

PH/PH/1 Queueing Models in Mathematica for Performance Evaluation

Hsing Luh* and Zheng-Zhong Xu

Department of Mathematical Sciences, National ChengChi University, District Wen-Shan, Taipei, Taiwan, R.O.C.

Abstract—A review of queueing applications indicates that many researchers have intelligently adapted its theoretical results to develop an easy and effective analytical tool that can be applied to manufacturing system planning. In particular, the PH/PH/1 distribution has been studied extensively for GI/G/1 queue models. We present *Mathematica* programs that calculate algebraically the probability distribution of the system states from the Matrix-Geometric solution procedures of a PH/PH/1 queue with first-come first-served discipline. The advantage in using *Mathematica* packages (1996) for solving a general queueing problem is also described.

Keywords—Queueing theory, Phase-type distribution, Matrix-geometric solution

1. INTRODUCTION

One of the major factors that affect the efficiency of a manufacturing system is wasted production capacity relating to queues forming at the processing centers, or inadequate service capacity. Indeed, the problem of inadequate service capacity is especially critical in an automated environment. This paper is a report of two subsystems that can exist within an automated factory described and analyzed using queueing models. The goal is to compute its performance measure by *Mathematica*.

There has been a wealth of research results that remark upon using queueing theory to study the performance of automated storage/retrieval system design, automated guided vehicle system design, inventory systems analysis and assembly line balancing. See Buzacott and Shanthikumar (1993). However, the majority of the research on inventory assembly line balancing, material handling, and other systems assumes deterministic task times and employs deterministic mathematical programming techniques. The mathematical modeling approach often becomes cumbersome as system complexity increases and provides less opportunity for studying variations in the governing rules. In contrast, simulation is preferred as it allows a greater variety of performance data to be gathered and remains adaptable for complex system analysis. But, the cost of running an experiment on simulation in general is much higher than the cost spent in a mathematical modeling analysis.

On the other hand, reports on the successful application of basic queueing elements indicate that queueing research has proven to be a valuable tool. Even the simpler single-server queueing models prove to be relevant in some manufacturing subsystems. The computational effort to manufacturing subsystems provide more accurate models than those afforded by deterministic Mathematical models

but with the cost less than that taken by simulation approaches. For this reason, it appears that a worth while endeavor would be to investigate the effects of incorporating queueing considerations into Manufacturing Systems.

2. PROBLEM DESCRIPTION

As an example to illustrate how queueing techniques are applied to a manufacturing working environment, we consider a flow-line system with 4 stages numbered as station 1, 2, 3, 4, respectively. Assume each station i has an exponential processing time with mean $1/\mu_i$, in addition, $\mu_1 = \mu_2$ and $\mu_3 = \mu_4$. Suppose there are two operators who monitor the process of jobs to ensure the quality of work. One operator inspects stations 1 and 2; the other operator inspects stations 3 and 4. Both operators can only monitor one job on one station at any time. All jobs start from station 1 and are completed at station 4. After finishing at station 2, the job is moved forward to stations 3 and 4 in series. It is clear that jobs can be only accumulated before station 3 since at any time each operator can not have more than one job to check. If the raw material for jobs to process is unlimited, then what is a proper buffer capacity designed for station 3? In other words, what is the efficiency production capacity relating to queues forming at station 3?

Since series stations at process could be represented as an Erlang distribution, consider this manufacturing system as a single server queueing model with general service times. Furthermore, neither the interarrival time distribution nor the service time distribution are exponential. It falls into the category of a GI/G/1 queueing model.

A review of queueing applications illustrates that many researchers have intelligently adapted its theoretical results

* Corresponding author's email: slu@nccu.edu.tw

over a decade ago to develop an easy and effective analytical tool that can be applied to Manufacturing System planning. The stationary (steady-state) probability distribution is one of the most common elements characterizing a queueing system. Every standard textbook on queueing theory, for example, Gross and Harris (1985) and Kleinrock (1975), shows the stationary probability distribution solved from the state balance equations transforms and derives the expressions for mean values of the queue size and other performance measures. However, it is very difficult in the sense of methodology and numerical computation to obtain the steady-state probability in a GI/G/1 queue. The reason is easily found in every textbook.

