An efficient computational method for solving nonlinear matrix equation and its application in queuing analysis

ARTICLE *in* JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY · MAY 1996

Impact Factor: 0.67 · DOI: 10.1007/BF02943134 · Source: DBLP

3 AUTHORS, INCLUDING:

Jun Steed Huang
Jiangsu University
117 PUBLICATIONS **65** CITATIONS

An Efficient Computational Method for Solving Nonlinear Matrix Equation and Its Application in Queuing Analysis

HUANG Jun (黄 骏)†, ZHU Tao (朱 涛), Jeremiah F. HAYES

Department of Electrical & Computer Engineering
Concordia University, 1455 de Maisonneuve Blvd. W.
Montreal, Quebec, H3G 1M8, Canada
†e-mail: huang@ece.concordia.ca

Received July, 1995.

Abstract

The matrix analytic analysis of queues with complex arrival, vacation and service characteristics requires the solution of nonlinear matrix equation. The complexity and large dimensionality of the model require an efficient and smart algorithm for the solution. In this paper, we propose an efficient Adaptive Newton-Kantorovich (ANK) method for speeding up the algorithm solving the nonlinear matrix equation which is an inevitable step in the analysis of the queue with embedded Markov chain such as BMAP/SMSP/1/∞ queue or its discrete version. BMAP/SMSP/1/∞ is a queuing model with a Semi Markov Service time Process (SMSP) and a Batch Markovian Arrival Process (BMAP). The numerical result is presented for the discrete case of N-MMBP/D/1 queue which arises in analyzing traffic aspect of computer communication network, where MMBP is Markov Modulated Bermoulli Process. The comparisons of Adaptive Newton-Kantorovich (ANK) with Modified Newton-Kantorovich (MNK) show that ANK saves 30% of CPU time when the number of user N is 50.

Keywords: Fast algorithm, queuing model.

1 Introduction

The BMAP (Batch Markovian Arrival Process) has been studied by D.M. Lucantoni[1]. Results for queues with SM (Semi-Markov) services have been obtained by M.F. Neuts[2]. Vacation queues are surveyed by both[3]. A more general queue BMAP/SM/1 which combined these three aspects is studied in [4]. Discrete queue is explored in [5]. The basis of foregoing analyses is that there exist an embedded Markov chain at specific observing instants. Here we give a new fast algorithm for the solution of above general queues which possess such an embedded Markov chain.

The general model of state dependent arrivals, and service times is useful for the performance analysis of computer communication networks. The intent of such networks is to handle a large population of users and a wide range of traffic types which may be modeled as sources driven by underlying Markov chains. The flow and congestion control algorithms that have been proposed may also be modeled by means of service policies which are also governed by underlying Markov chains[9].

A major consequence of the new results is that it allows us to deal with any kind of queue which has an embedded Markov chain by means of the fast algorithm ANK which saves up to 30% CPU time (when the dimension is 50) over the MNK method.

1.1 M/G/1/∞ Type Queuing Model

The Batch Markovian Arrival Process (BMAP) is a generalized Poisson process which allows the arrival intensity to be governed by an underlying Markov chain. BMAP is characterized by a two dimensional Markov process $\{N(t), J(t)\}$, where $N(t)$ is the total number of arrivals in time $(0,t]$, and $J(t)$ is the phase of the process at time t. The transition probability matrix $P(n,t)$ is well structured[1].

The Semi Markov (SM) service time is constructed as follows. Assume that the service times of successive customers form a Semi Markov process with a finite number, m, of states. Theses states are called the types of the service time. The transition probability matrix $H(t)$ of the successive service times is an $m \times m$ matrix of probability mass function on $[0,\infty)$. By assumption its row sums

$$\tilde{H}_i(t) = \sum_{j=1}^{m} \tilde{H}_{ij}(t) \tag{1}$$

are a non-degenerate, proper probability distribution of finite mean d_i, $1 \le i \le m$. The matrix $H = \tilde{H}(\infty)$ is an irreducible stochastic matrix which governs the underlying Markov chain for service time, and the service time distribution for the i-th type happens to be given by $\tilde{H}_i(t)$.

