SIAM J. Matrix Anal. Appl.
Vol. 18, No. 2, pp. 464–481, April 1997

CIRCULANT PRECONDITIONERS FOR MARKOV-MODULATED POISSON PROCESSES AND THEIR APPLICATIONS TO MANUFACTURING SYSTEMS[*]

WAI KI CHING[†], RAYMOND H. CHAN[‡], AND XUN YU ZHOU[§]

Abstract. The Markov-modulated Poisson process (MMPP) is a generalization of the Poisson process and is commonly used in modeling the input process of communication systems such as data traffic systems and ATM networks. In this paper, we give fast algorithms for solving queueing systems and manufacturing systems with MMPP inputs. We consider queueing systems where the input of the queues is a superposition of the MMPP which is still an MMPP. The generator matrices of these processes are tridiagonal block matrices with each diagonal block being a sum of tensor products of matrices. We are interested in finding the steady state probability distributions of these processes which are the normalized null vectors of their generator matrices. Classical iterative methods, such as the block Gauss–Seidel method, are usually employed to solve for the steady state probability distributions. They are easy to implement, but their convergence rates are slow in general. The number of iterations required for convergence increases like $O(m)$, where m is the size of the waiting spaces in the queues. Here, we propose to use the preconditioned conjugate gradient method. We construct our preconditioners by taking circulant approximations of the tensor blocks of the generator matrices. We show that the number of iterations required for convergence increases at most like $O(\log_2 m)$ for large m. Numerical results are given to illustrate the fast convergence.

As an application, we apply the MMPP to model unreliable manufacturing systems. The production process consists of multiple parallel machines which produce one type of product. Each machine has exponentially distributed up time, down time, and processing time for one unit of product. The interarrival of a demand is exponentially distributed and finite backlog is allowed. We consider hedging point policy as the production control. The average running cost of the system can be written in terms of the steady state probability distribution. Our numerical algorithm developed for the queueing systems can be applied to obtain the steady state distribution for the system and hence the optimal hedging point. Furthermore, our method can be generalized to handle the case when the machines have a more general type of repairing process distribution such as the Erlangian distribution.

Key words. Markov-modulated Poisson process, preconditioned conjugate gradient squared method, manufacturing systems, hedging point policy

AMS subject classifications. 65C20, 65F10

PII. S0895479895293442

1. Introduction.

The Markov-modulated Poisson process (MMPP) is a generalization of the Poisson process and is widely used as the input model of communication systems such as data traffic systems [8] and ATM networks [23]. An MMPP is a Poisson process whose instantaneous rate is itself a stationary random process which varies according to an irreducible n-state Markov chain. If n is 1, then the process is just a Poisson process. We say that the MMPP is in phase k, $1 \le k \le n$, when the underlying Markov process is in state k, and in this case the arrivals occur according

* Received by the editors October 20, 1995; accepted for publication (in revised form) by D. P. O'Leary June 11, 1996.
http://www.siam.org/journals/simax/18-2/29344.html

† Department of Applied Mathematics, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong and Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, Hong Kong (wkching@se.cuhk.edu.hk).

‡ Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong (rchan@math.cuhk.edu.hk). This research was supported by HKRGC grant CUHK 316/94E.

§ Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, Hong Kong (xyzhou@se.cuhk.edu.hk). This research was supported by RGC Earmarked grant CUHK 489/95E and RGC Earmarked grant CUHK 249/94E.

to a Poisson process of rate λ_k. The process is characterized by the generator matrix Q of the underlying Markov process and the rates $\lambda_1, \lambda_2, \ldots, \lambda_n$.

In this paper, we first discuss a numerical algorithm for solving the steady state probability distributions of queueing systems with MMPP inputs. We then relate queueing systems with MMPP inputs to the production process in unreliable manufacturing systems under the hedging point production control. Our algorithm can be applied to solve for the steady state probability distribution of these systems and hence their optimal hedging points.

We consider a queueing system with $(q+1)$ trunks, where each trunk has m waiting spaces and s multiple exponential servers. The analysis of these queueing systems can be used to determine call congestions in teletraffic networks with alternate routing; see Meier-Hellstern [13]. A call will overflow to other trunks if its first destination trunk is full and will be blocked from the system if all the trunks are full. The analysis of these queueing systems can be decomposed into the study of each trunk independently; see Meier-Hellstern [13]. For each trunk, the overflow from other trunks is modeled by a 2^q-state MMPP which is a superposition of q independent 2-state MMPPs; i.e., each trunk is an (MMPP/M/$s/s+m$) queue. The generator matrices of these processes are $(s+m+1)2^q \times (s+m+1)2^q$ tridiagonal block matrices with each diagonal block being a sum of tensor products of matrices. We are interested in finding the steady state probability distributions of the queues which are the normalized null vectors of the generator matrices.

Usually classical iterative methods, such as the block Gauss–Seidel method, are used to solve for the steady state probability distribution. They are easy to implement, but their convergence rates are slow in general; see the numerical results in section 7. Here, we propose to use the preconditioned conjugate gradient (PCG) method. Our preconditioners are constructed by taking circulant approximations of the tensor blocks of the generator matrix. We prove that the preconditioned system has singular values clustered around 1 independent of the size of the waiting spaces m. Hence the conjugate gradient method will converge very fast when employed to solve the preconditioned system for large m. In fact, we prove that the number of iterations required for convergence grows at most like $O(\log_2 m)$. Numerical examples are given in section 7 to illustrate the fast convergence. For the case of a single server ($s = 1$), our generator matrix corresponds to a class of quasi-birth–death (QBD) processes which can be solved efficiently by the folding algorithm; see Ye and Li [22]. We will compare the complexity of our PCG method with that of the folding algorithm in section 7. The cost of our PCG method increases more slowly than that of the folding algorithm when the problem size increases. In fact, for large values of q ($q \geq 6$), the PCG-type method is more efficient than the folding algorithm.

The analysis of the MMPP queueing systems can be applied to the production planning of manufacturing systems. We consider manufacturing systems of multiple parallel machines producing one type of product. Usually positive inventory is stored to hedge against uncertain situations such as the breakdown of machines and the shortfall of products; see Akella and Kumar [1]. It is well known that the hedging point policy is optimal for one-machine manufacturing systems in some simple situations; see [1, 3, 10, 11]. For two-machine flowshops, hedging policies are no longer optimal but near optimal; see [16, 15]. A hedging point policy is characterized by a number h: the machines keep producing the product at the maximum possible production rate if the inventory level is less than h, maintain the inventory level h as far as they can if the inventory level reaches h, and stop producing if the inventory level exceeds h.

When the optimal policy is a zero-inventory policy (i.e., the hedging point is zero), then the policy matches with the just-in-time (JIT) policy. The JIT policies have strongly been favored in real-life production systems for process discipline reasons even when they are not optimal. By using the JIT policy, the Toyota company has managed to reduce work-in-process and cycle time in the presence of the stochastic situations mentioned above; see Monden [14]. We focus on finding optimal hedging point policies for the manufacturing systems.