Neuts (1981) has developed the Matrix-Geometric solutions method to solve a particular GI/G/1 queue in which the interarrival and service time distributions are both of phase type, i.e., an exponential distribution at each phase. It is so called the PH/PH/1 queue in such a case. The phase-type distribution has been studied extensively for the last ten years. Particular features of the interarrival or service time distributions may be better exploited in treating one embedded Markov process by discussing the PH/PH/1 queue in the frame work of Quasi-Birth-and-Death (Q.B.D.) process. The main thrust of this paper is to apply *Mathematica* programs to calculate algebraically the probability distribution of the system states from the Matrix-Geometric solution procedures of an PH/PH/1 queue. Thus, it easily characterizes the basic queueing system structures that occur in a manufacturing subsystems, showing the effects of variability in the task times at each station, and queue lengths between indicating preferred buffer sizes.

Queueing techniques that are particularly developed for this type of distributions are presented in next section.

3. MODEL DESCRIPTION

In this section we review a general PH/PH/1 queueing system. Let us first introduce the notation for the PH/PH/1 queueing system presented in this paper.

We denote the interarrival time distribution by $F(\cdot)$; as its representation (α, T) where α is a row vector of dimension m and T is a square matrix of dimension m . The distribution function is given by $F(t) = 1 - \alpha \exp(Tt)e^t$, $t \geq 0$, and its mean is λ . The service time distribution $H(\cdot)$ has mean μ and the representation (β, S) of dimension n . The distribution is given by $H(t) = 1 - \beta \exp(S t)e^t$, $t \geq 0$. Both representation are irreducible, and α and β are initial probabilities so that $\alpha e = 1$ and $\beta e = 1$, where e is a column vector of all ones. The traffic intensity is defined as $\rho := \lambda / \mu$, which is assumed to be less than unity for the stability of the queue.

We consider the stationary PH/PH/1 queue at a departure epoch in which the queue becomes empty. The PH/PH/1 queue may be studied as a QBD process on the state space,

$$E = \{(0, j), 1 \leq j \leq m\} \cup \{(i, j, k), i \geq 1, 1 \leq j \leq m, 1 \leq k \leq n\}.$$

The index $i \geq 1$ denotes the number of customers in the system; the index j , $1 \leq j \leq m$, represents the phase of the PH-renewal process of arrivals, and the index k , $1 \leq k \leq n$, indicates the phase of the service in course. The states are labeled in the lexicographic order, that is, $(0, 1), \dots, (0, m), (1, 1, 1), \dots, (1, 1, n), (1, 2, 1), \dots, (1, 2, n), \dots$. The infinitesimal generator Q is given by

$$Q = \begin{bmatrix} T & T^0 A^0 \otimes \beta & 0 & 0 & \cdots & \cdots \\ I \otimes S^0 & A_1 & A_0 & 0 & \cdots & \cdots \\ & A_2 & A_1 & A_0 & \cdots & \cdots \\ & & A_2 & A_1 & A_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{bmatrix}$$

where

$$A_0 = [T^0 A^0] \otimes I, \tag{1}$$

$$A_1 = T \otimes I + I \otimes S, \tag{2}$$

$$A_2 = I \otimes S^0 B^0, \tag{3}$$

$$T^0 = [-T e, \dots, -T e]_{m \times m}, \tag{4}$$

$$A^0 = \text{diag}[\alpha_1 \cdots \alpha_m], \tag{5}$$

$$B^0 = \text{diag}[\beta_1 \cdots \beta_n], \tag{6}$$

$$\alpha = [\alpha_1 \cdots \alpha_m], \tag{7}$$

$$\beta = [\beta_1 \cdots \beta_n]. \tag{8}$$

Here the tensor product $C \otimes D$ is a formalism that transforms bilinear problems into linear ones; that property is exploited to write the balance equations of the PH/PH/1 system. This is explained in the following. Let C, D be two linear spaces (say over the real line). From a computational point of view, assume C, D to be finitely dimensional, with bases $(c_i)_{1 \leq i \leq m}, (d_j)_{1 \leq j \leq n}$. Then $C \otimes D$ is another linear space with a basis of $C \otimes D$ which is $(c_i \otimes d_j)_{1 \leq i \leq m, 1 \leq j \leq n}$.