Assume that the service times and the arrival processes are mutually independent. If we consider the BMAP/SM/1 queue immediately after the successive service completions and form a four dimensional sequence $\{I_n, J_n, K_n, X_n\}$ where I_n denotes the number of customers in system immediately after the n-th departure, J_n is the phase of arrival process at that instance, K_n is the type of the n-th service time, and X_n the time between the n-th and the $(n+1)$-th departures. We obtain the Semi-Markov process on the state space $\{(i,j,k) : i \ge 0, 1 \le j \le l, 1 \le k \le m\}$ with transition probability matrix $\tilde{P}(x)$,

$$\tilde{P}(x) = \begin{bmatrix} \tilde{B}_0(x) & \tilde{B}_1(x) & \tilde{B}_2(x) & \cdots & \cdots \\ \tilde{A}_0(x) & \tilde{A}_1(x) & \tilde{A}_2(x) & \cdots & \cdots \\ 0 & \tilde{A}_0(x) & \tilde{A}_1(x) & \cdots & \cdots \\ 0 & 0 & \tilde{A}_0(x) & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix}, \qquad x \ge 0, \tag{2}$$

where the block matrices are given by,

$$\tilde{A}_n(x) = \int_0^x P(n,t) \otimes d\tilde{H}(t) \tag{3}$$

$$\tilde{B}_n(x) = \sum_{j=0}^{n} \tilde{F}_{j+1}(x) \Diamond \tilde{A}_{n-j}(x) \tag{4}$$

where $\otimes$ is a Kronecker product defined by $P \otimes H = \{p_{ij}H\}$; $\Diamond$ is matrix convolution. The meaning of these terms are as follows:

$[\tilde{A}_n(x)]_{ij} = P$ {given a departure at time 0, which left at least one customer in the system and the background process is in state i, the next departure occurs no later than time x with the background process in state j, and during the service there were n arrivals}. The background process is constructed by ordering the arrival phase and service type lexicographically.

$[\tilde{B}_n(x)]_{ij} = P$ {given a departure at time 0, which left the system empty and the background process is in state i, the next departure occurs no later than time x with the background process in state j, and leaving n customers in the system}. Actually, there are $(n+1)$ customers having arrived, the first customer who initialized the busy period will depart the system, and leaving n customers there.

$[\tilde{F}_n(x)]_{ij} = P$ {given that an idle period begins at time 0, with the background process in state i, the end of the idle period occurs no later than time x with the background process in state j, and during the idle period there were n arrivals}.

The stationary probability transition matrix of the embedded Markov chain is

$$P = \tilde{P}(\infty) = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots \\ 0 & 0 & 0 & A_0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix} \tag{5}$$

Denote the invariant vector of P as $x = (x_0, x_1, \ldots)$, where x_i are $l \cdot m$ vectors, here l is the number of phases of the arrival process, m the number of type of service process, and thus $\hat{s} = l \cdot m$ is the number of states of the background process. We have

$$xP = x \tag{6}$$

that is

$$x_i = x_0 B_i + \sum_{v=1}^{i+1} x_v A_{i+1-v}, \qquad \text{for } i \geq 0. \tag{7}$$

Taking the z-transform of (7), we get

$$X(z)[zI - A(z)] = x_0[zB(z) - A(z)] \tag{8}$$

Now, we are to find x_0, since the Markov process given by $\tilde{P}(x)$ is spatially homogeneous, we can define the transition probability when queue decreases by one as G. The transition probability matrix of first passage satisfies

$$G = \sum_{v=0}^{\infty} A_v G^v \tag{9}$$

Similarly, for the first passage time of empty system, we define the transition probability of first passage for empty system as K, then

$$K = \sum_{v=0}^{\infty} B_v G^v \tag{10}$$

For a general queue, the probability of system being empty is

$$x_0 = \frac{k}{kk^*} \qquad \cdot \text{with } kK = k, ke = 1 \tag{11}$$

and vector $k^* = K'(1)e$ is the first order derivative (the row-sum means) of

$$K(z) = \sum_{v=0}^{\infty} B_v G^v(z) \tag{12}$$

(see [4] for detail), where e is an identity column vector. Above analysis applies to the discrete case as well. Since for discrete case the embedded point is fixed at the beginning of each time slot, instead of at the departure instant as for continuous case, we have

$$B_n = A_n \tag{13}$$

which leads to simpler calculations.

