Transient flows in queueing systems via closure approximations

C. Roger Glassey [a] and Sridhar Seshadri [b]

[a] *Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA*
[b] *Walter A. Haas School of Business, University of California, Berkeley, CA 94720, USA*

Received 20 April 1992; revised 2 July 1992

We present a method for obtaining approximations to the distribution of flow times of customers in arbitrary queueing systems. We first propose approximations for uni-variate and multi-variate distributions of non-negative random variables. Then using a closure approximation, we show that the distribution of flow time can be calculated recursively. Computational results for the single server, multi-server and tandem queues are encouraging, with less than 5% *average* error in the mean flow time in most cases. The average error in the variance of flow times is found to be less than 10% for the more regular distributions.

Keywords: Approximation of distributions; closure approximation; queueing network; sojourn time; transient analysis.

1. Introduction

In many applications understanding the transient behavior of queueing systems is at least as important as obtaining steady state results. Examples of such applications in manufacturing include due-date setting, performing dynamic load calculations, predicting congestion levels based on changing resource availability and computing exit time distribution of jobs based on current shop status. The lack of tractable analytical models or approximations for analyzing transients is currently a drawback; and to date, simulation has been the most convenient tool used for this purpose. In this paper, we present a method for obtaining approximations for the flow time of customers in an arbitrary queueing system. We first propose approximations for uni-variate and multi-variate distributions of non-negative random variables. Then based on a closure approximation, we show that the distribution of the flow time can be computed in an arbitrary network of queues.

The problem of computing the distribution of flow times is known to be computationally difficult and there is a trade-off between accuracy of the results and the time required to obtain them. Manufacturing data is never exactly accurate. It is also not useful to carry out calculations to greater degree of precision than the data. So, we arbitrarily chose an error of $\pm 10\%$ in the expected value of the flow time and up to $\pm 15\%$ in the variance as acceptable in our context of due-date setting in a manufacturing shop. But we required that the computing time should be significantly smaller than that needed for simulating the system to obtain comparable error bounds.

Closed form transient results for the single server queue have been known for the special case of birth-death processes [6, p.53] and it is possible to work with transforms and inversion techniques for characterizing transients in several variants of the M/G/1 queue. However, obtaining the transform of say the number in the system for the general GI/G/1 case is not possible. But by approximating the service and arrival distributions and thereby working with a continuous time Markov chain we can obtain differential equations for the time dependent probabilities. Because of the difficulty in working with a large system of differential equations, researchers, such as Kotiah [7], Leese and Boyd [8], and Rothkopf and Oren [13], have used approximations (truncation) of one form or the other to simplify the calculations. These approaches numerically solve a system of differential equations either directly or through transform inversion techniques. Such techniques however do not meet our computational objective, as calculating the steady-state distribution can itself take from several seconds to minutes on a computer. Diffusion approximations have been used in transient analysis, extensively so by Newell [12]. These approximations however need special traffic conditions before they can be applied. Finally we are not aware of reported results pertaining to transient quantities in multi-server queues or general queueing systems.

We therefore wanted to devise an approximation technique for a general queueing system, to compute the flow times of the first, second, third, ... customers successively, where each recursion uses just the *additional* information generated at the previous step. This way the computational requirements can be kept minimal. Secondly, we wanted the same algorithm to be applicable for different service and inter-arrival distributions. So a primary requirement was to approximate distributions of random variables using a common functional form. There are two approximation techniques available that can serve the purpose; namely through the use of bi-lateral phase-type distributions [16] or the Laguerre transform [5]. However we did not find ways to handle the calculations involved in keeping with the computational objectives set our earlier. Hence we have used approximation schemes that are tailored to permit efficient calculations.

The basic ideas used in the proposed closure approximations are the assumptions: (i) of a functional form for approximating the distributions of random variables from limited and easily computed information; and (ii) that under the operations of scaling, adding, finding extrema and computing the positive part of random variables, the resulting distribution can be re-cast in the same functional form. In this schema, a positive values random variable is represented as a mixture of a unit mass at zero, and two other random variables distributed as Erlang of suitable degree and rate. This representation is based on the first three moments of the random variable and the value of the Laplace–Stieljes transform at the inverse of the mean. For multi-variate distributions, we additionally use the covariance matrix and represent the distribution as a mixture of independent vectors of random variables, each component of which is distributed as Erlang of high degree. In the uni-variate case we show that the relevant parameters can be obtained through an efficient search procedure. In the multi-variate case, we use a pre-specified grid and a linear program with a column generation scheme to obtain the representation. We have performed tests for the single server and tandem queues. The *average* error in the expected waiting time for a wide range of arrival and service distributions was less than 5% in most cases and the average error in the variance was less than 10% for the more regular distributions. However, the computing time apparently increases exponentially with the number of random variables jointly approximated in a recursion. So our method as currently implemented is not competitive with simulation for four or more random variables, for example the GI/G/4 queue.

The major drawback in applying the closure approximation is the time required to compute transients in a system with four or more servers, and two or more classes of customers. The main computational issue is solving the linear programs for deciphering distributions in an efficient manner. Even here, it is not the linear program itself, but the non-linear program that has to be solved for generating columns, which takes most of the time. Improving the solution procedure for the non-linear program is an area for future research. Secondly, the error generated by the approximation needs analysis. Thirdly, it is possible to add a few more constraints to the linear program without significantly affecting the solution time. Determining the nature of additional constraints that will improve the estimates of the variance of the flow times is another area for future work. Finally, the methods outlined in this paper need testing for priority queues and situations where there is feedback. The presentation is in three sections, covering the single server, the multi-server queues, and the tandem case.

2. Single server queues

We consider the standard GI/G/1 queue. The service time of the nth customer is S_n, where the $\{S_n\}$ form an i.i.d. sequence of random variables. The inter-arrival time between the n-1st and the nth customers is T_n, where $\{T_n\}$ forms an i.i.d. sequence. The closure approximation for uni-variate distributions is outlined in this section and then the computational aspects of the recursion are given, followed by numerical results.

The proposed closure approximation

We propose the following simple approximating scheme for non-negative uni-variate distributions that can be used to carry out the recursion in the manner mentioned above. We define a class Ξ of random variables. Ξ contains non-negative random variables that can be expressed as a mixture of a unit mass at zero and two other random variables with the Erlang distribution. Given X, $Y \in \Xi$, the following properties are assumed:

A. *Representation*: The distribution of X can be approximated by a mixture of a unit mass at zero and two random variables with the Erlang distribution.
B. *Determination*: The parameters of the mixture can be obtained from the first three moments of X and the value of its Laplace–Stieljes transform (LST) at the inverse of the mean of X.
C. *Closure*: Under the operations of addition ($X + Y$), determining the positive part of the difference ($[X - Y]^+$), computing the maximum or minimum ($\max(X, Y)$, $\min(X, Y)$), or scaling; the resulting random variable belongs to Ξ. In other words Ξ is closed under the five operations.

The use of the first two moments and the Erlang distribution in approximations is in no way new. For example see Sevcik et al. [15] and Shanthikumar and Sumita [17]. The hyper-exponential form has also been used by Seelen et al. in producing the Tables for Multi-Server Queues [14].

We include the unit mass at zero, because the positive part operation is critical in carrying out the Lindley recursion [9] for the GI/G/1 queue. The Laplace–Stieljes transform value helps in determining this mass, because a larger value indicates a larger mass at zero. The choice of the point at which we evaluate the transform has been dictated by computational convenience.

Finally, the third moment has been included to *control* the variance. For example the variance of the steady state delay in the M/G/1 operated under the FCFS discipline depends on the third moment of the service time distribution [19, p.394].

We test the closure approximation in the experiments described below. In one set of cases we begin with service and inter- arrival time distributions in the class Ξ and use the closure assumption to calculate the flow time distributions of successive customers. In another set of experiments, we first approximate the distribution of the service and inter-arrival time by using the representation of this class, and then use the closure approximation.