We note that in [1, 3, 10, 11] only one-machine systems are considered, and, in addition, the repairing process of the machine is assumed to be exponentially distributed. Ching and Zhou [6] consider one-machine manufacturing systems with the repairing process being Erlangian distributed. The algorithm proposed here can deal with the more general case of multiple machines. Each machine is unreliable and has exponential up time and down time, and the demand is a Poisson process. The production process of the machines can be modeled as an MMPP. The generator matrix for the machine-inventory system is a particular case of the queueing systems discussed above, with the queue size m being the size of the inventory which in practice can easily go up to the thousands. Our numerical method developed for the queueing networks above is well suited for solving the steady state probability distribution for these processes. Given a hedging point, the average running cost of the machine-inventory system can be written in terms of the steady state probability distribution. Hence the optimal hedging point can also be obtained. Moreover, our algorithm can also handle the case when the repair time has a more general distribution, e.g., the Erlangian distribution.

The outline of the paper is as follows. In section 2, we present the generator matrix for the queueing system (MMPP/M/s/$s+m$). In section 3, we construct preconditioners by taking circulant approximations of the tensor blocks of the generator matrices. In section 4, we prove that the preconditioned systems have singular values clustered around 1. The cost count of our method is given in section 5. In section 6, we apply our method to the production planning of manufacturing systems with multiple parallel machines. Numerical examples are given in section 7 to illustrate the fast convergence rate of our method. Finally, concluding remarks are given in section 8.

2. The queueing system.

In this section, we present the queueing system (MMPP/M/s/$s+m$) arising in telecommunication networks; see, for instance, Meier-Hellstern [13]. In order to construct the generator matrix of the queueing process, we first define the following queueing parameters:

(i) $1/\lambda$, the mean arrival time of the exogenously originating calls,
(ii) $1/\mu$, the mean service time of each server,
(iii) s, the number of servers,
(iv) m, the number of waiting spaces in the queue,
(v) q, the number of overflow queues, and
(vi) (Q_j, Λ_j), $1 \le j \le q$, the parameters of the MMPP's modeling overflow parcels, where

$$Q_j = \begin{pmatrix} \sigma_{j1} & -\sigma_{j2} \\ -\sigma_{j1} & \sigma_{j2} \end{pmatrix} \quad \text{and} \quad \Lambda_j = \begin{pmatrix} \lambda_j & 0 \\ 0 & 0 \end{pmatrix}. \tag{1}$$

Here σ_{j1}, σ_{j2}, and λ_j, $1 \le j \le q$, are positive MMPP parameters. Conventionally, an infinitesimal generator Q has nonnegative off-diagonal entries and zero row sums.

For ease of presentation, in our discussion all the infinitesimal generators are of the form $-Q^t$, which has nonpositive off-diagonal entries and zero column sums.

The input of the queue comes from the superposition of several independent MMPPs, which is still an MMPP and is parametrized by two $2^q \times 2^q$ matrices (Q, Γ). Here

$$Q = (Q_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes Q_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes Q_q), \tag{2}$$

$$\Lambda = (\Lambda_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes \Lambda_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes \Lambda_q), \tag{3}$$

and

$$\Gamma = \Lambda + \lambda I_{2^q},$$

where I_2 and I_{2^q} are the 2×2 and $2^q \times 2^q$ identity matrices, respectively, and $\otimes$ denotes the Kronecker tensor product. In the following, we will drop the subscript of the identity matrix I if the dimension of the matrix is clear from the context.

We can regard our (MMPP/M/$s/s+m$) queue as a Markov process on the state space

$$\{(i, j) \mid 0 \le i \le s+m, 1 \le j \le 2^q\}.$$

The number i corresponds to the number of calls at the destination, while j corresponds to the state of the Markov process with generator matrix Q. Hence the generator matrix of the queueing process is given by the following $(s+m+1)2^q \times (s+m+1)2^q$ tridiagonal block matrix A:

$$A = \begin{pmatrix} Q+\Gamma & -\mu I & & & & & 0 \\ -\Gamma & Q+\Gamma+\mu I & -2\mu I & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\Gamma & Q+\Gamma+s\mu I & -s\mu I & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -\Gamma & Q+\Gamma+s\mu I & -s\mu I \\ 0 & & & & & -\Gamma & Q+s\mu I \end{pmatrix}. \tag{4}$$

For simplicity, let us write $n = (s+m+1)2^q$. The steady state probability distribution vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)^t$ is the solution to the matrix equation $A\mathbf{p} = \mathbf{0}$ with constraints

$$\sum_{i=1}^{n} p_i = 1$$

and

$$p_i \ge 0 \quad \text{for all } 1 \le i \le n.$$

Note that the matrix A is irreducible and has zero column sums, positive diagonal entries, and nonpositive off-diagonal entries. From Perron–Frobenius theory, the matrix A has a one-dimensional null space with a positive null vector; see Varga [20, p. 30]. Therefore, the steady state probability distribution vector $\mathbf{p}$ exists.

Many useful quantities such as the steady state distribution of the number of calls at the destination, the blocking probability, and the waiting time distribution can be obtained from the vector $\mathbf{p}$; see Meier-Hellstern [13]. We note that $\mathbf{p}$ can be obtained by normalizing the solution $\mathbf{x}$ of the nonsingular system

$$G\mathbf{x} \equiv (A + \mathbf{e}_n\mathbf{e}_n^t)\mathbf{x} = \mathbf{e}_n. \tag{5}$$

Here $\mathbf{e}_n = (0, \ldots, 0, 1)^t$ is an n-vector. The matrix G is nonsingular because it is irreducible diagonally dominant with the last column being strictly diagonally dominant. We will solve the linear system (5) by conjugate gradient (CG)-type methods; see [2, 18]. The convergence rate of CG-type methods depends on the distribution of the singular values of the matrix G. The more clustered the singular values of G are, the faster the convergence rate will be; see Axelsson and Barker [2].

However, this is not the case for our matrix G, and we will see in the numerical results in section 7 that the convergence for system (5) is very slow. To speed up the convergence, a preconditioner is used. In essence, we solve instead of (5) the preconditioned system

$$GC^{-1}\mathbf{w} = \mathbf{e}_n \tag{6}$$

for $\mathbf{w}$ by CG-type methods. Obviously, the solution $\mathbf{x}$ to (5) is given by $C^{-1}\mathbf{w}$. A good preconditioner C is an easy-to-construct matrix, the preconditioned matrix GC^{-1} has singular values clustered around 1, and the preconditioned system $C\mathbf{y} = \mathbf{r}$ can be solved easily for any vector $\mathbf{r}$; see Axelsson and Barker [2]. We will show that our preconditioner satisfies these three criteria in the next three sections.

3. Construction of our preconditioners.

In this section, we discuss the construction of preconditioners for the linear system (6). Our preconditioner C is constructed by exploiting the block structure of the generator matrix A in (4). Notice that the generator A can be written as the sum of tensor products:

$$A = I \otimes Q + B \otimes I + R \otimes \Lambda, \tag{7}$$

where B and R are $(s + m + 1) \times (s + m + 1)$ matrices given by

$$B = \begin{pmatrix} \lambda & -\mu & & & & & 0 \\ -\lambda & \lambda+\mu & -2\mu & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\lambda & \lambda + s\mu & -s\mu & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -\lambda & \lambda + s\mu & -s\mu \\ 0 & & & & & -\lambda & s\mu \end{pmatrix}$$

and

$$R = \begin{pmatrix} 1 & & & & 0 \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & 1 & \\ 0 & & & -1 & 0 \end{pmatrix}.$$

For small s, we observe that B and R are close to the tridiagonal Toeplitz matrices

$$\text{tridiag}[-\lambda, \lambda + s\mu, -s\mu] \qquad \text{and} \qquad \text{tridiag}[-1, 1, 0],$$

respectively. Our preconditioner is then obtained by taking the "circulant approximation" of the matrices B and R, which are defined by $c(B)$ and $c(R)$ as follows:

$$c(B) = \begin{pmatrix} \lambda + s\mu & -s\mu & & & -\lambda \\ -\lambda & \lambda + s\mu & -s\mu & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda & \lambda + s\mu & -s\mu \\ -s\mu & & & -\lambda & \lambda + s\mu \end{pmatrix} \tag{8}$$

and

$$c(R) = \begin{pmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & 1 & \\ 0 & & & -1 & 1 \end{pmatrix}. \tag{9}$$

We note that $c(B)$ and $c(R)$ are Strang's circulant approximations of the Toeplitz matrices tridiag$[-\lambda, \lambda+s\mu, -s\mu]$ and tridiag$[-1, 1, 0]$, respectively; see Chan [5]. Clearly, we have the following lemma.

