APPROXIMATIONS TO THE WAITING TIME PERCENTILES IN THE M/G/c QUEUE

L.P. SEELEN and H.C. TIJMS

Dept. of Actuarial Sciences and Econometrics, Vrije Universiteit
Amsterdam, The Netherlands

1. INTRODUCTION AND MAIN RESULT

In the last decade several useful approximations have been obtained for the average waiting time in the M/G/c queue, see [9] and [10] for discussion and further references. An appealing two-moment approximation to \( E(W) \) is given by

\[
E_{\text{app}}(W) = (1-c_s^2) E_{\text{det}}(W) + c_s^2 E_{\exp}(W),
\]

where \( c_s \) denotes the coefficient of variation (=ratio of standard deviation and mean) of the service time and \( E_{\text{det}}(W) \) and \( E_{\exp}(W) \) represent the average waiting time for the respective cases of deterministic service and exponential service with the same means \( E(S) \). This approximation agrees with the Pollaczek-Khintchine formula for the special case of \( c=1 \) server. In [2] it was already noticed that the Pollaczek-Khintchine formula can be written in the form (1) involving a linear interpolation on the squared coefficient of variation of the service time and it was pointed out that such a linear interpolation might yield useful approximations to more complex queueing models. The approximation (1) for the M/G/c queue was investigated in [1] and [5], where the reference [1] also presents a simple but accurate approximation to \( E_{\text{det}}(W) \) (see also [9] or [10] for a correction to a misprint in the latter result). The two-moment approximation (1) shows an excellent performance provided \( c_s \) is not too large (say, \( 0 < c_s < 2 \)); for larger values of \( c_s \) it is no longer true that measures of system performance are fairly insensitive to more than the first two moments of service time.

A natural question is whether an approximation of the form (1) also applies to the waiting time percentiles of the M/G/c queue. Letting \( W \) be the waiting time of a customer (excluding service time) when the system is in statistical equilibrium, the \( p \)-th waiting time percentile \( \xi(p) \) is defined by

\[
P(W > \xi(p)) = p, \quad 0 < p < 1.
\]

denoting the delay probability \( P(W > 0) \).

Also, let \( \xi_{\text{det}}(p) \) and \( \xi_{\exp}(p) \) denote the waiting time percentile \( \xi(p) \) for the particular cases of the M/D/c queue (deterministic service) and the M/M/c queue (exponential service) with the same average service times \( E(S) \). Then, the two-moment approximation

\[
\xi_{\text{app}}(p) = (1-c_s^2) \xi_{\text{det}}(p) + c_s^2 \xi_{\exp}(p)
\]

performs quite well for all values of \( p \) provided \( c_s^2 \) is not too large (say, \( 0 < c_s^2 < 2 \)). A similar statement applies to the conditional waiting time percentiles \( \eta(p) \) defined by

\[
P(W > \eta(p) | W > 0) = p, \quad 0 < p < 1.
\]

Note that, by the approximation for the waiting time percentiles, we can indirectly approximate the waiting time probabilities.

The two-moment approximation (2) is of practical value only when it is easy to compute the particular waiting time percentiles \( \xi_{\text{det}}(p) \) and \( \xi_{\exp}(p) \). The percentiles \( \xi_{\exp}(p) \) and \( \eta_{\exp}(p) \) are trivial to calculate, since for the M/M/c queue with arrival rate \( \lambda \) and service rate \( \mu=1/E(S) \) we have the well-known explicit result

\[
P(W > x) = \Pi_{\text{exp}} e^{-(\mu-\lambda)x}, \quad x>0,
\]

where \( \Pi_{\text{exp}} \) is Erlang's delay probability for which a simple explicit expression is available. In the next section we show that the waiting time percentiles \( \xi_{\text{det}}(p) \) and \( \eta_{\text{det}}(p) \) can rather easily be calculated by using a tailor-made numerical method for the M/D/c queue.