Consider $a = \sum_{i=1}^m a_i c_i, d = \sum_{j=1}^n b_j d_j$ then by bilinearity:

$$a \otimes d = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} a_i b_j c_i \otimes d_j$$

where $a \otimes d$ is called the kronecker product of a and d , and is defined as a Mathematica function whose procedure is described in the appendix. The steady-state probabilities in a stable PH/PH/1 queue are matrix-geometrically distributed with geometric generator R . This matrix is usually given as the minimal solution to a quadratic matrix equation.

$$R^2 A_2 + R A_1 + A_0 = 0$$

A straightforward iterative procedure to obtain R is given below:

$$R_0 = A_0 A_1^{-1}, \tag{9}$$

$$R_{k+1} = -A_0 A_1^{-1} - R_k^2 A_2 A_1^{-1}, \quad k = 0, 1, 2, \dots \tag{10}$$

$$\|R_{k+1} - R_k\|_\infty < \varepsilon, \tag{11}$$

where $\|\cdot\|_\infty$ is an infinite norm referred to Horn(1988) and ε is a preset value. The computation of matrix R may involve a high number of iterations. Lucantoni and Ramaswamy (1985) have studied the more general algorithm for solving R efficiently.

The stationary probability vector z of Q is of the form $[z_0, z_1, z_1 R, z_1 R^2, \dots]$. The vector z_0 and z_1 are of dimensions m and $m \times n$ respectively. They are obtained by solving the equations

$$z_0 T + z_1 (I \otimes S^0) = 0, \tag{12}$$

$$z_1 [A_1 + R A_2] + z_0 (T^0 A^0 \otimes \beta) = 0, \tag{13}$$

$$z_0 e + z_1 (I - R)^{-1} e = 1. \tag{14}$$

Matrix operations are intensively taken for solving the stationary probability in this approach. Many results for queues with phase-type interarrival and/or service times are available in the literature, see e.g. Neuts(1981). Calculating the result in matrix product forms is straightforward in principle, because it only involves the evaluation of the basic matrix operations of given matrices. However the expressions for these operations soon become so complicated as the number of phases goes up.

It is clear that sparsity of the matrix T plays a significant role in simplifying these equations further. In contrast, special features of the matrix S only affect the algorithm to a minor extent. In order to solve the problem described in previous section and illustrate the use of particular structure features of T , we shall consider $E_2/E_2/1$ where both the interarrival times and the service times are

2-phase Erlang, which is to be described in the next section.

4. AN $E_2/E_2/1$ QUEUE

Consider an $E_2/E_2/1$ queue. Assume the average arrival rate λ equals to 0.2 and the average service rate μ equals to 1. Their probability distributions are both Erlang with 2 phases. With the notations introduced before, we let

$$T // \text{MatrixForm} = \begin{pmatrix} -0.4 & 0.4 \\ 0 & -0.4 \end{pmatrix}$$

$$\alpha // \text{MatrixForm} = (1 \ 0)$$

and

$$T^0 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 \\ 0.4 & 0.4 \end{pmatrix}.$$

Let A^0 be a diagonal matrix consisting of α , i.e.,

$$A^0 // \text{MatrixForm} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Regarding the service time representation (β, S) , we let

$$S // \text{MatrixForm} = \begin{pmatrix} -2 & 2 \\ 0 & -2 \end{pmatrix}$$

$$\beta // \text{MatrixForm} = (1 \ 0)$$

and

$$S^0 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 \\ 2 & 2 \end{pmatrix} \text{ and } S^1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}.$$