Computational scheme for single server systems

There are two parts to carrying out the recursion for the GI/G/1 queue. At each stage we need to decipher the parameters of the delay distribution. Once the distribution is deciphered, the calculations for the next step in the recursion need to be carried out. In deciphering a distribution say of a random variable X, we are given the first three moments, EX, EX^2, EX^3, and the value of the LST at the inverse of the mean, $LST_X(1/EX)$. For notational convenience we shall call these four data items a, b, c, and d. The form of approximation sought is either

$$X \sim (1-\alpha-\beta)\cdot U_0 + \alpha\cdot E_{k-1}(r) + \beta\cdot E_k(r)$$

or

$$X \sim (1-\alpha-\beta)\cdot U_0 + \alpha\cdot E_1(r) + \beta\cdot E_1(s)\,,$$

where U_0 represents a unit mass at zero and $E_k(r)$ the Erlang distribution of degree k and rate r. The form chosen will depend on the one that gives the minimum error for the third moment. A search scheme is used to determine the parameters α, β, and k. The computation proceeds as follows:

ALGORITHM FOR DECIPHERING A DISTRIBUTION GIVEN a, b, d, AND d

```
Step 0: Select an error limit for the third moment ($\epsilon$). Select a step $\delta$. Set $err = 1000$ and $\alpha_0 = 0$.
        Note: $\alpha_0$ is the current value of $(1-\alpha-\beta)$. $\delta$ is set to 0.01 usually.
Step 1: Do while $err > \delta$;
        Step 1.1: Normalize the moments and value of the LST using $\alpha_0$, i.e. by setting $f = 1-\alpha_0$; $a_1 = a/f$; $b_1 = b/f$; $c_1 = c/f$; and $d_1 = (d-\alpha_0)/f$.
        Step 1.2: Set $z = b_1/a_1^2$. If $z > 2$; then do step 1.4. Else
        Step 1.3: Set $k-1 = 1, 2, 3, \ldots, 10$. Compute the error in the LST value. Choose the $k$ that minimizes the error in the LST value. Go to step 1.5.
        Note: We allow the degree to be at most 11 and also require $k \leqslant z/(z-1)$.
        The calculations are: $x = k/z$; $y = (x^2 - x(k-1))^{0.5}$; $\eta = 1-k+x+y$; and if $\eta \leqslant 1$ then $trial = x+y$; $test = (trial/(trial+1))^{k-1} \cdot k/(trial+1)$; $error = |(d_1 - test)/test|$
        For the $k$ that gives minimum error; set $\alpha = (1-\eta)\cdot f$; $\beta = \eta \cdot f$; and $r = (k-1+\eta)/a_1$
        Step 1.4: If $d_1 > z/(z+2)$ or $d \leqslant 0.5$ go to step 1.6. Else; $\mu = 1/a_1$; $\sigma = z/2 - d_1 \cdot (z/2+1)$; $x = (2\cdot\mu^2\cdot d_1 - \mu^2)/\sigma$; $y = b_1 \cdot x/(2a_1) + \mu$; $root = (y^2-4x)^{0.5}$; $root1 = (y+root)/2$; $root2 = (y-root)/2$; $\eta = (a_1 \cdot x - root1)/(root2 - root1)$.
        Set $\alpha = \eta\cdot f$; $\beta = (1-\eta)\cdot f$; $r = root2$; $s = root1$ and $k = 1$.
        Step 1.5: Compute the error $err$ in the third moment.
        Step 1.6: $\alpha_0 = \alpha_0 + \delta$.
        End do;
Step 2.0: Output the result.
END
```

It can be shown that we always obtain a suitable representation. However the value of the step size, δ, would need to be small and the starting value of α_0 set close to one for some extreme cases. The choice of $\delta = 0.01$ sufficed for determining the parameters of all the distributions of service and inter-arrival times reported in this paper.

For the second part of calculations, let D_n be the delay in queue of the nth customer. Then we have $D_1 = 0$; and

$$D_{n+1} = [D_n + S_n - T_n]^+, \quad n = 1, 2, 3, \dots .$$

The convolution is straightforward. To compute the positive part, it is enough to consider the case $[Y - X]^+$; where X is distributed as $E_k(r)$ and Y as $E_l(s)$. For coding purposes, it is easiest to work with the generic integral:

$$I_p(k, r, l, s) = \int_0^\infty \frac{x^{k-1} r^k}{k-1!} \mathrm{e}^{-rx} \int_x^\infty \frac{y^{l-1} s^l}{l-1!} \mathrm{e}^{-sy}\, \mathrm{d}y\, \mathrm{d}x .$$

The first integral is with respect to the inter-arrival distribution. The second involves the remaining work at the queue. Both the distributions are Erlang and are of suitable degree as well as rate. Using a function that returns the value of I_p, we can calculate the moments of $[Y - X]^+$, and also the value of the $LST_{[Y-X]^+}$ at the inverse of the mean. For example, the first moment and the value of the transform are given by

$$E(Y - X)^+ = \frac{l}{s} I_p(k, r, l+1, s) - \frac{k}{r} I_p(k+1, r, l, s),$$

$$E(\mathrm{e}^{-\mu(Y-X)^+}) = \left(\frac{r}{(r-\mu)}\right)^k \left(\frac{s}{(s+\mu)}\right)^l I_p(k, r-\mu, l, s+\mu) .$$

When computing the value of the transform, suitable modifications are required to guarantee against overflows. The overall algorithm for the GI/G/1 case is as follows:

ALGORITHM FOR THE GI/G/1 RECURSION

```
Step 0: Set $D_1 = 0$; number of customers $= N$; $i = 2$.
Step 1: Do while $i \leqslant N$
        Step 1.1: Decipher the distribution of $D_{i-1}$.
        Step 1.2: Compute the first three moments and the transform value of
        $D_i = [D_{i-1} + S - T]^+$.
        Step 1.3: $i := i + 1$.
        End do.
END
```

Numerical results for the GI/G/1 case

The algorithm for the GI/G/1 queue was coded in *f77* and implemented on the network of SUN machines in the Industrial Engineering and Operations Research Department, U.C. Berkeley. The numerical results for different service and inter-arrival distributions are shown in table 1. The distributions with squared coefficient of variation (scv) greater than one are hyper-exponential. The formula used for the distribution function of a hyper-exponential random variable T with scv C_T^2 is (cf. (15) in [17]):

$$P[T \leqslant t] = 1 - a\exp\{-rt\} - (1-a)\exp\{-st\}, \quad t>0;$$

$$a = \tfrac{1}{2}\left(1 - \sqrt{(C_T^2 - 1)/(C_T^2 + 1)}\right); \quad r = \frac{2a}{ET}; \quad S = \frac{2(1-a)}{ET}.$$

The calculations for the first 200 customers took less than 8 seconds in all cases. The running time for the simulation was four to five minutes on the average for 10000 replications. Note that the flow time distribution had to be tracked in the simulations. The simulation results in table 1 are based on 10000 replications. The columns give the mean of the expected waiting time (W) and variance of the waiting time (var(W)) for the last ten customers. Note that we obtain an approximation to the steady state value for the cases where the load (ρ) is less than 0.5. In the examples given in the table, the queues have not yet stabilized when $\rho = 0.9$. The measure of error we have used is the mean absolute deviation of the computed values from the simulated ones over *all* the customers. The error in the expected waiting time is less than 2% in all but one case, where the error is 2.524%. The error in the

Table 1
Transient flows in the GI/G/1 queue.

System [a)]	Mean W last ten [b)]		Mean % abs error	Var(W) last ten [b)]		Mean % abs error
	Simul.	Comput.		Simul.	Comput	
M/M (rho = 0.5)	2.04 ± 0.06	2.00	1.48	4.05 ± 0.59	3.99	1.60
M/M (rho = 0.9)	15.56 ± 0.43	15.48	1.13	205.05 ± 27.75	216.41	4.66
M/E_2 (rho = 0.5)	1.78 ± 0.04	1.75	1.47	2.08 ± 0.33	1.94	4.68
M/E_2 (rho = 0.9)	12.65 ± 0.34	12.80	1.33	127.52 ± 17.88	118.63	3.07
M/CS4 (rho = 0.5)	3.48 ± 0.16	3.50	1.35	29.18 ± 4.36	29.96	2.23
M/CS4 (rho = 0.9)	28.49 ± 0.89	28.13	1.21	870.50 ± 121.59	913.49	3.20
M/CS8 (rho = 0.5)	5.53 ± 0.30	5.50	1.67	102.76 ± 14.49	99.79	3.42
E_2/E_4 (rho = 0.5)	1.30 ± 0.02	1.28	1.95	0.56 ± 0.11	0.56	3.99
E_2/E_4 (rho = 0.9)	7.40 ± 0.19	7.52	1.08	40.63 ± 6.14	32.61	10.48
E_2/CS2 (rho = 0.5)	2.13 ± 0.08	2.08	1.99	7.69 ± 1.11	6.68	10.81
E_2/CS2 (rho = 0.9)	17.71 ± 0.54	17.73	1.74	329.51 ± 43.72	347.98	6.45
CS2/CS4 (rho = 0.5)	4.19 ± 0.19	4.08	2.00	39.54 ± 5.60	37.30	3.16
CS2/CS4 (rho = 0.9)	36.29 ± 1.11	36.08	1.76	1372.64 ± 172.69	1464.29	6.64
CS4/CS8 (rho = 0.5)	7.31 ± 0.36	7.19	1.97	147.44 ± 20.45	141.57	3.52
CS4/CS8 (rho = 0.9)	56.99 ± 1.78	55.41	2.52	3502.64 ± 438.98	3663.34	3.02

a) CS2, CS4 and CS8 stand for scv of 2, 4, and 8.
b) 99% confidence intervals are also averaged over the last ten customers.

variance is less than 7% in all but two cases, in which it is less than 11%. The approximation is therefore seen to be robust over different load conditions and for different values of the squared coefficient of variation of both arrival and service distributions. Figures 1 and 2 show the plots of the waiting times and variance of