LEMMA 1. $\text{rank}(B - c(B)) = s + 1$ *and* $\text{rank}(R - c(R)) = 1$.

Using the theory of circulant matrices (see Davis [7]) we also have the following.

LEMMA 2. *The matrices $c(B)$ and $c(R)$ can be diagonalized by the discrete Fourier transform matrix F; i.e.,*

$$F^*(c(B))F = \Phi \quad \text{and} \quad F^*(c(R))F = \Psi,$$

where both Φ and Ψ are diagonal matrices. The eigenvalues of $c(B)$ and $c(R)$ are given by

$$\phi_j = \lambda(1 - e^{\frac{-2\pi i(j-1)}{s+m+1}}) + s\mu(1 - e^{\frac{-2\pi i(j-1)(s+m)}{s+m+1}}), \quad j = 1, \ldots, s + m + 1 \tag{10}$$

and

$$\psi_j = 1 - e^{\frac{-2\pi i(j-1)}{s+m+1}}, \quad j = 1, \ldots, s + m + 1. \tag{11}$$

Thus, the matrices $c(B)$ and $c(R)$ can be inverted easily by using fast Fourier transforms.

We first approximate our matrix A in (7) (and hence G in (5)) by

$$D = I \otimes Q + c(B) \otimes I + c(R) \otimes \Lambda. \tag{12}$$

We observe that D is irreducible and has zero column sums, positive diagonal entries, and nonpositive off-diagonal entries. Hence D is singular and has a null space of dimension one. Moreover, D is unitarily similar to a diagonal block matrix:

$$(F^* \otimes I)D(F \otimes I) = I \otimes Q + \Phi \otimes I + \Psi \otimes \Lambda = \text{diag}(D_1, C_2, C_3, \ldots, C_{s+m+1}). \tag{13}$$

Here the blocks are

$$C_i = Q + \phi_i I + \psi_i \Lambda, \qquad i = 2, \ldots, s+m+1, \tag{14}$$

and $D_1 = Q$ with D_1 being the only singular block.

Let

$$C_1 = Q + \mathbf{e}_{2^q}\mathbf{e}_{2^q}^t, \tag{15}$$

where $\mathbf{e}_{2^q} = (0, \ldots, 0, 1)^t$ is a 2^q-vector. Since C_1 is irreducible diagonally dominant with the last column being strictly diagonally dominant, it is nonsingular. Our preconditioner C for the matrix G in (5) is defined as

$$C = (F \otimes I)\mathrm{diag}(C_1, C_2, \ldots, C_{s+m+1})(F^* \otimes I), \tag{16}$$

which is clearly nonsingular.

4. Convergence analysis.

In this section, we study the convergence rate of our algorithm when m, the number of waiting spaces, is large. In the queueing systems considered in Meier-Hellstern [13], the number of waiting spaces m in each queue is much larger than the number of overflow queues q. In section 6, we apply the MMPP to model manufacturing systems of q parallel machines and m possible inventory states. In practice, the number of possible inventory states is much larger than the number of machines in the manufacturing systems and can easily go up to thousands.

We prove that if the queueing parameters λ, μ, s, q, and σ_{ij} are fixed independent of m, then the preconditioned system GC^{-1} in (6) has singular values clustered around 1 as m tends to infinity. Hence when CG-type methods are applied to solve the preconditioned system (6), we expect fast convergence. Numerical examples are given in section 7 to demonstrate our claim. We start the proof by the following lemma.

Lemma 3. *We have* $\mathrm{rank}(G - C) \le (s+2)2^q + 2$.

Proof. We note that by (5) we have $\mathrm{rank}(G-A) = 1$. From (7), (12), and Lemma 1, we see that $\mathrm{rank}(A-D) = (s+2)2^q$. From (13), (15), and (16), we see that D and C differ by a rank-one matrix. Therefore, we have

$$\mathrm{rank}(G-C) \le \mathrm{rank}(G-A) + \mathrm{rank}(A-D) + \mathrm{rank}(D-C) = (s+2)2^q + 2.$$

Hence the inequality is proved. ∎

Theorem 1. *The preconditioned matrix* GC^{-1} *has at most* $2((s+2)2^q + 2)$ *singular values not equal to* 1.

Proof. We first note that

$$GC^{-1} = I + (G-C)C^{-1} \equiv I + L_1,$$

where $\mathrm{rank}(L_1) \le (s+2)2^q + 2$ by Lemma 3. Therefore,

$$C^{-*}G^*GC^{-1} - I = L_1^*(I + L_1) + L_1$$

is a matrix of rank at most $2((s+2)2^q + 2)$. ∎

Thus the number of singular values of GC^{-1} that are distinct from 1 is a constant independent of m. In order to show fast convergence of PCG-type methods with preconditioner C, one still needs an estimate of $\sigma_{\min}(GC^{-1})$, the smallest singular value of GC^{-1}. If $\sigma_{\min}(GC^{-1})$ is uniformly bounded away from zero independent of m, then the method converges in $O(1)$ iterations; if $\sigma_{\min}(GC^{-1})$ decreases like

$O(m^{-\alpha})$ for some $\alpha > 0$, then the method converges in at most $O(\log_2 m)$ steps; see Van der Vorst [19] or Chan [4, Lemma 3.8.1].

In the remainder of this section, we show that even in the worst case in which $\sigma_{\min}(GC^{-1})$ decreases in an order faster than $O(m^{-\alpha})$ for any $\alpha > 0$ (e.g., like $O(e^{-m})$), we can still have a fast convergence rate. Note that in this case the matrix equation (6) is very ill conditioned. Our trick is to consider a regularized equation of (6) as follows:

$$C^{-*}(G^*G + m^{-4-\beta}I)C^{-1}\mathbf{w} = C^{-*}G^*\mathbf{e}_n, \tag{17}$$

where β is any positive constant.

In the following, we prove that the regularized preconditioned matrix

$$C^{-*}(G^*G + m^{-4-\beta}I)C^{-1}$$

has eigenvalues clustered around 1 and its smallest eigenvalues decrease at a rate no faster than $O(m^{-4-\beta})$. Hence PCG-type methods will converge in at most $O(\log_2 m)$ steps when applied to solve the preconditioned linear system (17). Moreover, we prove that the 2-norm of the error introduced by the regularization tends to zero at a rate of $O(m^{-\beta})$. To prove our claim, we must get an estimate of the upper and lower bounds for $||C^{-1}||_2$. We begin our proof by the following lemma.