Let

$$B^0 // \text{MatrixForm} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

By equations (1), (2) and (3), we have A_0, A_1 , and A_2 , i.e.,

$$A_0 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.4 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \end{pmatrix}$$

$$A_1 // \text{MatrixForm} = \begin{pmatrix} -2.4 & 2 & 0.4 & 0 \\ 0 & -2.4 & 0 & 0.4 \\ 0 & 0 & -2.4 & 2 \\ 0 & 0 & 0 & -2.4 \end{pmatrix}$$

and

$$A_2 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

To compute R, we first set an initial value of R_0 in equation (9), that is

$$R_0 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.166667 & -0.138889 & -0.0277778 & -0.462963 \\ 0 & -0.16667 & 0 & -0.0277778 \end{pmatrix}$$

By the iterative procedures (9), (10) and (11), we have

$$R_1 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.182742 & 0.152285 & 0.0322431 & 0.05225 \\ 0.0231481 & 0.185957 & 0.00450103 & 0.0347437 \end{pmatrix}$$

$$R_2 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.185992 & 0.154994 & 0.0332738 & 0.0535604 \\ 0.0288166 & 0.190681 & 0.0058087 & 0.0366207 \end{pmatrix}$$

up to

$$R_{20} // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.186877 & 0.155731 & 0.0335698 & 0.0539299 \\ 0.0308365 & 0.192364 & 0.00629946 & 0.0373102 \end{pmatrix}$$

which satisfies equation (11) with ϵ equal to 0.0000001.

Thus we let $k = 20$ and

$$R // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.186877 & 0.155731 & 0.0335698 & 0.0539299 \\ 0.0308365 & 0.192364 & 0.00629946 & 0.0373102 \end{pmatrix}$$

After obtaining R, we shall solve the stationary probabilities denoted by (z_0, z_1) , where z_0 is a 2-dimensional vector corresponding to states (0, 1) and (0, 2), and z_1 four-dimensional corresponding to states (1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2). Let

$$z_0 = \{z_{01}, z_{02}\}$$

and

$$z_1 = \{z_{11}, z_{12}, z_{13}, z_{14}\}$$

which is associated with (1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2), respectively

Let eq1, eq2, and eq3 represent equations (12), (13) and (14). After algebraically computing in *Mathematica*, we have

$$\begin{aligned} \text{eq1} &= z_0.T + z_1.(Id[2] \otimes S) \\ &= \{-0.4 z_{01} + 2. z_{12}, 0.4 z_{01} - 0.4 z_{02} + 2. z_{14}\} \\ \text{eq2} &= z_0.((T_0.A_0) \otimes \beta) + z_1.(A_1 + R.A_2) \\ &= \{0. z_{01} + 0.4 z_{02} - 2.4 z_{11} + 0. z_{12} + 0.311461 z_{13} + 0.384728 z_{14}, 2. z_{11} - 2.4 z_{12} + 0. z_{13} + 0. z_{14}, 0. z_{01} + 0. z_{02} + 0.4 z_{11} + 0. z_{12} - 2.29214 z_{13} + 0.0746204 z_{14}, 0. z_{11} + 0.4 z_{12} + 2. z_{13} - 2.4 z_{14}\} \\ \text{eq3} &= z_0.e[2] + z_1.Inverse[Id[4]-R].e[4] \\ &= \{z_{01} + z_{02} + 1. z_{11} + 1. z_{12} + 1.46068 z_{13} + 1.28016 z_{14}\}, \end{aligned}$$

where $Id[2]$ and $e[4]$ are user-defined functions denoting a 2x2 identity matrix and a vector e of size 4 respectively.