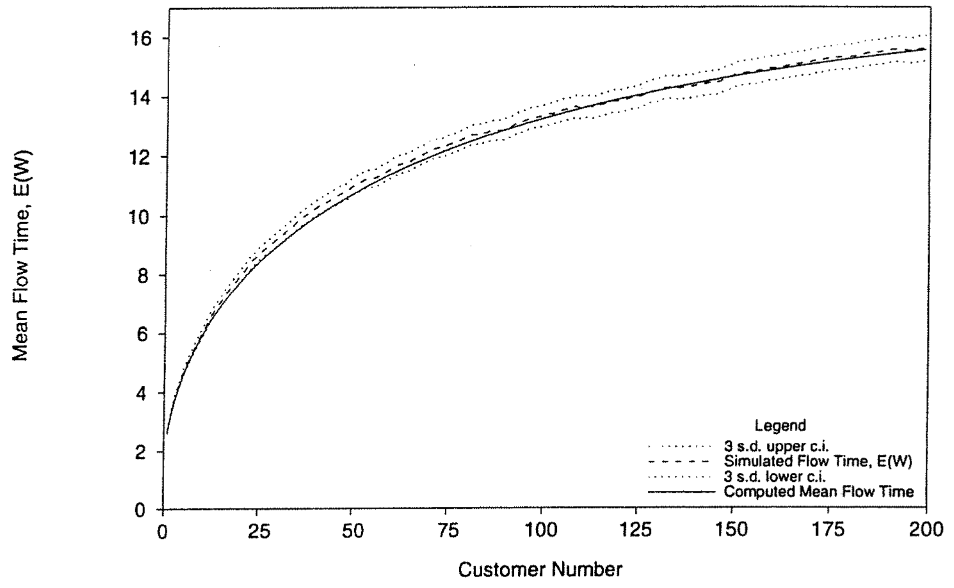

Fig. 1(a). Mean flow time in the M/M/1 queue (rho = 0.9).

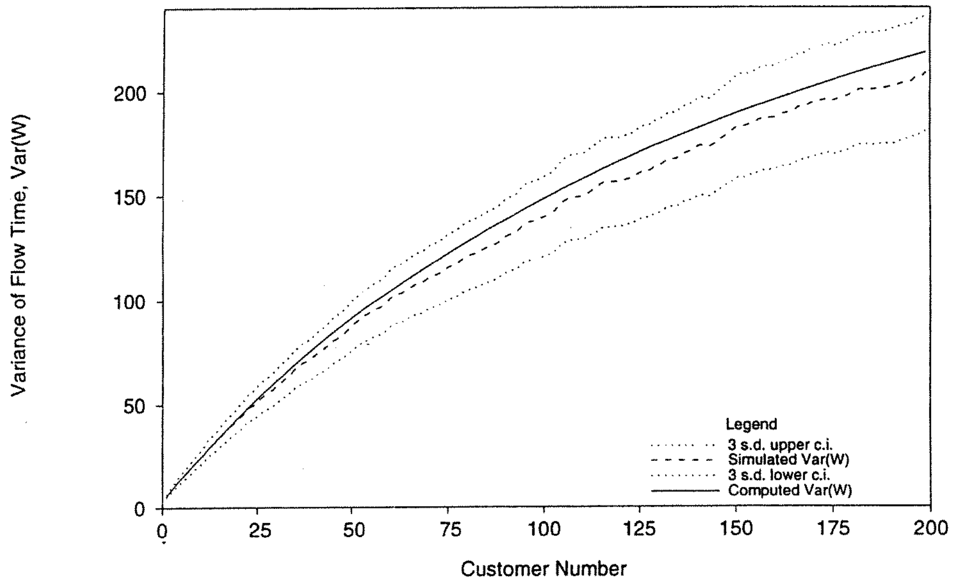

Fig. 1(b). Variance of flow time in the M/M/1 queue (rho = 0.9).

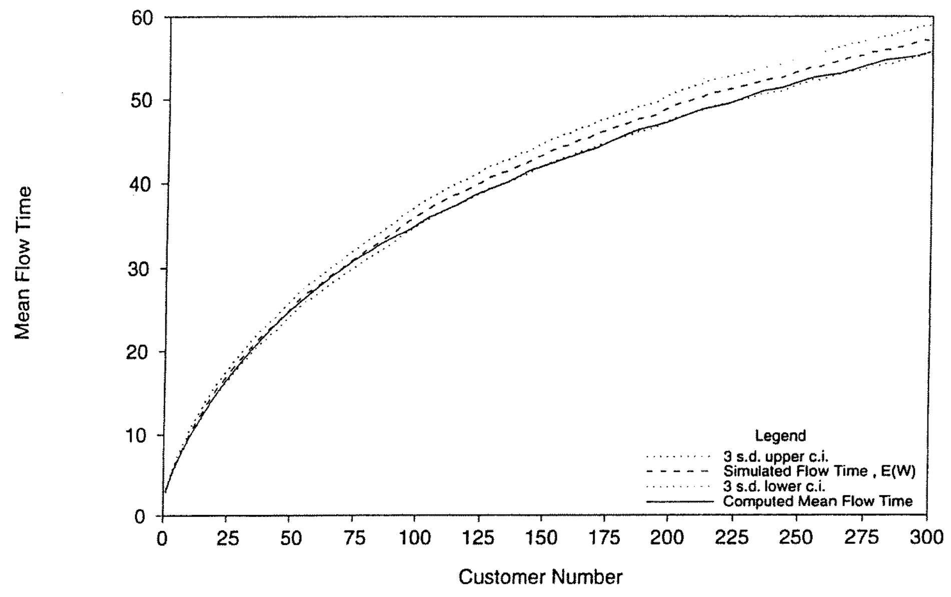

Fig. 2(a). Mean flow time in the CS4/CS8/1 queue (rho = 0.9).

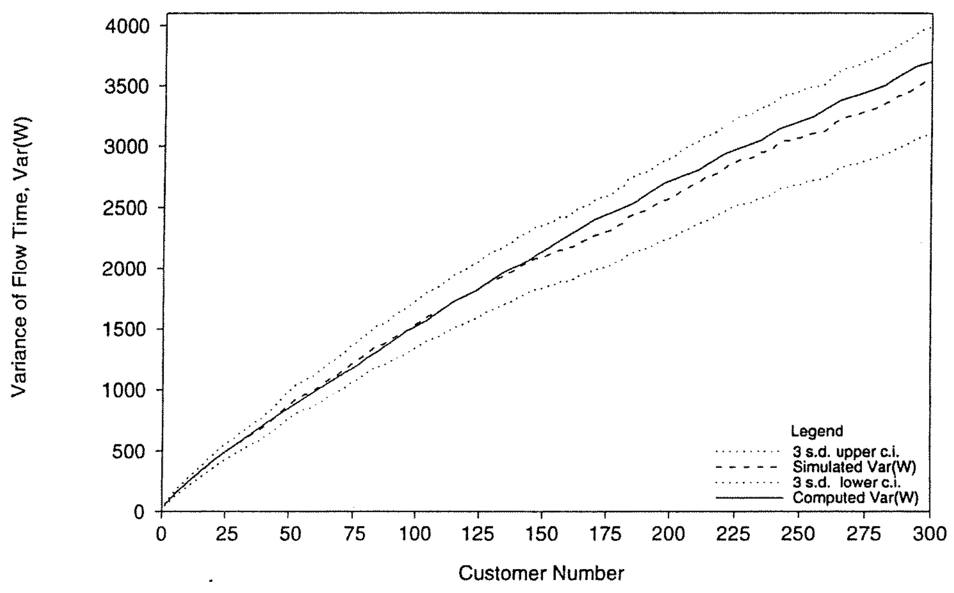

Fig. 2(b). Variance of flow time in the CS4/CS8/1 queue (rho = 0.9).

the waiting time for two cases where the load is 90%. The simulated values and the confidence bounds are plotted along with the computed values. The slow rate of convergence to the steady state values as seen in these plots is an important reason for conducting transient analysis of queueing systems.

3. Multi-server queues

In this section, we consider the standard GI/G/C queue, operating under the FCFS discipline. The main result in this section is the proposed approximation for multi-variate distributions and the resolution of computational issues in obtaining the approximate distribution.

Firstly, the GI/G/1 queue is known to be a difficult problem area (for example see the discussion related to the PH/PH/C system in [11]. A review of the approximations and exact results for steady state quantities is given in Hoorn and Seelen [3]). We are not aware of any results pertaining to transient analysis of this system. Secondly, there are no suitable approximations in the literature for multi-variate probability distributions. For example, in the bi-variate case, Gumbel's bi-variate exponential distribution [2] allows a range of correlation between -0.40365 and 1, and the Marshall and Olkin bi-variate exponential distribution [10] only positive correlation between the two variables. In fact, most bi-variate extreme value distributions suffer from this drawback (see for example chapters 40 and 41 of Johnson and Kotz [4] and chapter 8 of Springer [18]). The multi-variate phase-type distributions are conceptually the extension to the uni-variate phase-type distributions (see Assaf et al. [1]). But it is not easy to obtain a representation given the data for a distribution.