LEMMA 4. *Given any matrix W, if the smallest eigenvalue of $W + W^*$, denoted by $\lambda_{\min}(W + W^*)$, satisfies $\lambda_{\min}(W + W^*) \geq \delta > 0$, then $||W^{-1}||_2 \leq 2/\delta$.*

Proof. For any arbitrary $\mathbf{x}$, using the Cauchy–Schwarz inequality, we have

$$\delta||\mathbf{x}||_2^2 \leq \lambda_{\min}(W + W^*)||\mathbf{x}||_2^2 \leq \mathbf{x}^*(W + W^*)\mathbf{x} = 2\mathbf{x}^*W\mathbf{x} \leq 2||\mathbf{x}||_2||W\mathbf{x}||_2.$$

Since $W\mathbf{x}$ is arbitrary, this implies $||W^{-1}||_2 \leq 2/\delta$. ☐

Now we are ready to estimate $||C^{-1}||_2$.

LEMMA 5. *Let the queueing parameters λ, μ, s, q, and σ_{ij} be independent of m. Then there exist positive constants τ_1 and τ_2 independent of m such that*

$$\tau_1 \leq ||C^{-1}||_2 \leq \tau_2 m^2.$$

Proof. We first prove the left-hand side of the inequality. From (16), we see that C is unitarily similar to a diagonal block matrix. We therefore have

$$||C||_2 = \max\{||C_1||_2, ||C_2||_2, \ldots, ||C_{s+m+1}||_2\}.$$

Using (14), (10), and (11), it is straightforward to check that $||C_i||_1$ and $||C_i||_\infty$, $1 \leq i \leq s + m + 1$, are all bounded above by

$$\frac{1}{\tau_1} \equiv q\left(\max_j\{\sigma_{j1}\} + \max_j\{\sigma_{j2}\}\right) + 2(\lambda + s\mu + 1).$$

Using the inequality

$$||\cdot||_2 \leq \sqrt{||\cdot||_1||\cdot||_\infty},$$

we see that $||C_i||_2$, $i = 1, \ldots, s + m + 1$, are all bounded above by $1/\tau_1$. Thus $||C||_2 \leq 1/\tau_1$ and hence $\tau_1 \leq ||C^{-1}||_2$.

Next we prove the right-hand side of the inequality. We note again by (16) that

$$||C^{-1}||_2 = \max\left\{||C_1^{-1}||_2, ||C_2^{-1}||_2, \ldots, ||C_{s+m+1}^{-1}||_2\right\}.$$

From (15), we can see that C_1 is a $2^q \times 2^q$ nonsingular matrix with entries independent of m. Thus $||C_1^{-1}||_2$ is bounded independent of m. To obtain bounds for $||C_i^{-1}||_2$, $i = 2, \ldots, s+m+1$, we first symmetrize the matrices. Define $\Sigma = \Sigma_1 \otimes \cdots \otimes \Sigma_q$, where

$$\Sigma_j = \begin{pmatrix} 1 & 0 \\ 0 & \frac{\sigma_{j1}}{\sigma_{j2}} \end{pmatrix}, \qquad j = 1, \ldots, q.$$

We see that $||\Sigma||_2$ and $||\Sigma^{-1}||_2$ are bounded independent of m. By (1) and (2), we see that $Q\Sigma$ is a symmetric semidefinite matrix. Thus

$$C_i\Sigma = Q\Sigma + \phi_i\Sigma + \psi_i\Lambda\Sigma, \qquad i = 2, \ldots, s+m+1,$$

are symmetric matrices too. By (11), we see that $(\psi_i\Lambda\Sigma + (\psi_i\Lambda\Sigma)^*)$, $i = 2, \ldots, s+m+1$, are diagonal positive semidefinite matrices. Therefore,

$$\lambda_{\min}(C_i\Sigma + (C_i\Sigma)^*) \geq \lambda_{\min}(\phi_i\Sigma + (\phi_i\Sigma)^*), \qquad i = 2, \ldots, s+m+1. \tag{18}$$

From (10), we have

$$\lambda_{\min}(\phi_i\Sigma + (\phi_i\Sigma)^*) \geq \lambda||\Sigma^{-1}||_2^{-1} \sin^2\left(\frac{(i-1)\pi}{s+m+1}\right), \qquad i = 2, \ldots, s+m+1.$$

Since

$$\sin\theta \geq \min\left\{\frac{2\theta}{\pi}, 2\left(1 - \frac{\theta}{\pi}\right)\right\} \quad \forall \theta \in [0, \pi],$$

we have

$$\begin{aligned}\lambda_{\min}(\phi_i\Sigma + (\phi_i\Sigma)^*) &\geq \lambda||\Sigma^{-1}||_2^{-1} \min\left\{\frac{4(i-1)^2}{(s+m+1)^2}, 4\left(1 - \frac{i-1}{s+m+1}\right)^2\right\} \\ &\geq \frac{4\lambda}{m^2}||\Sigma^{-1}||_2^{-1}, \qquad i = 2, \ldots, s+m+1.\end{aligned}$$

By Weyl's theorem [9, p. 181], we then have

$$\lambda_{\min}(\phi_i\Sigma + (\phi_i\Sigma)^*) \geq \frac{\tau}{m^2}, \quad i = 2, \ldots, s+m+1,$$

where $\tau = 4\lambda||\Sigma^{-1}||_2^{-1}$ is a positive constant independent of m.

Thus by (18) we get

$$\lambda_{\min}(C_i\Sigma + (C_i\Sigma)^*) \geq \frac{\tau}{m^2}, \quad i = 2, \ldots, s+m+1.$$

Hence by Lemma 4 we have

$$||\Sigma^{-1}C_i^{-1}||_2 \leq \frac{2}{\tau}m^2, \qquad i = 2, \ldots, s+m+1.$$

Therefore,

$$||C_i^{-1}||_2 \leq ||\Sigma||_2||\Sigma^{-1}C_i^{-1}||_2 \leq \frac{2m^2}{\tau}||\Sigma||_2, \qquad i = 2, \ldots, s+m+1.$$

Since $||C_1^{-1}||_2$ is bounded above independent of m, we have

$$||C^{-1}||_2 \le \max\left\{||C_1^{-1}||_2, \frac{2m^2}{\tau}||\Sigma||_2\right\} \equiv \tau_2 m^2,$$

where τ_2 is a positive constant independent of m. Hence we have proved the lemma. ∎

THEOREM 2. *Let the queueing parameters λ, μ, s, q, and σ_{ij} be independent of m. Then for any positive β the regularized preconditioned matrix*

$$C^{-*}(G^*G + m^{-4-\beta}I)C^{-1} \tag{19}$$

has eigenvalues clustered around 1 *and the smallest eigenvalue decreases at a rate no faster than $O(m^{-4-\beta})$. Furthermore, the error introduced by the regularization is of the order $O(m^{-\beta})$.*

Proof. We note by Theorem 1 that

$$C^{-*}(G^*G + m^{-4-\beta}I)C^{-1} = I + L_2 + m^{-4-\beta}C^{-*}C^{-1},$$

where L_2 is a Hermitian matrix with $\mathrm{rank}(L_2) \le 2((s+2)2^q + 2)$. By Lemma 5, we have

$$\lim_{m\to\infty} m^{-4-\beta}||C^{-*}C^{-1}||_2 \le \lim_{m\to\infty} m^{-\beta} = 0.$$

Thus by Cauchy's interlace theorem [9, p. 184] the regularized preconditioned matrix in (19) has eigenvalues clustered around 1 as m tends to infinity. The error introduced by the regularization is given by $m^{-4-\beta}||C^{-*}C^{-1}||_2$, which by Lemma 5 tends to zero like $O(m^{-\beta})$.