By using a library function `Solve`, we write the following statements to solve (z_0, z_1)

```
Solve[{
eq1[[1]]==0,
eq1[[2]]==0,
eq2[[1]]==0,
eq2[[2]]==0,
eq2[[3]]==0,
eq3[[1]]==1},{z0, z1, z11, z12, z13, z14}].
```

The output is $\{z_{01}=0.341095, z_{02}=0.457141, z_{11}=0.0818627, z_{12}=0.0682189, z_{13}=0.0150414, z_{14}=0.0232093\}$

Let the average number of jobs in the buffer including the one in service be written in the formula as

$$L[np_] := z_1.e[4] + \sum_{n=2}^{np} (n * z_1.R^{n-1}.e[4])$$

Since R approaches to the limit as $k=20$, $L[20]$ will also reaches its limit such that

$$L[20] = \{0.22093\},$$

which explicitly suggests the preferred buffer sizes. The

mean processing time of a job in this system is calculated by

$$\frac{1}{\lambda} + L/\lambda = 5 + 1.10465 = 6.10465.$$

Now, suppose the system is renovated in order to reduce the processing time. As an example, we will consider a flow-line system with 6 stages. Similarly, assume each station i has an exponential processing time with its mean $1/w_i$, moreover, $w_1 = w_2$ and $w_3 = w_4 = w_5 = w_6$. There are two operators who monitor the process of jobs to ensure the quality of work. One operator inspects stations 1 and 2; the other operator inspects stations 3 through 6. After finishing at station 1, this job is assigned to rework at station 2 with probability p , and moved forward to station 3 with probability $1 - p$. Starting from station 3, this job is processed at station 4, 5 and 6 in series. Again, if the raw material for jobs to process is unlimited, then what would be a proper buffer capacity designed for station 3? Furthermore, would this new system yield a mean processing time which is less than that given by the $E_2/E_2/1$ queueing model? We shall apply the PH/PH/1 approach to this case again, though it becomes a $K_2/E_4/1$ queueing model with slightly different arrival process and service time distribution. This will be described in the next section.

5. A $K_2/E_4/1$ QUEUE

Consider a $K_2/E_4/1$ queue. Assume the interarrival time distribution and the service time distribution are both phase type. The service time distribution is Erlang and has 4 phases where each phase has an exponential distribution with rate 4. Thus, the average service rate is 1. The interarrival time distribution is a 2-phase Coxian distribution. Its probability density function is $K_2(t) = \frac{1}{4}e^{-\frac{1}{2}t}$ $t \geq 0$, and its average arrival rate is $\frac{1}{3}$. according to the definition of (α, T) , we have

$$T // \text{MatrixForm} = \begin{pmatrix} -0.5 & 0.25 \\ 0 & -0.5 \end{pmatrix}$$

and

$$\alpha // \text{MatrixForm} = (1 \ 0)$$

By (4) and (5), it computes

$$T^0 // \text{MatrixForm} = \begin{pmatrix} 0.25 & 0.25 \\ 0.5 & 0.5 \end{pmatrix}$$

and

$$A^0 // \text{MatrixForm} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Since the service time is Erlang-4, we have

$$S // \text{MatrixForm} = \begin{pmatrix} -4 & 4 & 0 & 0 \\ 0 & -4 & 4 & 0 \\ 0 & 0 & -4 & 4 \\ 0 & 0 & 0 & -4 \end{pmatrix}$$

and

$$\beta // \text{MatrixForm} = (1 \ 0 \ 0 \ 0)$$

Again by (4) and (5) for the service time distribution, we have

$$S^0 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 4 \end{pmatrix}, S^* = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 4 \end{pmatrix}$$

and

$$B^0 // \text{MatrixForm} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

From (1), we compute

$A_0 // \text{MatrixForm}$

$$= \begin{pmatrix} 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$A_1 // \text{MatrixForm}$

$$= \begin{pmatrix} -4.5 & 4 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & -4.5 & 4 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & -4.5 & 4 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & -4.5 & 4 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & -4.5 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4.5 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4.5 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4.5 \end{pmatrix}$$

and

$$A_2 // \text{MatrixForm} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \end{pmatrix}$$