However, the multi-variate phase-type distributions provided the idea for the proposed scheme given below. Consider the representation of the remaining work vector in the GI/G/C queue. If we intend to use the usual GI/G/C recursion [19, p.494], then the vector should be maintained in an ordered fashion. The phase-type representation of this vector will then be in the form of a rate matrix with blocks on the diagonal. Therefore it seemed reasonable to seek an approximation of the work vector W of say C components, in the form of mixtures of vectors of independent random variables distributed as Erlang of suitable degree and rate. Formally,

$$W = (W_1, W_2, \ldots, W_C)^T = \sum_j \alpha_j \cdot (X_{1_j}, X_{2_j}, \ldots, X_{C_j})^T ,$$

where

$$X_{i_j} \sim E_{k_{ij}}(r_{i_j}) \quad \text{or} \quad X_{i_j} \sim U_0, \quad \text{and} \quad \sum_j \alpha_j = 1 .$$

From a density type of argument, keeping the degree of the Erlang distributions the same and very large seems preferable ($k_{i_j} = k_{l_m} \gg 1$). With this motivation, we propose the following closure approximation.

Closure approximation for multi-variate distributions

Define a class Λ of multi-variate distributions on the non-negative orthant that have finite moments up to the order three. Given X and $Y \in \Lambda$, with support on the non-negative orthant of $\mathbf{R}^k$, the properties assumed for this class are:

A. *Representation*: On a suitable grid $\mathbf{G}^k = [(a_{11}, a_{12}, \ldots, a_{1l}) \times (a_{21}, a_{22}, \ldots, a_{2l}) \times \ldots \times (a_{k1}, a_{k2}, \ldots, a_{kl})]$, X can be written as a mixture of vectors of Erlangs of degree m with mean a_{ij} (or a unit mass at zero). The formal representation is:

$$X \sim \sum_{i=1}^{l} \alpha_i \cdot \left(E_m\left(\frac{1}{a_{i1}}\right), E_m\left(\frac{1}{a_{i2}}\right), \ldots, E_m\left(\frac{1}{a_{ik}}\right)\right),$$

where with some abuse of notation, the $E_m(.)$ represent independent random variables distributed as Erlang of degree m and corresponding rates. If the mean is zero (infinite rate), then we have a unit mass at zero.

B. *Determination*: The distribution of X can be obtained from the first three moments, the value of the LST at the inverse of the mean and the co- variance matrix of the components of X.

C. *Closure*: The class Λ is closed under scaling, addition, taking the positive part and computing extreme values, similar to Ξ.

Assumption A is based on the reasoning given previously. Assumption B is in the same spirit as B for Ξ. And C permits the recursion. The degenerate distributions satisfy the closure assumption. And multi-variate normal distributions can be approximated by this class and could satisfy the closure assumption as a good approximation. The empirical evidence given below shows that these assumptions lead to a good approximation for the remaining work vector in queueing systems.

Computational scheme for multi-server systems

The number of grid points will need to be reasonably large to permit this scheme to work for any arbitrary distribution. Even with just 25 points per dimension, the number of points in the grid becomes very large for small k (the number of dimensions). Leaving alone the problems of efficiency, we have far fewer constraints than variables for fitting a given distribution. It was this consideration that led to the use of a *fixed* grid as against allowing the grid points to be parameters. We can use an optimization approach for determining the probabilities α_i in any case. But with a fixed grid, linear programming can be used. This gives two advantages, namely, a good algorithm (permitting column generation) and a compact representation. The latter because the basic solutions to the linear program given below have at most $(1 + 4k + k(k-1)/2)$ non-zero components. Lastly, experiments with the degree m of the Erlang distributions, convinced us that but for a problem with stability (discussed later), we can use point masses in the approximations. The details of the linear program and the column generation scheme are given in the next subsection. And following that we discuss the recursion.

Deciphering a given distribution

We are given data on W a k-dimensional non- negative random vector. The data consists of the first three moments ($EW_i, EW_i^2, EW_i^3, i = 1,2,\dots,k$), the LST value of each component at the inverse of the mean ($LST_{W_i}(1/EW_i), i = 1,2,\dots,k$), and the product-moment matrix ($EW_iW_j, i = 1,2,\dots,k; j = 1,2,\dots,k;\ i \neq j$).

We construct the grid $\mathbf{G}^k$ on the rectangle ($[0,u_1] \times [0,u_2] \times \dots \times [0,u_k]$), where the $u_i, i = 1,2,\dots,k$, are pre-specified constants. There are p points in the interval $[0,u_i]$ and these are equally spaced. Thus the points in the grid are:

$$(x_{1,i_1}, x_{2,i_2}, \dots, x_{k,i_k}), \quad i_1, i_2, \dots, i_k \in \{1,2,\dots,p\};$$

$$x_{j,1} = 0; \quad (x_{j,l} - x_{j,l-1}) = u_j/p, \quad j = 1,2,\dots,k, \quad l = 2,3,\dots,p.$$

We shall index these points as $\{1,2,\dots,p^k\}$. Let $N = p^k$. The point with index j has coordinates $\{j_1, j_2, \dots, j_k\}$ obtained from the expansion of j in the base (p). The linear program on this grid, $\mathbf{P(G,W)}$, is set up to minimize the fitted error in the third moments and the values of the LST. We found it convenient to use the given values of the third moment and LST as *upper* bounds. The program can then be written as:

$$\mathbf{P(G,W)}\text{: } Min \sum_{i=1}^{k} (s_{3i} + s_{ti})$$

$$\text{st. } (1) \sum_{n=1}^{N} y_n = 1$$

$$(2) \sum_{n=1}^{N} x_{j,n_j} \cdot y_n = EW_j, \quad j = 1,2,\dots,k$$

$$(3) \sum_{n=1}^{N} x_{j,n_j}^2 \cdot y_n = EW_j^2, \quad j = 1,2,\dots,k$$

$$(4) \sum_{n=1}^{N} x_{j,n_j}^3 \cdot y_n + s_{3j} = EW_j^3, \quad j = 1,2,\dots,k$$

$$(5) \sum_{n=1}^{N} e^{-x_{j,n_j}/EW_j} \cdot y_n + s_{tj} = LST_{W_j}\left(\frac{1}{EW_j}\right), \quad j = 1,2,\dots,k$$

$$(6)\sum_{n=1}^{N} x_{j,n_j} \cdot x_{l,n_l} \cdot y_n = EW_jW_l, \quad l = 1,2,\ldots,k; j \neq l$$

and $\quad y_1, y_2, \ldots, y_k; s_{31}, s_{32}, \ldots, s_{3k}; s_{t1}, s_{t2}, \ldots, s_{tk} \geqslant 0.$

This linear program has typically far too many columns than rows. So a column generation scheme was devised to speed up the solution process. The procedure begins with artificial variables assigned to constraints (1), (2), (3) and (6). These and the slacks in (4) and (5) constitute the initial solution. Let **B** be the current basis. Let the costs of the variables in the current basic solution be $\mathbf{C_B}$. Then $\mathbf{C_B} \cdot \mathbf{B}^{-1}$ is the vector of row multipliers.

To find an entering variable (if one exists) that will improve the solution, we need to find a point j in the grid such that

$$C_B \cdot B^{-1} \cdot a_{j.} < 0\,,$$

where, the elements of $a_{j.}$ are obtained from the coefficients in the constraints (1)–(6). It is possible that one of the slacks is a good choice of an entering variable, in that case a suitable modification is made. To obtain the index j; we solve a non-linear program, $\mathbf{P_n}(\mathbf{C_B} \cdot \mathbf{B}^{-1}, \mathbf{G}^k)$:

$$\mathbf{P_n}(\mathbf{C_B} \cdot \mathbf{B}^{-1}, \mathbf{G}^k) : \text{arg min}\, j : [C_B \cdot B^{-1} \cdot a_{j.} : j \in \{1,2,\ldots,N\}]\,.$$

A grid refining method is adopted in the search procedure used for solving $\mathbf{P_n}$. Experience in solving the problem **P**, indicates that the following are crucial to obtain a solution within 100 or less iterations when $k = 3$:

Scaling: Suitable scaling of the right hand side (rhs) coefficients in (2), (3), (4) and (6) improves the rate of convergence to the desired solution. We found that scaling the rhs of variable j by the maximum of $[(EW_j^2)^{1/2}, (EW_j^3)^{1/3}]$ gave good results.

Upper Bounds: The upper bounds u_j need careful setting. In the implementation they were set at the value of the mean plus three times the standard deviation and in some cases to five times the standard deviation, when the linear program did not converge within 300 iterations.

Tolerance: Finding the optimal solution to the linear program is too time consuming. So, the linear programs were not solved to optimality. Instead a tolerance level, in terms of the percentage error with respect to the starting value of the objective function, was specified. The values used were 0.1% and 1%. The latter was used if the linear program did not converge with the initial conditions (see below).

Number of sub-divisions p: We found that using 25 subdivisions gave the best trade-off between accuracy and running time.