As for the smallest eigenvalue of the regularized preconditioned matrix in (19), we note that

$$\min_{||\mathbf{x}||_2=1} \frac{\mathbf{x}^*(G^*G + m^{-4-\beta})\mathbf{x}}{\mathbf{x}^*C^*C\mathbf{x}} \ge \frac{\min_{||\mathbf{x}||_2=1}\mathbf{x}^*(G^*G + m^{-4-\beta})\mathbf{x}}{\max_{||\mathbf{x}||_2=1}\mathbf{x}^*C^*C\mathbf{x}} \ge \frac{\tau_1}{m^{4+\beta}}, \tag{20}$$

where the rightmost inequality follows from Lemma 5. We recall that τ_1 and β are positive constants independent of m. Hence the smallest eigenvalue of the regularized preconditioned matrix in (19) decreases no faster than $O(m^{-4-\beta})$. ∎

Thus we conclude that PCG-type methods applied to (17) with $\beta > 0$ will converge in at most $O(\log_2 m)$ steps; see Van der Vorst [19] or Chan [4, Lemma 3.8.1]. To minimize the error introduced by the regularization, one can choose a large β. Recall that regularization is required only when the smallest singular value of the matrix GC^{-1} in (6) tends to zero faster than $O(m^{-\alpha})$ for any $\alpha > 0$. In view of Lemma 5 (or cf. (20)), this can happen only when the smallest singular value of G has the same decaying rate. This will imply that the matrix G is very ill conditioned. We note, however, that in all our numerical tests in section 7 we found that there is no need to add the regularization.

5. Cost analysis.

In this section, we derive the computational cost of the PCG-type method. We compare our PCG method with the block Gauss–Seidel (BGS) method used in Meier-Hellstern [13] and the folding algorithm of Ye and Li [22]. We show that the cost for PCG-type algorithms is $O(2^q(s+m+1)\log_2(s+m+1) + q(s+m+1)2^q)$. The computational cost per iteration of the BGS method is

$O((s+m+1)2^{2q})$; see Meier-Hellstern [13]. Thus PCG-type methods require an extra $O(\log_2(s+m+1))$ of work per iteration compared with the BGS method. However, as we will soon see in the numerical examples of section 7, the fast convergence of our method can more than compensate for this minor overhead in each iteration.

When the queue has a single server, i.e., $s = 1$, our generator matrix A corresponds to a class of QBD processes which can be solved efficiently by the folding algorithm of Ye and Li [22]. The complexity of the folding algorithm is approximately $\frac{26\alpha}{3}2^{3q}\log_2(s+m+1)+3(s+m+1)2^{2q}$ operations, where $1 \le \alpha < 2$. We will compare the computational cost of our PCG method with the folding algorithm in section 7. Our PCG method is more efficient than the folding algorithm for large problems.

In PCG-type algorithms for (6), the main cost per iteration is to compute the matrix-vector multiplication of the form $GC^{-1}\mathbf{y}$ twice for some vector $\mathbf{y}$. By using the block tensor structure of A in (7), the multiplication of $G\mathbf{z}$ requires $(s+m+1)q2^q$ operations for any vector $\mathbf{z}$. By (16), we see that $C^{-1}\mathbf{y}$ is given by

$$(F \otimes I)\text{diag}(C_1^{-1}, C_2^{-1}, \ldots, C_{s+m+1}^{-1})(F^* \otimes I)\mathbf{y}.$$

It involves the matrix-vector multiplications of the form

$$(F^* \otimes I)\mathbf{z} \qquad \text{and} \qquad (F \otimes I)\mathbf{z}.$$

By using fast Fourier transforms, they can be obtained in $6(s+m+1)2^q\log_2(s+m+1)$ operations. The vector

$$\text{diag}(C_1^{-1}, C_2^{-1}, \ldots, C_{s+m+1}^{-1})\mathbf{z}$$

can be obtained by solving $(s+m+1)$ linear systems involving the matrices C_i, $i = 1, \ldots, s+m+1$. Since each matrix is of size $2^q \times 2^q$, if GE is used, $O((s+m+1)2^{3q})$ operations will be required. We now show that it can be reduced to $O((s+m+1)q2^q)$ operations.

First we recall from the definitions of C_i, Q, and Λ in (14), (2), and (3) that

$$\begin{aligned} C_i &= ((Q_1 + \phi_i I + \psi_i\Lambda_1) \otimes I \otimes \cdots \otimes I) + (I \otimes (Q_2 + \phi_i I + \psi_i\Lambda_2) \otimes I \otimes \cdots \otimes I) \\ &\quad + \cdots + (I \otimes I \otimes \cdots \otimes (Q_q + \phi_i I + \psi_i\Lambda_q)), \end{aligned} \tag{21}$$

where Q_j and Λ_j, $j = 1, \ldots, q$, are given in (1). By using Schur's triangularization theorem [9, p. 79], we can find 2×2 unitary matrices U_{ij} and lower triangular matrices L_{ij} such that

$$U_{ij}^*(Q_j + \phi_i I + \psi_i\Lambda_j)U_{ij} = L_{ij}, \qquad 1 \le i \le s+m+1, \quad 1 \le j \le q. \tag{22}$$

For $i = 1, \ldots, s+m+1$, define

$$U_i \equiv U_{i1} \otimes \cdots \otimes U_{iq}$$

and

$$L_i \equiv (L_{i1} \otimes I \otimes \cdots \otimes I) + (I \otimes L_{i2} \otimes I \otimes \cdots \otimes I) + \cdots + (I \otimes I \otimes \cdots \otimes I \otimes L_{iq}).$$

We see from (21) and (22) that

$$U_i^* C_i U_i = L_i, \qquad 1 \le i \le s+m+1.$$

Hence the vector $C_i^{-1}\mathbf{w}$ can be computed as $U_i L_i^{-1} U_i^* \mathbf{w}$.

The matrix-vector multiplication of the form $U_j\mathbf{w}$ and $U_j^*\mathbf{w}$ can be done in $2(q2^q)$ operations by making use of the formula

$$U_j\mathbf{w} = (U_{1j} \otimes I \otimes \cdots \otimes I)(I \otimes U_{2j} \otimes I \otimes \cdots \otimes I) \cdots (I \otimes I \otimes \cdots \otimes I \otimes U_{qj})\mathbf{w}.$$

We note that the matrix L_i is a lower triangular matrix and each row of it has at most q nonzero entries. Hence $L_i^{-1}\mathbf{w}$ can be obtained in $q2^q$ operations. Thus for any vector $\mathbf{w}$ the vector $C_i^{-1}\mathbf{w}$ can be obtained in $3(q2^q)$ operations. Hence we conclude that the vector

$$\mathrm{diag}(C_1^{-1}, C_2^{-1}, \ldots, C_{s+m+1}^{-1})\mathbf{r}$$

can be computed in approximately $3(s+m+1)q2^q$ operations.