To compute R, we start from (9) with initial R_0 , i.e. $R_0 = -A_0A_1^{-1} // \text{MatrixForm}$

$$\begin{pmatrix} 0.056 & 0.049 & 0.044 & 0.039 & 0.0031 & 0.0055 & 0.0073 & 0.0087 \\ 0 & 0.056 & 0.049 & 0.044 & 0 & 0.0031 & 0.0055 & 0.0073 \\ 0 & 0 & 0.056 & 0.049 & 0 & 0 & 0.0031 & 0.0055 \\ 0 & 0 & 0 & 0.056 & 0 & 0 & 0 & 0.0031 \\ 0.11 & 0.099 & 0.088 & 0.078 & 0.0062 & 0.011 & 0.015 & 0.017 \\ 0 & 0.11 & 0.099 & 0.088 & 0 & 0.0062 & 0.011 & 0.015 \\ 0 & 0 & 0.11 & 0.099 & 0 & 0 & 0.0062 & 0.011 \\ 0 & 0 & 0 & 0.11 & 0 & 0 & 0 & 0.0062 \end{pmatrix}$$

After the first iteration, we have

$$R_1 // \text{MatrixForm} = \begin{pmatrix} 0.0569088 & 0.0505856 & 0.044965 & 0.0399689 & 0.00322843 & 0.00568003 & 0.00754697 & 0.00892891 \\ 0.00171274 & 0.057078 & 0.050736 & 0.0450987 & 0.000142729 & 0.00329787 & 0.00575011 & 0.00761669 \\ 0.00216769 & 0.00192684 & 0.0572683 & 0.0509052 & 0.000147189 & 0.000237881 & 0.00339302 & 0.00584408 \\ 0.00274348 & 0.00243865 & 0.00216769 & 0.0574824 & 0.000160883 & 0.000278488 & 0.000367972 & 0.00352055 \\ 0.116524 & 0.103577 & 0.0920685 & 0.0818387 & 0.00674088 & 0.0117462 & 0.015556 & 0.0183741 \\ 0.00685097 & 0.117201 & 0.104179 & 0.0926032 & 0.000570915 & 0.00701864 & 0.0120265 & 0.0158348 \\ 0.00867076 & 0.00770735 & 0.117962 & 0.104855 & 0.000588756 & 0.000951524 & 0.00739925 & 0.0124024 \\ 0.0109739 & 0.00975461 & 0.00867076 & 0.118818 & 0.000643533 & 0.00111395 & 0.00147189 & 0.00790937 \end{pmatrix}$$

Then, after 11 iterations, the algorithm converges and results in

$$R_{11} // \text{MatrixForm} = \begin{pmatrix} 0.0569791 & 0.0506481 & 0.0450205 & 0.0400182 & 0.00323659 & 0.00569075 & 0.00755959 & 0.00894287 \\ 0.00181358 & 0.0571676 & 0.0508157 & 0.0451695 & 0.000152576 & 0.0033116 & 0.00576674 & 0.00763541 \\ 0.00231263 & 0.00205567 & 0.0573828 & 0.051007 & 0.000159102 & 0.000255628 & 0.00341516 & 0.00586942 \\ 0.00295208 & 0.00262407 & 0.00233251 & 0.0576289 & 0.000175239 & 0.000301549 & 0.000397628 & 0.00355505 \\ 0.117126 & 0.104112 & 0.0925441 & 0.0822615 & 0.00681103 & 0.0118383 & 0.0156642 & 0.0184938 \\ 0.00772378 & 0.117977 & 0.104868 & 0.0932162 & 0.000656574 & 0.00713788 & 0.0121708 & 0.0159972 \\ 0.00993984 & 0.00883542 & 0.118965 & 0.105747 & 0.000693889 & 0.00110765 & 0.00759373 & 0.0126248 \\ 0.0128253 & 0.0114003 & 0.0101336 & 0.120119 & 0.000772532 & 0.00132005 & 0.00173635 & 0.00821669 \end{pmatrix}$$