1.2 General Algorithm for Analyzing the Queue with Embedded Markov Chain

The following algorithm provides a general procedure for solving a general M/G/1 type queue that has an embedded Markov chain.

Step 1. Computation of matrices A_n and B_n. For a general service process, A_n need to be numerically integrated. The sequence $\{A_n\}$ is computed for a suitably truncated index N_ϵ. The simplest criterion for the truncation can be

$$N_\epsilon = \max_n \left\{ e^T \left(e - \sum_{v=0}^{n} A_v e \right) < \epsilon \right\} \tag{14}$$

where tolerance ϵ is a small positive number.

Step 2. Computation of π, β and ρ,

$$A = \sum_{j=0}^{\infty} A_j \tag{15}$$

$$\pi A = A, \qquad \pi e = 1 \tag{16}$$

$$\beta = \sum_{v=1}^{M} v A_v e \tag{17}$$

For a stable system,

$$\rho = \pi\beta < 1 \tag{18}$$

Step 3. Computation of matrix G, solving the non-linear matrix equation (this is the most time consuming part)

$$G = \sum_{n=0}^{\infty} A_n G^n \tag{19}$$

Step 4. Computation of g and $\overline{\mu}$,

$$gG = g, \qquad ge = 1 \tag{20}$$

thus g is solved. $\overline{\mu}$ is then given by

$$\overline{\mu} = (I - G + eg)[I - A + (e - \beta)g]^{-1} e \tag{21}$$

Step 5. Computation of K, k, k^* and x_0,

$$K = \sum_{v=0}^{\infty} B_v G^v \tag{22}$$

$$kK = k, \qquad ke = 1 \tag{23}$$

$$k^* = e + \left(\sum_{v=1}^{\infty} vB_v G^{v-1}\right)\overline{\mu} \tag{24}$$

$$x_0 = \frac{k}{kk^*} \tag{25}$$

Step 6. Computation of x_i, we have (see [6] for details),

$$x_i = \left(x_0\overline{B}_i + \sum_{j=1}^{i-1} x_j\overline{A}_{i+1-j}\right)(I - \overline{A}_i)^{-1} \tag{26}$$

where

$$\overline{B}_i = \sum_{v=i}^{\infty} B_v G^{v-i}, \qquad \overline{A}_i = \sum_{v=i}^{\infty} A_v G^{v-i} \tag{27}$$

Through out the steps, the Power iteration method is suggested in the evaluation of all invariant vectors, because the dominant eigenvalue of probability transition matrix of stable queuing system is one, and thus the premise of using Power iteration method is satisfied.

2 An Efficient Method for Solving the Nonlinear Matrix Equation $G = \sum_{n=1}^{\infty} A_n G^n$

2.1 Some Previous Methods

The most time consuming step in the algorithm is Step 3. The existing methods are the following.

(a) Successive Substitutes (SS): The method uses iteration directly

$$G_0 = 0, \qquad G_{k+1} = \sum_{n=0}^{\infty} A_n G_k^n, \qquad k \geq 0 \tag{28}$$

This method is simple, but it converges extremely slowly when ρ is close to one.

(b) Modified Successive Substitutes (MSS): Reforming (28), we have

$$G_0 = 0, \qquad G_{k+1} = (I - A_1)^{-1}\left(A_0 + \sum_{n=2}^{\infty} A_n G_k^n\right), \qquad k \geq 0 \tag{29}$$

This method is faster than SS method, since the righthand side of the equation consists of no linear term, but only square and higher power terms.

(c) Newton-Kantorovich method (NK): Recall that for scalar case, Newton method has square rate of convergence, since it makes use of the derivative; and so should be its matrix version method.

$$G_0 = 0, \qquad G_{k+1} = G_k - [F^{(1)}(G_k)]^{-1}F(G_k), \qquad k \geq 0 \tag{30}$$

where

$$F(x) = X - \sum_{n=0}^{\infty} A_n X^n \tag{31}$$

and $F^{(1)}(x)$ is the first order derivative. Unfortunately, it is quite difficult to find the inverse of the matrix version derivative for $F(x)$, this is the key to the success of a method.