As can be seen from the discussion, the process of deciphering the distribution is not simple as in the case of uni-variate distributions. The linear programs normally converge within 30 iterations for $k = 2$, and 80 iterations for $k = 3$. 80–100 iterations imply that the approximations were taking almost as much or more time to calculate as doing simulations; and so in keeping with out initial objectives we restricted the numerical experiments to k less than or equal to 3. The overall algorithm for deciphering a multi-variate distribution can now be stated.

ALGORITHM FOR DECIPHERING A MULTI-VARIATE DISTRIBUTION

```
Input: The first three moments, LST values, and covariance matrix of a $k$-dimensional r.v. The number of sub-divisions $p$, upper bound factor $f_u$, and tolerance % $\epsilon$. Also scale factors $s_u$ and $s_\epsilon$ to be used if the linear program was found to be infeasible or did not converge within 300 iterations. Finally a tolerance level $\delta$ for the non-linear program (usually set to $10^{-7}$).
    Step 0: Scale the right hand sides of (2), (3), (4), and (6) as discussed under scaling. Create the initial basis, the starting solution (by adding artificial variables) and compute the initial value $z_0$ of this solution. Note: The artificial variable corresponding to (1) was given a weight of 100 and those associated with (2), (3) and (6) a weight of 1 in the objective function. Set Flag= 0.
    Step 1.0: Do while the number of iterations less than 300; and the current value $z$ of the objective function is greater than $\epsilon$% of $z_0$.
        Step 1.1: Calculate $C_B.B^{-1}$. Check if any slack variable can enter the solution. If not,
        Step 1.2: Solve the non-linear program $\mathbf{P_n}$ and if the solution is $>\delta$ do step 1.4. Else,
        Step 1.3: Increase the upper bounds temporarily by a factor of 1.25. If the increased bounds are less than $s_u.f_u$ then do step 1.1. Else do step 3.
        Step 1.4: Revise the basis, the basic solution and value of the objective function.
    Enddo
    Step 2: Output the solution and return.
    Step 3: If Flag= 1, then return "Problem infeasible". Else revise the factors $f_u$ and the error tolerance by $s_u$ and $s_e$. Set Flag = 1. Do Step 1.
END
```

GI/G/C recursion calculations

There are two different recursion schemes available for determining the remaining work vector in the GI/G/C queue. In the first one, we simply keep track of which queue the arrival joins. This scheme was tried and did not do very well. The other scheme involves keeping the work vector in an ordered fashion. This proved to be better as it gave lower error in the approximated mean flow times. As in the case of the GI/G/1 queue, at each stage we have the representation of the work vector. The basic data required for the next step are the three moments, the LST values and the product moment matrix. Once this data has been obtained, it can be fed into the linear program to get the representation, etc. To obtain this data, it is adequate to consider the generic integral:

$$I(k,r,x) = \int_0^x \frac{y^{k-1} r^k}{(k-1)!} \mathrm{e}^{-ry} \, \mathrm{d}y$$

The integral is with respect to the inter-arrival distribution. As per the closure approximation, the remaining work is a point mass (at x in the integral above). Once this integral has been coded, computing the moments and LST values is through suitable calls to this function. Observe that by using point masses in the approximation, it is only required to sort the representation of work to keep the vector of work ordered. This was the basic consideration that led to the choice of point masses. This has certain drawbacks as explained in the following section. The details of the overall algorithm for the GI/G/2 case are given below. The general case is similar.

ALGORITHM FOR THE GI/G/2 QUEUE

Input: The first three moments and the LST value at the inverse of the mean for both the service time S and the inter-arrival time T distributions, the number of customers N, and the tolerance specifications required for solving the linear program.

Step 0: Decipher the distributions of the service and inter-arrival times using the three moment approximation used for the GI/G/1 queue. Initialize the work vector W^1 as $(S, 0)$. Note: The linear program is difficult to solve with one variable having too small a mean. So for small values (less than 0.001) of the mean we set the random variable equal to one that is exponentially distributed with mean 0.001. Set $i = 2$.

Step 1: Do while $i \leqslant N$;

Step 1.1: Decipher the work vector W^{i-1} using the linear program. Note: The output is at most 10 pairs of numbers, i.e. point masses, and the probability associated with each pair. Let the number of pairs be n, the associated probabilities p_j, and the numbers themselves $(x_j, y_j), j = 1, 2, \ldots, n$.

```
    **Step 1.2**: For each pair of point masses $(x_j, y_j)$, order the pair as $(s, l)$ (small, large); and computer the first three moments of $(s-T)^+$ and $(l-T)^+$ using suitable calls to the generic integral. Then compute the value of $LST_{(s-T)^+}(1/(E(s-T)^+))$ and $LST_{(l-T)^+}(1/(E(l-T)^+))$, again via the same type of integral.
    **Step 1.3**: Add the service time $S$ to $(s-T)^+$ using the approximation technique for uni-variate distributions.
**Enddo**
**END**
```

Numerical results for the GI/G/2 and GI/G/3 queues

The results for the two server and the three server queues are shown in tables 2(a) and (b). The format is the same as that of table 1. The mean absolute percentage error in the computed expected waiting times (W) of customers is less than 8% in all but two cases. The largest errors of nearly 14% are in the case where the service distribution has an scv of eight and the inter-arrival distribution an scv of four. As for the variance of W, the mean error is less than 15% in almost all cases. The exceptions are in the two cases mentioned above and also for the uniform distribution. With the uniform distribution (for the service time) we get a large error in the computed variance because the service times were approximated as Erlang distributions and so have a larger third moment. However the accuracy of the expected flow times does not get affected.

Some of the error in the variance is due to the tolerance levels set for the linear program. With very large third moments, there is a gradual loss in precision as the iterations progress. For the more variable service distributions, we have tried to correct the variance alone by scaling the point masses (usually up) to avoid loss of precision. This possibly accounts for the larger variances in these instances.

Figures 3 to 5 show the plots of the expected waiting time (W) and variance of W versus the customer number for typical cases. Another problem becomes evident from these plots, the delays do not increase monotonically. This is termed *instability* and has been referred to in two places earlier in the paper. We have used solid lines in the figures to show the computed values to highlight this problem. One reason for this instability is the use of point masses in the approximation. Preliminary testing with Erlang distributions of degree eight instead gave better results. The other reason for the instability is that we are not solving the linear program to optimality. The instability becomes more pronounced towards the end, when the queues are beginning to stabilize and the errors tend to compensate each other; thereby causing the swings. In fact, this self-correction is so important that when we tried to solve the successive linear programs starting with the solution obtained in the previous iteration, the mean waiting times kept increasing. A final observation in this regard is that a smooth curve drawn through the calculated values will track the simulation results rather well.

Table 2(a)
Transient flows in the GI/G/2 queue.

System [a]	Mean W last ten [b]		Mean % abs	Var(W) last ten [b]		Mean % abs
	Simul.	Comput.	error	Simul.	Comput	error
M/E_2 (rho = 0.5)	1.26 ± 0.03	1.29	1. 70	0.76 ± 0.12	0.83	7.49
M/E_2 (rho = 0.9)	6.42 ± 0.16	6.57	1. 91	28.21 ± 4.18	34.78	15.73
M/M (rho = 0.5)	1.33 ± 0.04	1.36	1. 60	1.47 ± 0.21	1.48	2.56
M/M (rho = 0.8)	4.24 ± 0.12	4.32	1. 87	15.18 ± 2.16	16.51	9.06
M/UNI (rho = 0.9)	12.44 ± 0.30	11.81	4.57	96.86 ± 14.76	118.21	22.22
E_2/E_2 (rho = 0.9)	4.92 ± 0.12	4.81	1. 89	15.47 ± 2.36	13.59	9.80
E_2/E_4 (rho = 0.9)	4.18 ± 0.09	4.39	4. 91	8.86 ± 1.49	9.51	9.50
M/CS4 (rho = 0.5)	1.75 ± 0.08	1.64	5. 00	7.82 ± 1.38	6.37	13.56
M/CS4 (rho = 0.9)	13.07 ± 0.42	14.15	5.92	199.47 ± 25.06	250.67	20.24
CS4/CS8 (rho = 0.9)	20.89 ± 0.71	21.62	14.40	543.04 ± 66.71	618.36	11.45

[a] CS2, CS4 and CS8 refer to scv of 2, 4, & 8; and UNI to the uniform dist.
[b] 99% confidence intervals are also averaged over the last ten customers.

Table 2(b)
Transient flows in the GI/G/3 queue.