It converges in every element in the matrix. So, we have
 $R = R_{11}$

Let

$$z0 = \{z01, z02\}$$

and $z1 = \{z11, z12, z13, z14, z15, z16, z17, z18\}$

By (12), (13) and (14), we let respectively,

$$\text{eq1} = z0.T + z1.(Id[2] \otimes S) \text{ i.e.} \\ \{-0.5 z01 + 4 z14, 0.25 z01 - 0.5 z02 + 4 z18\},$$

$$\text{eq2} = z0.(T0.A0 \otimes \beta) + z1.(A_1 + R.A_2), \text{ i.e.,}$$

$$\{0.25 z01 + 0.5 z02 - 4.33993 z11 + 0.180678 z12 + \\ 0.204028 z13 + 0.230516 z14 + 0.329046 z15 + 0.372865 \\ z16 + 0.422986 z17 + 0.480475 z18, 4. z11 - 4.5 z12 + 0. \\ z13 + 0. z14 + 0. z15 + 0. z16 + 0. z17 + 0. z18, 0. z11 + 4. \\ z12 - 4.5 z13 + 0. z14 + 0. z15 + 0. z16 + 0. z17 + 0. z18, 0. \\ z11 + 0. z12 + 4. z13 - 4.5 z14 + 0. z15 + 0. z16 + 0. z17 \\ + 0. z18, 0. z01 + 0. z02 + 0.285771 z11 + 0.0305416 z12 \\ + 0.0234777 z13 + 0.0142202 z14 - 4.42602 z15 + \\ 0.0639886 z16 + 0.0504992 z17 + 0.0328668 z18, 0. z11 + \\ 0.25 z12 + 0. z13 + 0. z14 + 4. z15 - 4.5 z16 + 0. z17 + 0. \\ z18, 0. z11 + 0. z12 + 0.25 z13 + 0. z14 + 0. z15 + 4. \\ z16 - 4.5 z17 + 0. z18, 0. z11 + 0. z12 + 0. z13 + 0.25 z14 + \\ 0. z15 + 0. z16 + 4. z17 - 4.5 z18\}$$

and

$$\text{eq3} = z0.e[2] + z1.Inverse[Id[8]-R].e[8], \text{ i.e.,}$$

$$\{z01 + z02 + 1.2603 z11 + 1.19875 z12 + 1.1376 z13 + \\ 1.07702 z14 + 1.53582 z15 + 1.41676 z16 + 1.29982 z17 + \\ 1.18578 z18\},$$

where $Id[8]$ and $e[8]$ are user-defined functions denoting a 8x8 identity matrix and a vector e of size 8 respectively. The most beneficial part of using *Mathematica* is to solve those equations algebraically. Using the command `Solve`, we have

$$\text{Solve}\{ \\ \text{eq1}[[1]] == 0, \\ \text{eq1}[[2]] == 0, \\ \text{eq2}[[1]] == 0, \\ \text{eq2}[[2]] == 0, \\ \text{eq2}[[3]] == 0, \\ \text{eq2}[[4]] == 0, \\ \text{eq2}[[5]] == 0, \\ \text{eq2}[[6]] == 0, \\ \text{eq2}[[7]] == 0, \\ \text{eq3}[[1]] == 1\}, \\ \{z01, z02, z11, z12, z13, z14, z15, z16, z17, z18\} \\ \{z01 0.380356, z02 0.300124, z11 0.0676951, \\ z12 0.0601735, z13 0.0534875, z14 0.0475445, \\ z15 0.00556234, z16 0.00828727, z17 0.010338, \\ z18 0.0137433\}$$

It yields the solution corresponding to the stationary probability at each system state as follows:

$$z01 = 0.380355645465741876 \\ z02 = 0.300123862158583998$$

$$z11 = 0.06769511331895815755 \\ z12 = 0.0601734517240724997 \\ z13 = 0.0534875126436200076 \\ z14 = 0.0475444556832177767 \\ z15 = 0.00556233542007625914 \\ z16 = 0.00828726769140514463 \\ z17 = 0.0103379886503390183 \\ \text{and} \\ z18 = 0.0137432549282141303$$

Define the average number of jobs in the buffer including the one in service as

$$L[np_] := z1.e[8] + \sum_{n=2}^{np} (n * z1.R^{n-1}.e[8])$$

Compute the average number of jobs in the buffer, i.e.,
 $L[11] = \{0.366433\}$

In average, a process time for each job is
 $3 + 0.366433/(1/3) = 4.09929$.