(d) Modified Newton-Kantorovich scheme (MNK): To avoid the complexity of calculating the inverse of derivative of $F(x)$, the following approximation is used[10]

$$G_{k+1} = G_k - (F(G_k) + A_1 Z_k + A_2(Z_k G_k + G_k Z_k)) \tag{32}$$

$$G_0 = 0, \qquad Z_k = (I - A_1)^{-1} F(G_k) \tag{33}$$

This method is good, but it uses the derivative of first iteration for all consequent iterations, though it saves the time of calculation of the inverse, it will fail to trace the change of derivative for the heavy traffic and the large dimension cases. The improvement of MNK is possible as the following section shows.

2.2 The New Adaptive Newton-Kantorovich (ANK) Method

The basic idea of Adaptive Newton-Kantorovich method (ANK) is that we make use of the property that the derivative $F^{(1)}(x)$ can be written as

$$F^{(1)}(x_k) = I - J_k \tag{34}$$

for the k-th iteration, where the maximum eigenvalue of J_k is less than one. And thus we can approximate

$$(F^{(1)}(x_k))^{-1} = (I - J_k)^{-1} \tag{35}$$

by

$$B_{k,N} = I + J_k + J_k^2 + \cdots + J_k^N \tag{36}$$

and we dynamically increase the index of N and update the value of J_k when the number of iteration k goes up, in order that we do not waste time on the calculation of $\{J_k^1, J_k^2, \ldots, J_k^N\}$ when k is small; and also do not fail to trace the change of derivative, so that we need fewer iterations. In summary, we try to use more accurate derivatives when we are closer to the fixed point X^0 of the non-linear equation, while the iterations are being carried on.

a) To present the method, we need the concept of Gateaux derivative.

Definition 1. *A mapping $F : D \subset R^n \to R^n$ is Gateaux- (or G-) differentiable at an interior point x of D if there exists a linear operator $A \in L(R^n \to R^n)$ such that , for any $\check{h} \in R^n$,*

$$\lim_{i \to 0} \left(\frac{1}{t}\right) \parallel F(x + t\check{h}) - F(x) - At\check{h} \parallel = 0 \tag{37}$$

Where t is a real number, and $t\check{h} \in R^n$, $\parallel\ \parallel$ is the symbol of the norm.

Note that by the Norm Equivalence Theorem, the limit in (37) is independent of the particular norm on R^n; thus, we can choose the simplest norm, e.g. l_∞ norm which is defined as $\parallel x \parallel_\infty = \max \mid x_i \mid$, $1 \le i \le n$, for saving calculation time of program. As in the one-dimensional case, there is at most one linear operator A, since $\check{h}$ was arbitrary.

With each $\hat{s} \times \hat{s}$ matrix X we may associate the vector $vec\ (X)$ which is the $\hat{s}^2$-vector formed by stacking the successive columns of X one underneath the other. Noting that the equation $AXB = C$, where all matrices are of order $\hat{s} \times \hat{s}$, can be written as $(B^T \otimes$

$A)(vec\ (X)) = vec\ (C)$, where B^T is the transpose of B. Thus we may equivalently consider the mapping Σ:

$$X \to \sum_{n=0}^{\infty} A_n X^n \tag{38}$$

as the nonlinear map on the normed space (R^{s^2}, l_∞) given by $\tilde{\Sigma}$:

$$vec\ (X) \to \sum_{n=0}^{\infty} (A_n^T \otimes I)(vec\ (X^n)) \tag{39}$$

From Definition 1, the Gateaux derivative J of Σ is easily seen to be $\Sigma^{(1)}$:

$$G \to \sum_{v=1}^{\infty} \left(\sum_{j=0}^{v-1} X^j G X^{v-1-j} \right) A_v \tag{40}$$

which is equivalent to map $vec\ (G) \to \tilde{J}(vec\ (G))$, where

$$\tilde{J} = \sum_{v=1}^{\infty} \sum_{j=0}^{v-1} (A_v^T (X^T)^{v-1-j} \otimes X^j) \tag{41}$$

Recall the Newton method, we have

$$vec(X_{k+1}) = vec(X_k) - (I - \tilde{J}_k)^{-1} vec(F(X_k)) \tag{42}$$

b) To develop the algorithm, we need the following Neumann lemma.