System [a]	Mean W last ten [b]		Mean % abs	Var(W) last ten [b]		Mean % abs
	Simul.	Comput.	error	Simul.	Comput	error
M/E_2 (rho = 0.66)	2.65 ± 0.06	2.67	2.21	3.38 ± 0.55	3.30	6.19
M/E_2 (rho = 0.90)	6.94 ± 0.16	7.19	4.15	28.48 ± 4.39	34.27	11.48
M/M (rho = 0.66)	2.85 ± 0.08	2.89	3.03	6.52 ± 0.92	6.33	9.03
M/UNI (rho = 0.90)	6.69 ± 0.15	7.26	5.74	23.48 ± 3.78	28.74	20.27
E_2/CS2 (rho = 0.66)	2.87 ± 0.10	3.03	5.75	11.76 ± 1.98	11.70	7.25
E_2/CS2 (rho = 0.90)	8.83 ± 0.27	8.76	4.13	77.76 ± 10.37	74.68	8.50
E_2/E_4 (rho = 0.66)	2.26 ± 0.03	2.24	1.57	1.31 ± 0.28	1.26	4.57
E_2/E_4 (rho = 0.90)	4.85 ± 0.09	5.08	4.06	9.19 ± 1.68	9.26	5.91
CS2/CS4 (rho = 0.66)	4.30 ± 0.18	4.31	7.78	35.81 ± 5.77	33.92	11.91
CS2/CS4 (rho = 0.90)	14.29 ± 0.46	14.99	6.67	232.40 ± 29.59	212.11	11.80
CS4/CS8 (rho = 0.66)	5.80 ± 0.28	6.32	13.12	88.02 ± 15.20	96.15	21.38
CS4/CS8 (rho = 0.90)	19.57 ± 0.68	19.87	6.87	518.72 ± 65.42	499.72	14.16

[a] CS2, CS4 and CS8 refer to scv of 2, 4, & 8; and UNI to the uniform dist.
[b] 99% confidence intervals are also averaged over the last ten customers.

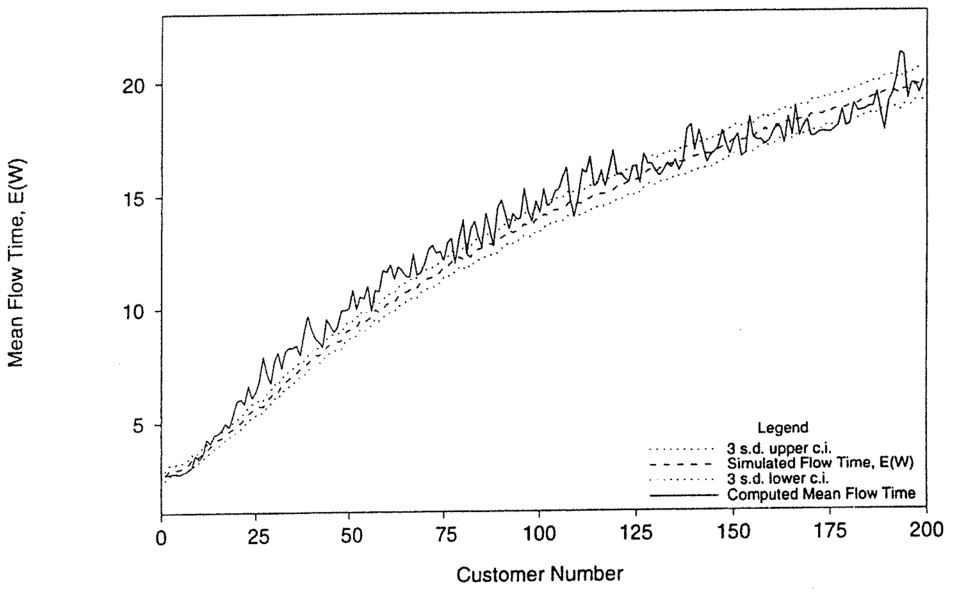

Fig. 3(a). Mean flow time in the CS4/CS8/3 queue (rho = 0.9).

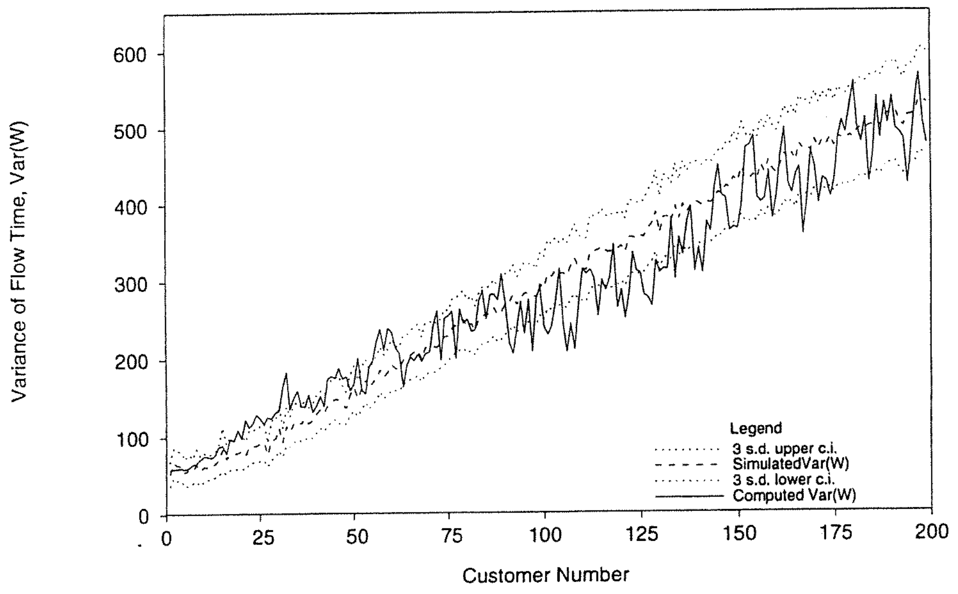

Fig. 3(b). Variance of flow time in the CS4/CS8/3 queue (rho = 0.9).

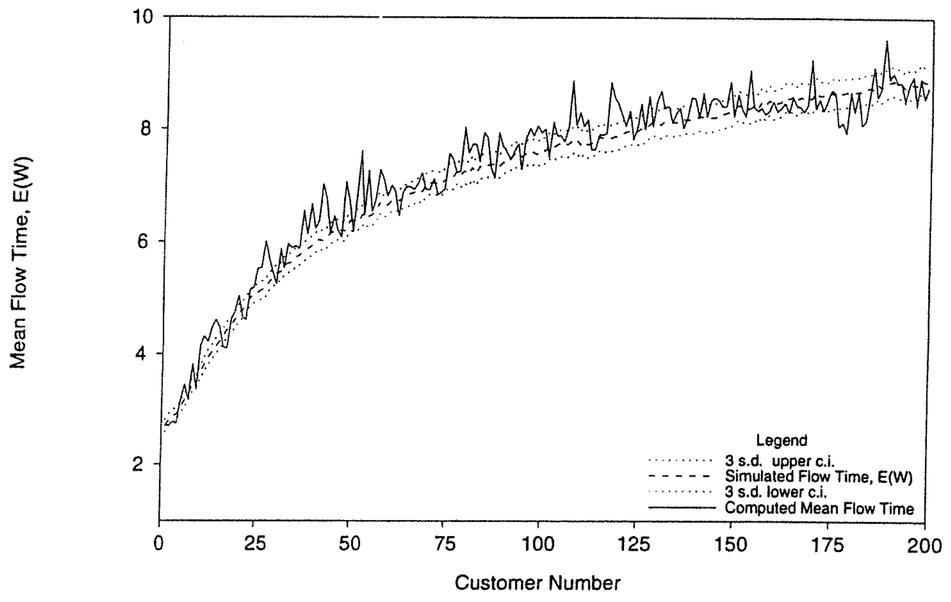

Fig. 4(a). Mean flow time in the E2/CS2/3 queue (rho = 0.9).

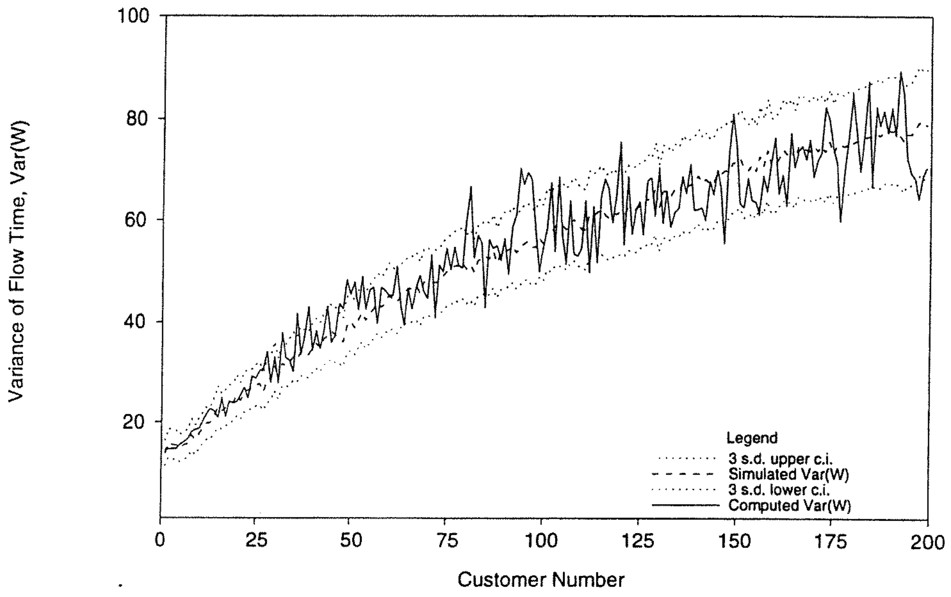

Fig. 4(b). Variance of flow time in the E2/CS2/3 queue (rho = 0.9).