6. CONCLUDING REMARKS

This paper discusses and, to some extent, identifies manufacturing sub-systems that can be represented using accepted queueing models. Aggregate performance measures can be obtained through application of these models. Depending on the complexity of the congestion characteristics of the system, the standard results from queueing theory texts may yield guidelines for selection of optimal operational parameters.

In this paper, we show several *Mathematica* functions that greatly simplify the manipulation of symbolic matrices for computing the stationary probability distribution. We first review the formula of the stationary probability distribution in the first-come first-served system the average queue size. Then, we consider two fundamental numerical cases, i.e., $E_2/E_2/1$ and $K_2/E_4/1$. In each case, we start to write the equations for computing the stationary probability distribution. We have also provided a *Mathematica* program that symbolically calculates the quantity and rearranges the terms for better appearance.

REFERENCES

1. Buzacott, John A., and J. George Shanthikumar. (1993). *Stochastic Models of Manufacturing Systems*. Prentice-Hall.
2. Lucantoni, D.M. and Ramaswamy, V. (1985). Efficient algorithms for solving the non-linear matrix equations arising in phase type queues. *Stochastic Models*, 1: 29-52.
3. Gross, D. and Harris, C.M. (1985). *Fundamentals of Queueing Theory 2nd*. John Wiley & Sons.
4. Kleinrock, L. (1975). *Queueing Systems Vol I*. Wiley-interscience.
5. *Mathematica*. Microsoft Windows Version 3.0, Wolfram Research, (1996).
6. Neuts, M.F. (1981). *Matrix-Geometric Solutions in Stochastic Models*. An Algorithmic Approach, Baltimore; Johns Hopkins University Press.

7. Horn, R.A. and Johnson. (1988). *Matrix Analysis*.
 Cambridge.

APPENDICES

(1) Kronecker product

First, we give a small example of a Kronecker product as follows:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix} \otimes \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \begin{bmatrix} e & f \\ g & h \end{bmatrix} & c \begin{bmatrix} e & f \\ g & h \end{bmatrix} \\ b \begin{bmatrix} e & f \\ g & h \end{bmatrix} & d \begin{bmatrix} e & f \\ g & h \end{bmatrix} \end{bmatrix}$$

Symbolically, \otimes is defined by

$(x_) \otimes (y_) := \text{Kronecker}[x, y]$.

where,

$\text{Kronecker}[x_, y_] :=$

$\text{Module}[i, j, k, l, r, s, bM, bN, bP, bQ, m, n, t, temp, Pp,$

$bP = \text{Dimensions}[y][[1]]; bQ = \text{Dimensions}[y][[2]];$

$Pp = \text{Table}[0, i, bM*bP, j, bN*bQ];$

$\text{For}[i = 1, i <= bM, i++,$

$\text{For}[j = 1, j <= bN, j++,$

$\text{For}[k = 1, k <= bP, k++,$

$\text{For}[l = 1, l <= bQ, l++,$

$r = (i - 1)*bP + k;$

$s = (j - 1)*bQ + l;$

$\text{temp} = Pp[[r]];$

$\text{temp}[[s]] = x[[i]][[j]]*y[[k]][[l]];$

$Pp[[r]] = \text{temp};$

$]$

$]$

$]$

$];$

$\text{Kronecker}[x, y] = Pp;$

$\text{Print}[x//\text{MatrixForm}, " ",$

$y//\text{MatrixForm}, "=", Pp//\text{MatrixForm}]$

$]$

(2) $e[n_] := \text{Table}[1, i, 1, n, j, 1, 1]$

(3) $d[n_] := \text{IdentityMatrix}[n]$