2. SPECIAL-PURPOSE ALGORITHM FOR THE M/D/c QUEUE

In this section we discuss a special-purpose algorithm for calculating the state probabilities and the waiting time probabilities in the M/D/c queue with Poisson arrivals at rate \( \lambda \) and a deterministic service time \( D \) such that the server utilization \( \rho=\lambda D/c \) is smaller than 1. The
algorithm is extremely efficient and easy to program, and requires very small computing times even for a very large number of servers and high traffic. The method uses a number of fundamental results that go essentially back to Crommelin's paper [3]. The algorithm calculates first the steady-state probabilities and calculates next the waiting time probabilities by using the values of the state probabilities. A special paper [3]. The algorithm calculates first the steady-state probabilities and calculates next the waiting time probabilities by using the values of the state probabilities. A special iterative method is applied to calculate the state probabilities from the equilibrium equations derived in [3]. This iterative method will be described later. Suppose for the moment that the state probabilities have been computed. In the original paper [3] of Crommelin two closed-form expressions are given for the waiting time probability \( P(W_{q}x) \). The first of these two expressions is a finite sum involving terms that alternate in sign and the second one is an infinite sum involving positive terms only. Unfortunately, as already realized by Crommelin, both representations offer numerical difficulties when the traffic is non-light. Incidentally, for multi-server queues the server utilization \( \rho \) is in general not a suitable measure for the traffic load on the system and for that purpose one should use the delay probability \( \Pi_{w} \) rather than \( \rho \). For non-light traffic, the numerical evaluation of the above mentioned sum with terms alternating in sign will be hampered by roundoff errors due to loss of significance, while for the other closed-form representation the numerical problem is the slow convergence of the infinite sum where the calculations may be halted by the occurrences of underflow and overflow before convergence is achieved. Fortunately, we can provide a practically useful alternative for the calculation of the waiting time probabilities (percentiles). A computationally more useful representation of the waiting time probabilities is actually contained in Crommelin's paper [3], but was apparently overlooked in the aim at a closed-form solution. By an ingenious probabilistic argument, he obtained the following result. Letting \( \Pi_{j} \) be the steady-state probability of having \( j \) customers in the system and using the representation

\[
x = mD + u \quad \text{for some integer } m \geq 0 \text{ and } 0 \leq u < D,
\]

the waiting time probability \( P(W_{q}x) \) may be expressed as

\[
P(W_{q}x) = b_{mc+c-1}(u),
\]

where the \( b_{j}(u) \)'s satisfy the equation

\[
\delta
\]

allowing for a recursive computation of the \( b_{j}(u) \)'s starting with \( b_{0}(u) = e^{-\lambda u} \). Although this recursion scheme also involves the taking of differences, it offers considerably less numerical difficulties than the closed-form expressions discussed earlier. Actually, a computationally better form of the recursion relation (4) is obtained by rewriting it as

\[
\begin{align*}
\delta
\end{align*}
\]

Thus the desired waiting time probability \( P(W_{q}x) \) may be calculated by applying the recursion scheme (5) until \( b_{mc+c-1}(u) \) is obtained. This recursion scheme however should not be applied blindly, since it will also ultimately be hampered by roundoff errors for large values of \( \lambda u \) when the traffic is non-light. The recursion scheme (5) should be used in an appropriate combination with the asymptotic expansion

\[
P(W_{q}x) \approx ae^{-\delta x} \quad \text{for } x \text{ large.}
\]

Extensive numerical experiments show that in practical applications the asymptotic expansion (6) is already very accurate long before the recursion scheme (5) offers numerical difficulties. Our empirical finding is that for practical purposes the asymptotic expansion (6) may be used for

\[
x \geq D/c
\]

provided the delay probability \( \Pi_{j} \) is not too small (say, \( \Pi_{j} \leq 0.2 \)). Also, we found that for any \( \rho > 0.9 \) the \( p \)-th conditional waiting time percentile \( \eta(p) \) may be calculated by using (6) provided the traffic is not too light. The coefficients \( \alpha \) and \( \delta \) of the asymptotic expansion (6) are easily computed. It is well-known that the constant \( \delta \) is the unique positive solution to the equation (cf. [3] and [8])

\[
\lambda(e^{\delta D/c-1}) = \delta.
\]

Letting

\[
\tau = 1 + \delta / \lambda,
\]

it follows from results in [8] and [9] that

\[
\alpha = \frac{\eta \delta}{\lambda(\tau - 1)^{2} c - 1},
\]

where \( \eta \), being defined as \( \lim_{j \to \infty} \frac{1}{j} p_{j} \), is given by

\[
\eta = \left( \frac{D}{\tau} - AD \right) c - 1 \sum_{i=0}^{c-1} \frac{1}{\tau^{i-1}} c - 1.
\]

Notice that the amplitude factor \( \alpha \) needs only the first \( c \) state probabilities.
To the end of this section, we discuss a special-purpose iterative method for the calculation of the state probabilities \( p_j \) from the equilibrium equations

\[
p_j = e^{-\lambda D AD} j \sum_{k=0}^T p_k + \sum_{k=c+1}^{T-1} e^{-\lambda D AD} (j-k+c)! p_k
\]

for \( j=0,1,\ldots \) (11)

together with the normalizing equation

\[
\sum_{j=0}^N p_j = 1.
\]