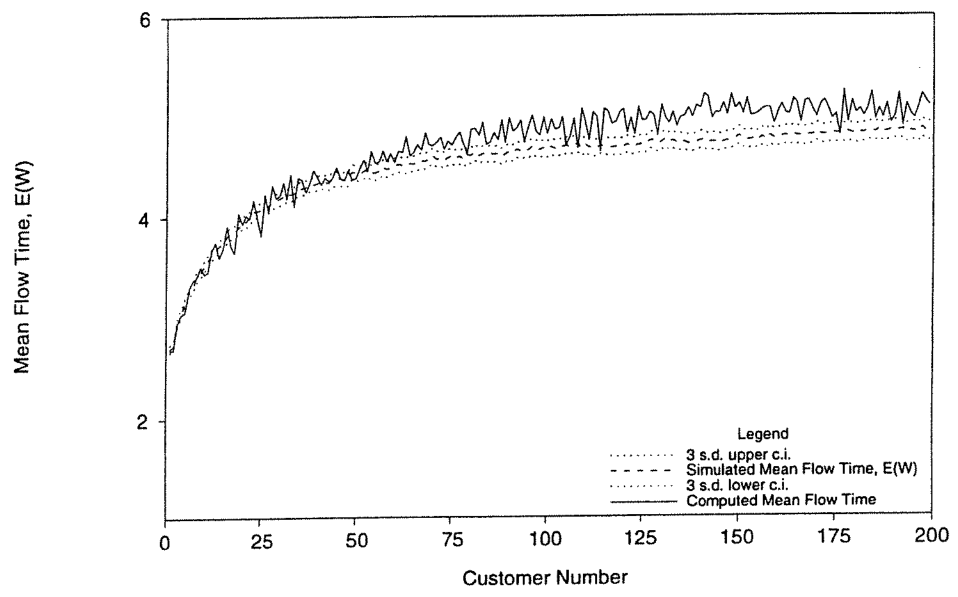

Fig. 5(a). mean flow time in the E2/E4/3 queue (rho = 0.9).

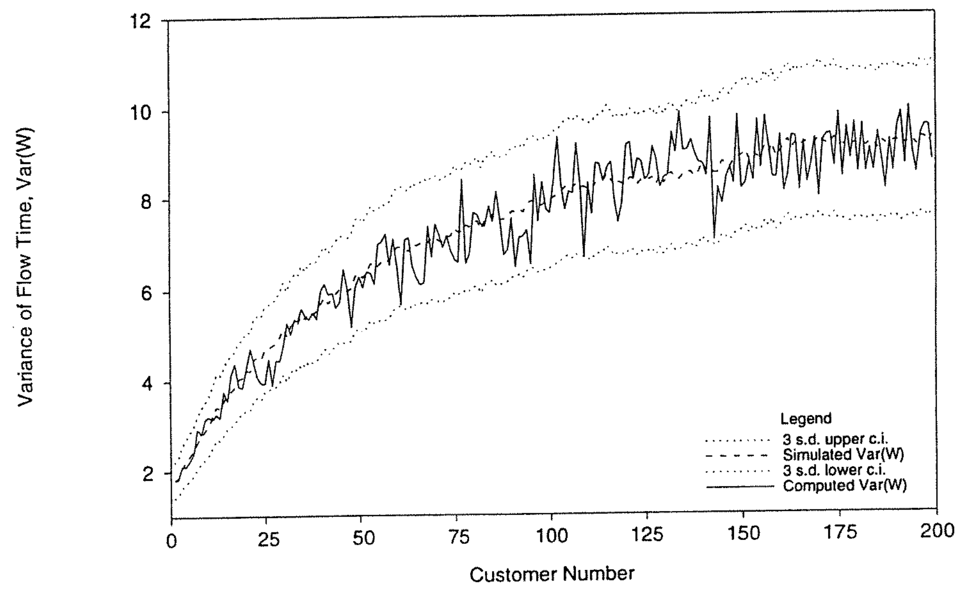

Fig. 5(b). Variance of flow time in the E2/E4/3 queue (rho = 0.9).

Instead of directly trying to improve stability, we are now working towards speeding up the solution to the linear program. To calculate the flow time distributions of 200 customers for the GI/G/2 queue now takes about 5 minutes; and for the GI/G/3 queue about 25 minutes. Almost all of the computational work is in solving the linear programs.

4. Tandem queues

In this section, an outline of the computations in the case of a tandem queue is given. The system consists of poisson arrivals to a single server queue that feeds work to a two-server queue. The FCFS discipline is used at both queues. The inter-arrival time between customers n and $n+1$ to the system is T_n. The service time sequence at the first queue (#1) is $\{S_n^{(1)}, n = 1, 2, \ldots, \}$, where the $S_n^{(1)}$ are i.i.d. random variables. Similarly, the service times sequence at the second queue (#2) is $\{S_n^{(2)}, n = 1, 2, \ldots, \}$. The two sequences are independent of one another. The delay $D_n^{(1)}$ of the nth customer at queue #1 follows the usual recursion:

$$D_{n+1}^{(1)} = (D_n^{(1)} + S_n^{(1)} - T_n)^+; \quad D_0^{(1)} = 0; \quad n = 1, 2, \ldots ,$$

The delay $D_n^{(2)}$ of the nth customer at queue #2 will depend on the idle time at queue #1. The recursion for this quantity can be simplified by writing out the inter-departure times from queue #1. Let $T_n^{(1)}$ be the inter-departure time between the nth and the $(n+1)$st customer from queue #1. Then the delay at queue #2 can be calculated once these quantities are known, by using the GI/G/2 recursion equation. Unfortunately, the $T_n^{(1)}$'s are in general no longer independent of one another. The equation for $T_n^{(1)}$ in terms of "known" quantities is

$$T_{n+1}^{(1)} = \max[S_n^{(1)}, (D_n^{(1)} + S_n^{(1)} - T_n)^-] + S_{n+1}^{(1)}; \quad T_1^{(1)} = S_1^{(1)}; \quad n = 1, 2, \ldots .$$

Let $(R_{n,1}, R_{n,2})$ be the remaining work vector at queue #2, after the nth customer has joined that queue. If the joint distribution of $D_n^{(1)}$ and this vector is known, then the recursion can be carried forward for the entire system. (In general, if the joint distribution of the remaining work at the arrival epochs (to each queue) is known, than the recursion calculations are quite simple.) With this idea in mind, we can "reduce" the entire computational work to the evaluations of two generic integrals. The first integral has to do with the work at queue #1 and is given by

$$I_{T,1}(k, r, l, s, w1) = \int_0^{w1} \frac{x^{k-1}}{(k-1)!} \mathrm{e}^{-rx} \int_0^{\infty} \frac{y^{l-1}}{(l-1)!} \mathrm{e}^{-sy} \, \mathrm{d}y \, \mathrm{d}x + \int_{w1}^{\infty} \frac{x^{k-1}}{(k-1)!} \mathrm{e}^{-rx} \int_0^{\infty} \frac{y^{l-1}}{(l-1)!} \mathrm{e}^{-sy} \, \mathrm{d}y \, \mathrm{d}x ,$$

where $w1$ is the work (a point mass at w1) at queue #1 after the last customer joined that queue. The first integration in each of the repeated integrals above is with

Table 3
Flow times in the tandem queue.

Distributions [a]	Mean W last ten [b]		Mean% abs.	Var(W) last ten [b]		Mean% abs.	Mean% abs. error in W	
First → Second	Simul.	Comput.	error	Simul.	Comput.	error	Queue 1	Queue 2
M→M (rho = 0.5)	4.69 ± 0.10	4.71	2.33	10.24 ± 1.68	10.96	9.63	1.82	3.61
M→M (rho = 0.9)	30.03 ± 0.56	29.98	1.96	346.70 ± 58.76	358.73	5.34	1.31	3.96
M→CS4 (rho = 0.9)	40.35 ± 0.90	41.16	3.07	900.67 ± 135.32	1111.96	25.50	1.35	5.60
M→CS2 (rho = 0.9)	34.05 ± 0.68	34.69	1.99	508.27 ± 81.43	539.97	13.47	1.39	3.91
CS2→CS4 (rho = 0.9)	46.69 ± 0.97	48.02	4.78	1051.82 ± 150.98	1177.37	17.43	1.66	7.68

[a]CS2 and CS4 refer to scv of 2 and 4; and inter-arrival distribution is exponential.
[b]99% confidence intervals are also averaged over the last ten customers.

