0}) := -\text{sum1}$ ;  $\text{sum1} := 0$  } else
{ determine the memory positions of the points needed in (3.3) with the aid of
(8.3);
sum2 := 0;
for  $h := 1$  to  $s$  do
[calculate  $b(k; \mathbf{n}, h)$  according to (3.3);
sum2 := sum2 +  $b(k; \mathbf{n}, h)$ ];
sum1 := sum1 + sum2;
for  $\nu := 1$  to 2 and  $j := 1$  to  $s$  do
 $f_\nu(m; j) := f_\nu(m; j) + n_j^\nu \times \text{sum2}$  (cf. (3.6))].
    
```

The variable sum1 in (8.4) is used to determine  $b(k; \mathbf{0})$ ,  $k = 1, \dots, M$ , according to (3.4).

**Acknowledgement**

The author thanks Dr. J. Kapenga for a helpful discussion.

**References**

[1] J.P.C. Blanc, A note on waiting times in systems with queues in parallel, *J. Appl. Probab.* 24 (1987) 540–546.

- [2] J.P.C. Blanc, On a numerical method for calculating state probabilities for queueing systems with more than one waiting line, *J. Comput. Appl. Math.* 20 (1987) 119–125.
- [3] J.P.C. Blanc, A numerical study of a coupled processor model, in: *Computer Performance and Reliability*, eds. G. Iazeolla, P.J. Courtois and O.J. Boxma (North-Holland, Amsterdam, 1988) pp. 289–303.
- [4] O.J. Boxma and W.P. Groenendijk, Pseudo conservation laws in cyclic-service systems, *J. Appl. Probab.* 24 (1987) 949–964.
- [5] O.J. Boxma and B.W. Meister, Waiting-time approximations in multi-queue systems with cyclic service, *Perform. Eval.* 7 (1987) 59–70.
- [6] C. Brezinski, *Accélération de la Convergence en Analyse Numérique*, Lecture Notes in Mathematics 584 (Springer, Heidelberg, 1977).
- [7] R.B. Cooper, Queues served in cyclic order: waiting times, *Bell Syst. Techn. J.* 49 (1970) 399–413.
- [8] G. Hooghiemstra, M. Keane and S. van de Ree, Power series for stationary distributions of coupled processor models, *SIAM J. Appl. Math.* 48 (1988) 1159–1166.
- [9] P.J. Kühn, Multi-queue systems with non-exhaustive cyclic service, *Bell Syst. Techn. J.* 58 (1979) 671–698.
- [10] L.D. Servi, Average delay approximation of M/G/1 cyclic service queues with Bernoulli schedules, *IEEE J. Sel. Areas Comm. SAC-4* (1986) 813–822.
- [11] H. Takagi, *Analysis of Polling Systems* (The MIT Press, Cambridge, MA, 1986).
- [12] K.S. Watson, Performance evaluation of cyclic service strategies – a survey, in: *Performance '84*, ed. E. Gelenbe (North-Holland, Amsterdam, 1985) pp. 521–533.
- [13] P. Wynn, On the convergence and stability of the epsilon algorithm, *SIAM J. Numer. Anal.* 3 (1966) 91–122.