A common approach for solving this infinite system of linear equations is to truncate first the system by a sufficiently large chosen integer \( L \) such that \( \sum_{j=1}^L p_j \leq 10^{-6} \) (say), where \( L \) is found by using explicit results for the state probabilities in the \( M/M/c \) queue, and to solve next the truncated system of linear equations by the standard successive overrelaxation method, cf. [4]. This computational approach may be considerably improved in two respects. Firstly, a successive overrelaxation method with a dynamically adjusted relaxation factor may be used in order to avoid the difficulty of not knowing on beforehand the optimal value of the relaxation factor. Secondly, in view of the theoretical result

\[
\frac{p_{j+1}}{p_j} \approx \tau \quad \text{for all } j \text{ sufficiently large}
\]

(13)

with \( \tau \) given by (8), the infinite system of linear equations (11)-(12) may be reduced to a finite system of linear equations by using the asymptotic estimate

\[
p_j \approx \tau^{N-j} p_N \quad \text{for } j \geq N
\]

(14)

when \( N \) is chosen sufficiently large. The asymptotic expansion (13) applies usually already for relatively small values of \( j \), in particular when the traffic load on the system increases. Thus, for non-light traffic, an integer \( N \) such that (14) is sufficiently accurate will typically be much smaller than the truncation integer \( L \) discussed earlier. On the contrary for very light traffic situations the asymptotic expansion (14) may apply not before the state probabilities are negligibly small, but the algorithm below is designed in such a way that in those situations it operates automatically as if a truncation integer \( L \) as above would be used. A good choice of \( N \) is usually not known on beforehand and an initial guess with \( N \) very large would be inefficient. In the algorithm the problem of determining an appropriate value of \( N \) is solved by using an adaptive scheme starting with a "low" estimate of \( N \) and increasing this estimate when necessary. By a specially designed successive overrelaxation method with a variable relaxation factor, a sequence of finite systems of linear equations with increasing sizes is solved. Here the solution of the system associated with some value of \( N \) is used as starting point for the system associated with the next value of \( N \). The efficiency of the algorithm is further improved by choosing adaptively the accuracy number of the stopping criterion of the iterative method; this accuracy number \( \varepsilon \) is chosen smaller as \( N \) increases. We next describe the details of the algorithm sketched above.

Supposing an estimate for the integer \( N \) such that (14) holds and replacing the probabilities \( p_j \) by \( p_j^{N-j} \) for \( j \geq N \), we obtain from (11) and (12) after some manipulations the following system of linear equations

\[
P_j = \sum_{k=0}^N a_{jk} p_k, \quad j=0,\ldots,N
\]

(15)

\[
\sum_{j=0}^N p_j + \frac{T \tau}{1-\tau} P_N = 1
\]

(16)

where

\[
a_{jk} = a_{\min(j,j-k+c)}, \quad \text{otherwise.}
\]

Here \( a_i = e^{-\lambda D AD}/i! \), \( i \geq 0 \). Following the usual notation for the successive overrelaxation method, the operator \( B_\omega \) associated with a relaxation factor \( \omega \) transforms each vector \( x=(x_0,\ldots,x_N) \) into the vector \( B_\omega x \) whose components \( (B_\omega x)_i \) are recursively defined by

\[
(B_\omega x)_i = (1-\omega)x_i + \omega \left( \sum a_{ij} (B_\omega x)_j + \sum a_{ij} x_j \right)
\]

for \( i=0,1,\ldots,N \).

Assuming that the integer \( N \) is sufficiently large so that (15) has a solution, then this solution is an eigenvector of \( B_\omega \) with associated eigenvalue 1. Letting \( \lambda_1(\omega) \) be the eigenvalue having the largest absolute value among the eigenvalues of \( B_\omega \) unequal to 1, the standard successive overrelaxation method with a fixed relaxation factor \( \omega \) converges only if \( |\lambda_1(\omega)| < 1 \).