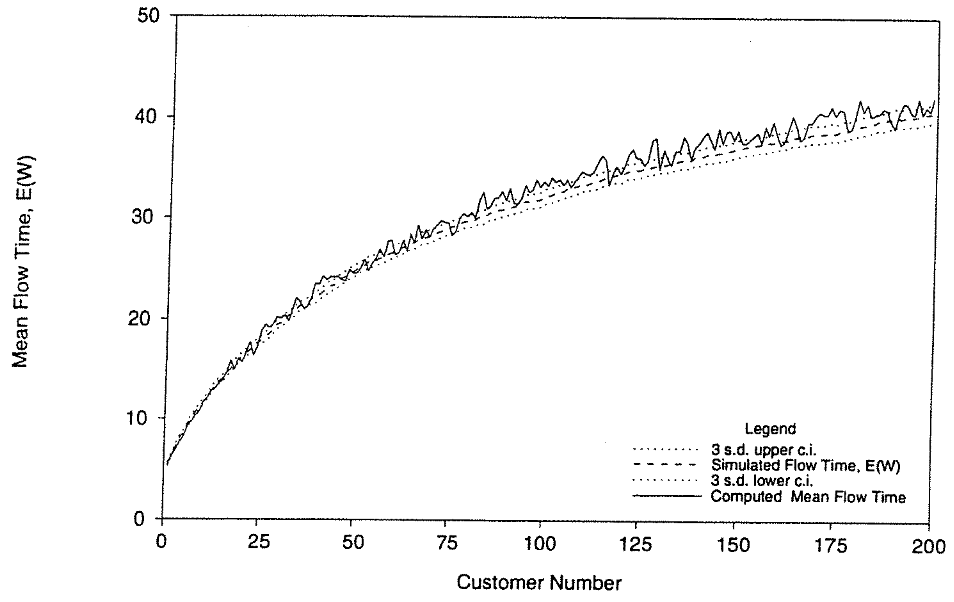

Fig. 6(a). Mean flow time in the CS2/1 → CS4/2 tandem queue (rho = 0.9).

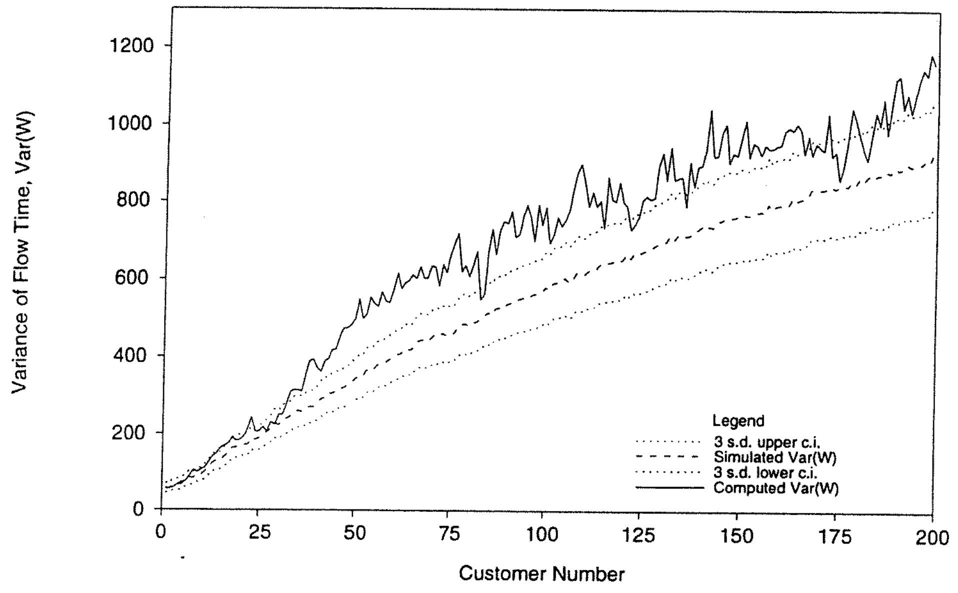

Fig. 6(b). Variance of flow time in the CS2/1 → CS4/2 tandem queue (rho = 0.9).

respect to the inter-arrival distribution. The second integration accounts for the service time at queue #1.

Similarly, if (r_1, r_2) is the work vector at a server in queue #2 after the last customer joined it, then the second generic integral is

$$I_{T,2}(k,r,l,s,r_1) = \int_0^{w1} \frac{x^{k-1}}{(k-1)!} e^{-rx} \int_0^{r_1} \frac{y^{l-1}}{(l-1)!} e^{-sy}\, dy\, dx + \int_{w1}^{r_1+w1} \frac{x^{k-1}}{(k-1)!} e^{-rx} \int_0^{r_1+w1-x} \frac{y^{l-1}}{(l-1)!} e^{-sy}\, dy\, dx,$$

where $w1$ is the remaining work at queue #1 and r_1 represents a component of the work vector at queue #2. The integrators are the same as in $I_{T,1}$. When the inter-arrival time is smaller than the remaining work at queue #1, there is no idle time. Therefore the first part of the expression in $I_{T,2}$. When there is idleness, the inter-arrival time to the second queue is larger by that amount, and so the rest of $I_{T,2}$ follows. (This last expression is cumbersome and involves a binomial expansion when the algebra is written out in full. There is another form of the recursion for the tandem queue; that involves two approximations using the linear program for each recursion. The resulting form is easier to code, but requires more running time on the computer.)

The detailed algorithm for the tandem queue is similar in structure to the GI/G/C routines; therefore only the numerical results are given. Table 3 shows the results for a few test cases. In all the systems tested, the mean absolute error in the expected total flow time through the system, W, was less than 5%. The corresponding errors at the two queues are shown alongside. The error in the expected flow time at the single-server queue was less than 2%, in keeping with the GI/G/1 results given earlier. The contribution of error from the two-server queue, is larger (4–8%). The variance of flow times, Var(W), has a mean error between 9 and 25%. However, the worst case occurs when the service time distribution at queue #2 has an scv of 4, and is shown in fig. 6. The larger values of error are also due to the correction we have made to retain precision (see the discussion in section 3).

Acknowledgements

We are extremely grateful for the discussions and the guidance provided by Prof. J.G. Shanthikumar. He gave us the idea of using a closure approximation. We wish to thank Prof. Prabhu and an anonymous referee for their helpful comments that led to improvements in the exposition of this paper. This research was supported by grants to the University of California, Berkeley by the Semiconductor Research Corporation and the California State MICRO program.

References:

[1] D. Assaf, N.A. Langberg, T.H. Savits and M. Shaked, Multivariate phase-type distributions, Oper. Res. 32 (1984) 688–702.

[2] E.J. Gumbel, Bivariate exponential distributions, J. Amer. Statist. Inst. 55 (1960) 698–707.

[3] M.H. Van Hoorn and L.P. Seelen, Approximations for the GI/G/C queue, J. Appl. Prob. 23 (1986) 484–494.

[4] N.J. Johnson and S. Kotz, *Distributions in Statistics: Continuous Multivariate Distributions* (Wiley, New York, 1972).

[5] J. Keilson and W.R. Nunn, Laguerre transformations as a tool for the numerical solution of integral equations of convolution type, Appl. Math. Comp. 5 (1979) 313–359.

[6] L. Kleinrock, *Queueing Systems: Vol. 1: Theory* (Wiley, New York, 1975).

[7] T.C.T. Kotiah, Approximate transient analysis of some queueing systems, Oper. Res. 28 (1978) 333–346.

[8] E.L. Leese and D.W. Boyd, Numerical methods for determining the transient behavior of queues with variable arrival rates, J. Canad. Oper. Res. Soc. 4 (1966) 1–13.

[9] D.V. Lindley, On the theory of queues with a single server, Proc. Camb. Philo. Soc. 48 (1952) 277–289.

[10] A.W. Marshall and I. Olkin, A generalized bivariate exponential distribution, J. Appl. Prob. 4 (1967) 291–302.

[11] M.F. Neuts, *Matrix-Geometric Solutions in Stochastic Models* (Johns Hopkins University Press, Baltimore, MD, 1981).

[12] G.F. Newell, *Applications of Queueing Theory* (Chapman & Hall, London, 1971).

[13] M.H. Rothkopf and S.S. Oren, A closure approximation for the non-stationary M/M/1 Queue, Management Sci. 25 (1979) 522–534.

[14] L.P. Seelen, H.C. Tijms and M.H. Van Hoorn, *Tables for the Multi-Server Queue* (North-Holland, Amsterdam, 1984).

[15] K.C. Sevcik, A.I. Levy, S.K. Tripathi and J.L. Zahorjan, Improving approximations of aggregated queueing network systems, *Proc. Computer Performance Modelling, Measurement and Evaluation* (1977) pp.1–22.

[16] J.G. Shanthikumar, Bilateral phase-type distributions, Naval Res. Logist Quaterly 32 (1985) 119–136.

[17] J.G. Shanthikumar and U. Sumita, Approximations for the time spent in a dynamic job shop with applications to Due-date Assignment, Int. J. Prod. Res. 26 (1988) 1329–1352.

[18] M.D. Springer, *The Algebra of Random Variables* (Wiley, New York, 1979).

[19] R.W. Wolff, *Stochastic Modelling and the Theory of Queues* (Prentice-Hall, Englewood Cliffs, NJ, 1990).