In summary, each iteration of PCG-type methods needs $2(6(s+m+1)2^q \log_2(s+m+1)+4(s+m+1)q2^q) \approx O(m\log_2 m)$ operations, as compared to $O((s+m+1)2^{2q}) \approx O(m)$ operations required by the BGS method. As we proved in section 4, PCG-type methods will converge in at most $O(\log_2 m)$ steps (see also the numerical results in section 7); therefore, the total complexity of our methods will be $O(m\log_2^2 m)$. As a comparison, the numerical results in section 7 show that the number of iterations required for convergence for the BGS method increases linearly like $O(m)$. Therefore, the total complexity of the BGS method is about $O(m^2)$ operations.

As for storage, PCG-type methods, the folding algorithm, and the BGS method require $O(2^q(s+m+1))$ memory. Clearly, at least $O(2^q(s+m+1))$ memory is required to store the approximated solution in each iteration.

6. The failure-prone manufacturing systems.

In this section, we study a general kind of failure-prone manufacturing system. These systems consist of q multiple parallel machines producing one type of product. Each machine is subject to random breakdowns and repairs. The processing time for one unit of product, the up time, and the down time of each machine are exponentially distributed. The interarrival time of a demand is exponentially distributed. The systems allow finite backlog and a penalty cost is associated with the rejection of a demand. Moreover, there is an inventory cost for holding each unit of product and a shortfall cost for each unit of backlog.

The hedging point policy has been shown to be optimal for one-machine one-product manufacturing systems with repair time exponentially distributed; see [1, 3, 10, 11]. In those works, the discrete inventory levels of the product are approximated by a continuous fluid flow model. Analytic optimal control is found to be threshold (hedging point)-type by solving a pair of Hamilton–Jacobi–Bellman equations. The control is optimal in the sense that it minimizes the average (or discounted) running cost of the manufacturing systems. In this paper, we focus on finding the optimal hedging point for the manufacturing systems under consideration.

It should be noted that in [1, 3, 10, 11, 21, 17] only one machine is considered and the machine has only two states—up and down. Here we consider q parallel unreliable machines. The production process of the machines is then an MMPP. The states of the machines and the inventory level can be modeled as an irreducible continuous time Markov chain. For different values of the hedging point h, the average running cost $C(h)$ can be written in terms of the steady state distribution of the Markov chain. Therefore, the optimal hedging point can be obtained by varying different values of h. Let us first define the following parameters for the manufacturing systems as follows (see Ching and Zhou [6]):

(i) q, the number of machines,
(ii) $1/\sigma_{j1}$, the mean up time of the machine j, $j = 1, \ldots, q$,
(iii) $1/\sigma_{j2}$, the mean repair time for the machine j, $j = 1, \ldots, q$,
(iv) $1/\lambda_j$, the mean processing time for one unit of product on machine j, $j = 1, \ldots, q$,
(v) $1/\mu$, the mean interarrival time of demand,
(vi) h, the hedging point, and
(vii) g, the maximum allowable backlog.

For each machine j, $j = 1, \ldots, q$, let Q_j be the generator matrix of the machine states and Λ_j be the corresponding production rate matrix. Here

$$Q_j = \begin{pmatrix} \sigma_{j1} & -\sigma_{j2} \\ -\sigma_{j1} & \sigma_{j2} \end{pmatrix} \qquad \text{and} \qquad \Lambda_j = \begin{pmatrix} \lambda_j & 0 \\ 0 & 0 \end{pmatrix}$$

(cf. (1)). Each machine has two states—either "up" or "down." Since there are q machines, there are 2^q states for the system of machine. We denote the set of machine states by Ω. The superposition of the q machines forms an MMPP and is characterized by the following $2^q \times 2^q$ generator matrix:

$$Q = (Q_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes Q_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes Q_q)$$

(cf. (2)). The corresponding production rate matrix is given by

$$\Lambda = (\Lambda_1 \otimes I_2 \otimes \cdots \otimes I_2) + (I_2 \otimes \Lambda_2 \otimes I_2 \otimes \cdots \otimes I_2) + \cdots + (I_2 \otimes \cdots \otimes I_2 \otimes \Lambda_q)$$

(cf. (3)).

We let $\alpha(t)$ be the state of the system of machines at time t. Therefore, $\alpha(t)$ has 2^q possible states. The inventory level takes integer value in $[-g, h]$ because we allow maximum backlog of g and the hedging point is h. Here negative inventory means backlog. We let $x(t)$ be the inventory level at time t. The machine-inventory process $\{(\alpha(t), x(t)), t \geq 0\}$ forms an irreducible continuous time Markov chain in the state space

$$\{(\alpha, x) \mid \alpha \in \Omega,\ x = -g, \ldots, 0, \ldots, h\}.$$

Each time it visits a state the process stays there for a random period of time that has an exponential distribution and is independent of the past behavior of the process. If we order the state spaces of the machine-inventory process lexicographically, we get the following $(h+g+1)2^q \times (h+g+1)2^q$ generator matrix H for the machine-inventory system:

$$H = \begin{pmatrix} Q+\Lambda & -\mu I & & & & & 0 \\ -\Lambda & Q+\Lambda+\mu I & -\mu I & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\Lambda & Q+\Lambda+\mu I & -\mu I & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -\Lambda & Q+\Lambda+\mu I & -\mu I \\ 0 & & & & & -\Lambda & Q+\mu I \end{pmatrix},$$

where I is the $2^m \times 2^m$ identity matrix. Clearly, the matrix H has the same tensor block structure as that of the generator matrix A in (4). In fact, H is a particular case of A with $s = 1$, $\lambda = 0$, and $m = h + g - 1$. Therefore, the techniques and algorithms developed in the previous sections can be used to obtain the steady state distribution of the process efficiently. Numerical results are given in section 7 to illustrate the fast convergence.

Important quantities such as the average running cost of the machine-inventory system can be written in terms of its steady state distribution. Let

$$p(\alpha, x) = \lim_{t\to\infty} \text{Prob}\,\{\alpha(t) = \alpha, x(t) = x\}$$

be the steady state probability distribution, and let

$$p_j = \sum_{k\in\Omega} p(k, j), \quad j = -g, -(g-1), \ldots, 0, \ldots, h$$

be the steady state distribution of the inventory level of the system. The average running cost for the machine-inventory system is then given by

$$C(h) = c_I \sum_{j=1}^{h} jp_j - c_B \sum_{j=-g}^{-1} jp_j + c_P \mu p_{-g}, \qquad 0 \le h \le b, \tag{23}$$

where c_I is the inventory cost per unit of product, c_B is the backlog cost per unit of product, c_P is the penalty cost for rejecting an arrival demand, and b is the maximum inventory capacity; see Ching and Zhou [6]. Hence once p_j are given, we can easily find h^* which minimizes the average running cost function $C(h)$ by evaluating $C(h)$ for all $0 \le h \le b$.

We remark that our method can be generalized to handle the case in which each machine has the Erlangian distribution of l phases. Suppose the mean times of repair for machine $j, j = 1, \ldots, q$, are the same in each phase and are equal to $1/\sigma_{j2}$. In this case, the generator matrix for the machine-inventory system can be obtained by replacing the generator matrix of the machine state and its corresponding production rate matrix by $\bar{Q}_j$ and $\bar{\Lambda}_j$, respectively, where

$$\bar{Q}_j = \begin{pmatrix} \sigma_{j1} & & & & -\sigma_{j2} \\ -\sigma_{j1} & \sigma_{j2} & & & \\ & -\sigma_{j2} & \sigma_{j2} & & \\ & & \ddots & \ddots & \\ 0 & & & -\sigma_{j2} & \sigma_{j2} \end{pmatrix} \quad \text{and} \quad \bar{\Lambda}_j = \begin{pmatrix} \lambda_i & & & 0 \\ & 0 & & \\ & & 0 & \\ & & & \ddots & \\ 0 & & & & 0 \end{pmatrix}.$$

Hence we see that the techniques and algorithms developed previously can be applied to this case too.