Moreover, the standard overrelaxation method has the best convergence rate for that value of \( \omega \) for which \( |\lambda_1(\omega)| \) is smallest. It should be noted that the optimal value of \( \omega \) may be rather sensitive to the parameters of the specific
problem considered and in some cases will be close to 1; in the algorithm below we keep \( w \) always between 1 and 2. In case \( \lambda_1(\omega) \) is real, it is possible to estimate \( \lambda_1(\omega) \) after some iterations of the overrelaxation method (this is done by the parameter \( r^h \) in the algorithm below). This estimate provides a method to formulate a successive overrelaxation algorithm in which the relaxation factor is dynamically adjusted in order to search for that value of \( \omega \) for which \( |\lambda_1(\omega)| \) is smallest. In [6] such an approach was proposed to solve the balance equations arising in the continuous-time Markov chain analysis of multi-server queueing systems; the resulting algorithm was instrumental in compiling the tablebook [7]. For the linear equations (15) and (16) this approach needs a modification in order to avoid divergence problems when the estimate of \( N \) is too low.

We now give the steps of the algorithm for the calculation of the state probabilities in the M/D/c queue.

Special-purpose overrelaxation method for the M/D/c queue.

Step 0. Choose \( N > c \) and \( x^0 \geq 0 \) with \( \sum_{i=0}^{N} x_i^0 = \frac{1}{1-c} \). Also, \( h := 0 \) and \( w := 1.20 \).

Step 1. \( \omega := 0 \), \( \lambda(\omega^{old}) := 1 \), \( f^h := r^h := \infty \).

Step 2. \( h := h + 1 \). Compute the vectors

\[
\begin{align*}
\mathbf{x}^h &: = \mathbf{x}^{h-1}, \\
\mathbf{x}_1^h &: = \left[ \sum_{i=0}^{N} \frac{x_i^h - x_i^{h-1}}{x_i^h} \right], \\
\mathbf{x}^\infty &: = \left[ \sum_{i=0}^{N} \frac{x_i^h}{\mathbf{x}_1^h} \right]
\end{align*}
\]

and the scalar

\[
f^h := \frac{1}{N} \left[ \sum_{i=0}^{N} \frac{x_i^h - x_i^{h-1}}{x_i^h} \right].
\]

If \( f^h \leq \epsilon_f \), then go to step 4. Otherwise

\[
r^h := \frac{f^h}{f^h + 1}.
\]

If \( r^h \leq 1 \) or \( r^h \geq 10 \), then \( \omega \) is likely too large and decrease \( \omega \) as \( \omega := 1.1 \omega \), put \( x^0 := \mathbf{x}^h \) and \( h := 0 \), and go to step 1. If \( r^h < 1 \) and \( r^h \) has sufficiently converged according to \( |r^h - r^{h-1}| / r^h | < 0.025 \), then go to step 3; otherwise return to step 2.

Step 3. \( \lambda(\omega) := r^h \). Test for one of the following four possibilities: (a) \( \omega \geq 0 \) and \( \lambda(\omega) > \lambda(\omega^{old}) \); (b) \( \omega > 0 \) and \( \lambda(\omega) \leq \lambda(\omega^{old}) \); (c) \( \omega < 0 \) and \( \lambda(\omega) > \lambda(\omega^{old}) \); (d) \( \omega = 0 \) and \( \lambda(\omega) \leq \lambda(\omega^{old}) \).

For the cases (a) and (d),

\[
\omega^{old} := \omega, \lambda(\omega^{old}) := \lambda(\omega), \omega := 1 + 0.85(\omega - 1),
\]

whereas for the cases (b) and (c),

\[
\omega^{old} := \omega, \lambda(\omega^{old}) := \lambda(\omega), \omega := 1 + 1.25(\omega - 1).
\]

Next, \( x := \mathbf{x}^h \), \( h := 0 \), \( f^h := r^h := \infty \), and go to step 2.

Step 4. If

\[
| \sum_{i=0}^{N} x_i^h + \sum_{i=0}^{N} \frac{x_i^h}{\mathbf{x}_1^h} | < 10^{-5}
\]

and

\[
| \sum_{i=0}^{N} x_i^h + \sum_{i=0}^{N} \frac{x_i^h}{\mathbf{x}_1^h} - \mathbf{c}^h | < 10^{-5},
\]

then the algorithm is stopped and the state probabilities \( p_i \) are obtained from \( p_i := x_i^h / \mathbf{x}_1^h \) for \( 0 \leq i \leq N \) (the above stopping criteria use that the probabilities sum to 1 and that the average number of busy servers equals \( c^h \)). Otherwise

\[
x_1^h := x_1^h \text{ for } 0 \leq i \leq c^h,
\]

\[
x_1^h := x_1^h / \mathbf{x}_1^h \text{ for } N < i \leq N+10
\]

and go to step 1.