7. Numerical results.

In this section, we illustrate the fast convergence rate of our method by examples in queueing systems and manufacturing systems. The conjugate gradient squared (CGS) method (see Sonneveld [18]) is used to solve the preconditioned system (6). The method does not require the transpose of the iteration matrix GC^{-1}. Using the folding algorithm, one can obtain the steady state probability vector with a residual error of order 10^{-13} to 10^{-16}; see Ye and Li [22]. In order to compare our method with the folding algorithm, the stopping criterion for the CGS and BGS methods is set to be

$$||A\mathbf{p}_k||_2 \le 10^{-12},$$

where $\mathbf{p}_k$ is the computed steady state probability distribution at the kth iteration and

$$||(y_1, y_2, \ldots, y_n)^t||_2 \equiv \sqrt{\sum_{i=1}^{n} y_i^2}.$$

In all our numerical examples, the residual errors lie between 10^{-13} to 10^{-16}, which is comparable to the folding algorithm; see Ye and Li [22]. The initial guess for both methods is the vector of all ones normalized such that its l_2-norm is equal to 1. All the computations were done on an HP 712/80 workstation with MATLAB.

Let us first give the numerical results for the queueing networks. We compare the numerical results of the CGS, preconditioned CGS, and BGS methods for the number of overflow queues $q = 1, 2, 3, 4$ and the number of servers $s = 2$. The MMPP parameters are arbitrarily chosen to be $\sigma_{j1} = 2/3, \sigma_{j2} = 1/3, j = 1, \ldots, q$. The other queueing parameters are given by $\mu = 2$, $\lambda = 1$, and $\lambda_j = 1/q, j = 1, \ldots, q$. We recall that the size of the matrix is $(s+m+1)2^q \times (s+m+1)2^q$. The number of iterations required for convergence is given in Table 1. The symbols I, C, and BGS represent the methods used, namely, CGS without preconditioner, CGS with our preconditioner C in (16), and the block Gauss–Seidel method, respectively. Numbers of iterations greater than 2000 are signified by "$**$."

TABLE 1
Number of iterations for convergence.

$s = 2$	$q = 1$			$q = 2$			$q = 3$			$q = 4$		
m	I	C	BGS	I	C	BGS	I	C	BGS	I	C	BGS
16	36	7	130	36	9	112	38	12	107	40	13	110
32	155	8	171	154	9	143	158	12	145	161	13	137
64	$**$	7	242	$**$	9	207	$**$	12	213	$**$	13	199
128	$**$	8	366	$**$	10	325	$**$	12	340	$**$	14	317
256	$**$	8	601	$**$	10	549	$**$	12	582	$**$	14	530
512	$**$	8	1051	$**$	10	988	$**$	12	1046	$**$	14	958
1024	$**$	8	$**$	$**$	10	$**$	$**$	12	$**$	$**$	14	$**$

We see that the numbers are roughly constant independent of m for the CGS method with our preconditioner C. For the BGS method, the convergence rate is approximately linear in m. Recall from section 5 that the costs per iteration of the CGS method with preconditioning and of the BGS method are, respectively, $O(2^q(s+m+1)\log_2(s+m+1))$ and $O(2^{2q}(s+m+1))$ operations. We conclude that the total cost of obtaining the steady state probability distribution vector for the CGS method with preconditioning is approximately $O(2^q(s+m+1)\log_2(s+m+1))$ operations while for the BGS method it is approximately $O(2^{2q}m(s+m+1))$ operations.

We next compare the flop counts between our PCG method and the folding algorithm for the single server case ($s = 1$). For simplicity, we set $(s+m+1) = 2^q$ and we consider $q = 1, 2, \ldots, 7$. Our PCG method converges within 25 iterations for all the numerical examples tested. We recall that the number of operations in each iteration of PCG is

$$2\{6(s+m+1)2^q \log_2(s+m+1) + 4(s+m+1)q2^q\}.$$

Therefore, the total number of operations is at most

$$50\{6(s+m+1)2^q \log_2(s+m+1) + 4(s+m+1)q2^q\}.$$

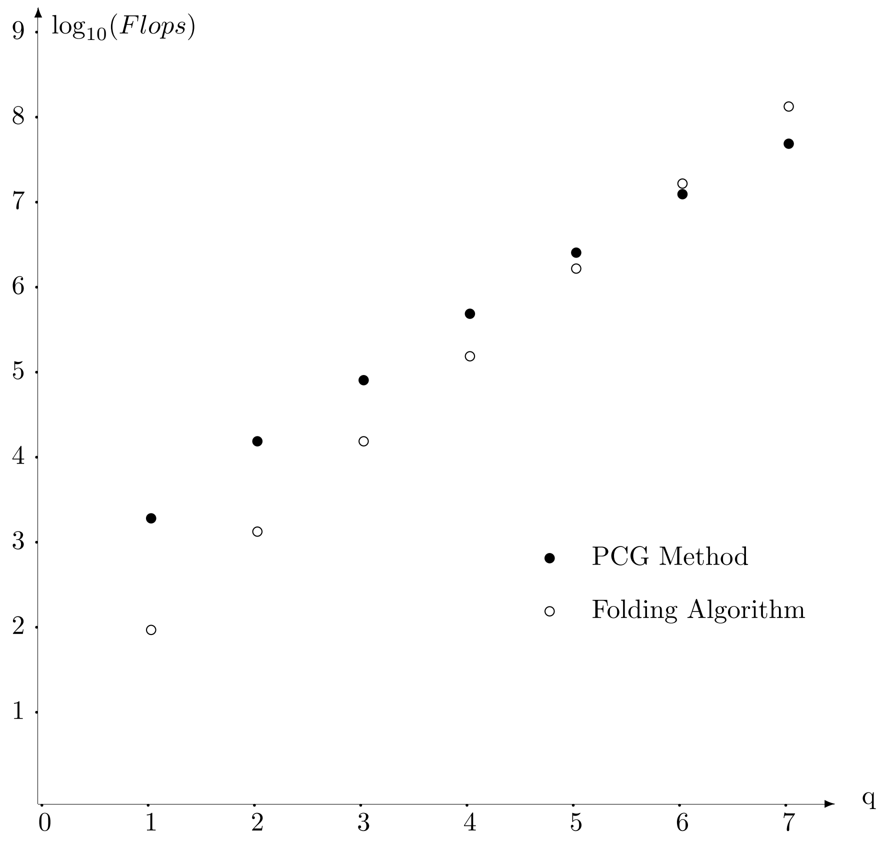

FIG. 1. *Computational flops of the PCG method and the folding algorithm for the single server case.*

The minimum cost of the folding algorithm is given by

$$\frac{26}{3}2^{3q}\log_2(s+m+1)+3(s+m+1)2^{2q};$$

see Ye and Li [22]. In Figure 1, we depict the computational costs of our PCG method and the folding algorithm for different values of q. We see that the computational cost of our PCG method increases at a slower rate than that of the folding algorithm. The crossover point is at $q=6$.

Next we test our algorithm for the failure-prone manufacturing systems. We assume that all q machines are identical, and in each month (four weeks) each machine breaks down once on average. The mean repairing time for a machine is one week. Therefore, we have $\sigma_{j1}=1/3, \sigma_{j2}=1, j=1,\ldots,q$. The mean time for the arrival of demand is $1/5$ week and the mean time for the machine system to produce one unit of product is one day; therefore, we have $\mu=5$ and $\lambda_j=7/q, j=1,\ldots,q$.

In Table 2, we give the number of iterations required for convergence for all three methods. As in the queueing systems case, we see also that the numbers are roughly constant independent of $(g+h)$ for the CGS method with our preconditioner C. For the BGS method, the convergence rate is again approximately linear in $(g+h)$.

Finally, we consider examples of finding the optimal hedging point h^*. We keep the values of the machine parameters the same as in the manufacturing system example above, except that we set $q=4$ and $g=50$. Moreover, the inventory cost c_I

TABLE 2
Number of iterations for convergence.

	$q=1$			$q=2$			$q=3$			$q=4$		
$g+h$	I	C	BGS	I	C	BGS	I	C	BGS	I	C	BGS
16	52	6	565	54	7	603	60	8	601	63	9	685
32	173	6	1491	177	8	1682	231	8	1443	180	10	1904
64	**	6	**	**	8	**	**	9	**	**	10	**
128	**	8	**	**	8	**	**	9	**	**	10	**
256	**	8	**	**	9	**	**	9	**	**	10	**
512	**	8	**	**	9	**	**	9	**	**	11	**
1024	**	8	**	**	9	**	**	9	**	**	11	**

TABLE 3
The optimal $(h^*, C(h^*))$ for different λ_i and μ.

	$\mu=1$	$\mu=2$	$\mu=3$
$\lambda_i=1$	(3,181)	(10,533)	(200,14549)
$\lambda_i=1.5$	(2,128)	(5,270)	(11,576)

and backlog cost c_B per unit of product are 50 and 2000, respectively; the maximum inventory capacity b is 200; and the penalty cost c_P for rejecting a demand is 20000 (see (23)). In Table 3, we give the optimal pair of values $(h^*, C(h^*))$, the optimal hedging point h^*, and its corresponding average running cost per week $C(h^*)$ for different values of λ_i and μ.

8. Concluding remarks.

In this paper, we proposed a fast algorithm for solving the steady state probability distribution for queueing systems with MMPP inputs. The MMPP is commonly used in modeling the inputs of many physical systems; see Heffes and Lucantoni [8] and Meier-Hellstern [13], for instance. Here we related the MMPP to the production process of unreliable manufacturing systems under the hedging point production control. Our algorithm derived for the queueing systems can be applied to obtain the optimal hedging point. Numerical examples were reported to illustrate the fast convergence rate of our algorithm.

For the manufacturing systems, there are two possible generalizations of the model. The maximum allowable backlog g and the number of machines q (with an associated cost) can be considered as decision variables for the optimization problem. We can also consider the machine failure rate σ_{j1} to be dependent on the production rate λ_j. Note that in this case it has been shown that the optimal policy is still of hedging point type if σ_{j1} is a linear function of the production rate λ_j in the one-machine case; see Hu, Vakili, and Yu [12]. It would be interesting to extend our method to these two cases.

Acknowledgment. The authors would like to thank the referees for their constructive comments and helpful suggestions in revising the paper.

REFERENCES

[1] R. Akella and P. Kumar, *Optimal control of production rate in a failure prone manufacturing system*, IEEE Trans. Automat. Control, 31 (1986), pp. 116–126.
[2] O. Axelsson and V. Barker, *Finite Element Solution of Boundary Value Problems, Theory and Computation*, Academic Press, New York, 1984.
[3] T. Bielecki and P. Kumar, *Optimality of zero-inventory policies for unreliable manufacturing systems*, Oper. Res., 36 (1988), pp. 532–541.

[4] R. Chan, *Iterative methods for overflow queueing models* I, Numer. Math., 51 (1987), pp. 143–180.

[5] R. Chan, *Circulant preconditioners for Hermitian Toeplitz systems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 542–550.

[6] W. Ching and X. Zhou, *Matrix methods for production planning in failure prone manufacturing systems*, Lecture Notes in Control and Inform. Sci. 214, Springer-Verlag, London, 1996, pp. 2–30.

[7] P. Davis, *Circulant Matrices*, John Wiley, New York, 1979.

[8] H. Heffes and D. Lucantoni, *A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance*, IEEE J. Select. Areas Commun., SAC-4 (1986), pp. 856–868.

[9] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, London, 1985.

[10] J. Hu and D. Xiang, *The queueing equivalence to optimal control of a manufacturing system with failures*, IEEE Trans. Automat. Control, 38 (1993), pp. 499–502.

[11] J. Hu, *Production rate control for failure prone production systems with no backlog permitted*, IEEE Trans. Automat. Control, 40 (1995), pp. 291–295.

[12] J. Hu, P. Vakili, and G. Yu, *Optimality of hedging point policies in the production control of failure prone manufacturing systems*, IEEE Trans. Automat. Control, 39 (1994), pp. 1875–1880.

[13] K. Meier-Hellstern, *The analysis of a queue arising in overflow models*, IEEE Trans. Commun., 37 (1989), pp. 367–372.

[14] Y. Monden, *Toyota Production System*, Industrial Eng. Manufacturing Press, Atlanta, GA, 1983.

[15] C. Samaratunga, S. Sethi, and X. Zhou, *Computational evaluation of hierarchical production control policies for stochastic manufacturing systems*, Oper. Res., 45 (1997), to appear.

[16] S. Sethi, H. Yan, Q. Zhang, and X. Zhou, *Feedback production planning in a stochastic two-machine flowshop: Asymptotic analysis and computational results*, Internat. J. Production Econ., 30–31 (1993), pp. 79–93.

[17] S. Sethi, Q. Zhang, and X. Zhou, *Hierarchical controls in stochastic manufacturing systems with machines in tandem*, Stochastics Stochastics Rep., 41 (1992), pp. 89–118.

[18] P. Sonneveld, *CGS, a fast Lanczos-type solver for non-symmetric linear systems*, SIAM J. Sci. Comput., 10 (1989), pp. 36–52.

[19] H. Van der Vorst, *Preconditioning by Incomplete Decomposition*, Ph.D. thesis, Rijksuniversiteit te Utrecht, The Netherlands, 1982.

[20] R. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1963.

[21] H. Yan, X. Zhou, and G. Yin, *Finding optimal number of kanbans in a manufacturing system via stochastic approximation and perturbation analysis*, in Lecture Notes in Control and Information Sciences, G. Cohen and J.-P. Quadrat, eds., Springer-Verlag, London, 1994, pp. 572–578.

[22] J. Ye and S. Li, *Analysis of multi-media traffic queues with finite buffer and overload control, Part* I: *Algorithm*, Proc. IEEE INFOCOM '91, (1991), pp. 1464–1474.

[23] H. Young, B. Byung, and K. Chong, *Performance analysis of leaky-bucket bandwidth enforcement strategy for bursty traffics in an ATM network*, Comput. Net. ISDN Syst., 25 (1992), pp. 295–303.