3. NUMERICAL RESULTS

In this section we present some numerical results showing that the two-moment approximation (2) performs quite well for practical purposes provided \( c^h \) is not too large (say, \( 0 \leq c^h \leq 2 \)). For Erlang-2 service \( (c^h = 0.5) \) and \( H_2 \) service with balanced means and \( c^h = 2 \), table 1 gives the approximate and (nearly) exact values of the conditional waiting time percentiles \( n(p) \) with \( p = 0.5, 0.9, 0.95 \) and \( 0.99 \) for several values of the number \( c \) of servers. The number \( c \) of servers is varied as 2, 5, 10, 25 and 50. In all examples we assume a server utilization \( p = 0.8 \) and the normalization \( E(S) = 1 \). The (nearly) exact values of the conditional waiting time percentiles for Erlang-2 and \( H_2 \) services were computed from an extremely accurate approximation of the waiting time distribution function by a sum of a "sufficiently" number of exponential functions whose coefficients were obtained from the exactly computed state probabilities, see [6] and [7]; the accurateness of this approximation has been tested by using the exact relation \( E[L(L-1)\ldots(L-k+1)] = \mathbf{E}[W_i^k], k \geq 1 \) for the \( M/G/1 \) queue. Also, we include for convenience in table 1 the exact values of the delay probability \( P_\infty \).
Table 1. Numerical results.

<table>
<thead>
<tr>
<th>p</th>
<th>0.5</th>
<th>0.9</th>
<th>0.95</th>
<th>0.99</th>
<th>( \Pi_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>c=2</td>
<td>exa</td>
<td>1.341</td>
<td>4.286</td>
<td>5.554</td>
<td>8.498</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>1.332</td>
<td>4.282</td>
<td>5.551</td>
<td>8.497</td>
</tr>
<tr>
<td>c=5</td>
<td>exa</td>
<td>0.554</td>
<td>1.735</td>
<td>2.242</td>
<td>3.420</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.556</td>
<td>1.732</td>
<td>2.239</td>
<td>3.417</td>
</tr>
<tr>
<td>c=10</td>
<td>exa</td>
<td>0.285</td>
<td>0.881</td>
<td>1.135</td>
<td>1.724</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.286</td>
<td>0.877</td>
<td>1.131</td>
<td>1.721</td>
</tr>
<tr>
<td>c=25</td>
<td>exa</td>
<td>0.118</td>
<td>0.364</td>
<td>0.467</td>
<td>0.703</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.119</td>
<td>0.365</td>
<td>0.466</td>
<td>0.701</td>
</tr>
<tr>
<td>c=50</td>
<td>exa</td>
<td>0.061</td>
<td>0.189</td>
<td>0.241</td>
<td>0.361</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.061</td>
<td>0.191</td>
<td>0.243</td>
<td>0.362</td>
</tr>
</tbody>
</table>

**H**2-service with \( c_S^2 = 2 \)

<table>
<thead>
<tr>
<th>p</th>
<th>0.5</th>
<th>0.9</th>
<th>0.95</th>
<th>0.99</th>
<th>( \Pi_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>c=2</td>
<td>exa</td>
<td>2.361</td>
<td>8.818</td>
<td>11.62</td>
<td>18.11</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>2.534</td>
<td>8.705</td>
<td>11.37</td>
<td>17.55</td>
</tr>
<tr>
<td>c=5</td>
<td>exa</td>
<td>0.890</td>
<td>3.409</td>
<td>4.526</td>
<td>7.123</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.967</td>
<td>3.445</td>
<td>4.510</td>
<td>6.981</td>
</tr>
<tr>
<td>c=10</td>
<td>exa</td>
<td>0.426</td>
<td>1.634</td>
<td>2.186</td>
<td>3.479</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.468</td>
<td>1.699</td>
<td>2.231</td>
<td>3.466</td>
</tr>
<tr>
<td>c=25</td>
<td>exa</td>
<td>0.161</td>
<td>0.603</td>
<td>0.811</td>
<td>1.314</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.179</td>
<td>0.651</td>
<td>0.865</td>
<td>1.362</td>
</tr>
<tr>
<td>c=50</td>
<td>exa</td>
<td>0.077</td>
<td>0.280</td>
<td>0.376</td>
<td>0.612</td>
</tr>
<tr>
<td></td>
<td>app</td>
<td>0.085</td>
<td>0.309</td>
<td>0.412</td>
<td>0.658</td>
</tr>
</tbody>
</table>

REFERENCES