Lemma 1. *Let operator $B \in L(R^n)$ and assume that the spectral radius of linear operator B is less than one, then $(I - B)^{-1}$ exists, and*

$$(I - B)^{-1} = \lim_{N \to \infty} \sum_{i=0}^{N} B_i \tag{43}$$

Proof. Since the spectral radius of operator B is less than one, $(I - B)$ has no zero eigenvalues and hence is invertible. Note that the identity

$$(I - B)(I + B + B^2 + \cdots + B^N) = I - B^{N+1} \tag{44}$$

yields

$$(I + B + B^2 + \cdots + B^N) = (I - B)^{-1} - B^{N+1}(I - B)^{-1} \tag{45}$$

we see (43) hold.

If we approximate $(I - B)^{-1}$ with $B_N = (I + B + B^2 + \cdots + B^N)$, then the error is $B^{N+1}(I - B)^{-1}$. Obviously the greater the N is, the smaller the error would be. In reference [2], it is shown that the spectral radius of J is less than one as long as the load intensity of queue $\rho \neq 1$. We assume that $\rho < 1$, then we can use this approximation, thus (42) becomes

$$vec(X_{k+1}) = vec(X_k) - \tilde{B}_{k,N}(vec(F(X_k))) \tag{46}$$

where

$$\tilde{B}_{k,n} = I + \tilde{J}_k + \tilde{J}_k^2 + \cdots + \tilde{J}_k^N \tag{47}$$

c) To prove the convergence of the iterations, we need the concepts of partial order and order-convex.

Definition 2. *For $x, y \in R^n$, we say that $x \leq y$ under partial ordering, if and only if $x_i \leq y_i$, where $i = 1, \ldots, n$. For matrices, we say $X \leq Y$, if and only if $vec(X) \leq vec(Y)$.*

It is easily seen that $B_{k,N}$ is the sub-inverse of $F^{(1)}(x_k)$, since

$$F^{(1)}(x_k)B_{k,N} \leq I, \qquad B_{k,N}F^{(1)}(x_k) \leq I \tag{48}$$

Definition 3. *A mapping $F : D \subset R^n \to R^n$ is order-convex on a convex subset $D_0 \subset D$ if*

$$F(\lambda x + (1-\lambda)y) \leq \lambda F(x) + (1-\lambda)F(y) \tag{49}$$

whenever $x, y \in D_0$ are comparable and $\lambda \in (0, 1)$.

It is not difficult to verify that the matrix version algebraic polynomial $F(x)$ given by (31) is of order-convex. And also for such a function, we have

$$F(y) - F(x) \geq F^{(1)}(x)(y - x), \qquad \text{for } x, y \subset D_0 \tag{50}$$

Now we are able to prove the convergence theorem for the new ANK algorithm.

Theorem 1. *Consider the iteration $X_0 = 0$,*

$$X_{k+1} = X_k - B_{k,N}F(X_k) \tag{51}$$

then for all k, $N \geq 0$

$$0 \leq X_k \leq X_{k+1} \leq X^0, \qquad F(X_k) \leq 0 \tag{52}$$

further, if there exists a non-singular $B \geq 0$, such that

$$\lim_{k\to\infty, N\to\infty} B_{k,N} \leq B \tag{53}$$

then

$$\lim_{k\to\infty} X_k = X^0 \tag{54}$$

Proof. Firstly assume that (52) holds for $k (k \geq 0)$, then since $F(X_k) \leq 0$, $X_{k+1} = X_k - B_{k,N}F(X_k) \geq X_k$, and from the convexity of F, We have

$$F(X_{k+1}) \leq F(X_k) + F^{(1)}(X_k)(X_{k+1} - X_k) \tag{55}$$

that is

$$F(X_{k+1}) \leq (I - F^{(1)}(X_k)B_{k,N})(F(X_k)) \tag{56}$$

since $F^{(1)}(x_k)B_{k,N} \leq I$, we get $F(X_{k+1}) \leq 0$, $X_{k+2} \geq X_{k+1}$. Secondly,

$$\begin{aligned} X^0 &= X^0 - B_{k,N}F(X^0) \\ &= X_{k+1} + (X^0 - X_K) + B_{k,N}(F(X_k) - F(X^0)) \end{aligned} \tag{57}$$

Again using the property of convexity,

$$X^0 \geq X_{k+1} + (X^0 - X_k) + B_{k,N}F^{(1)}(X_k)(X_k - X^0) \tag{58}$$

that is

$$X^0 \geq X_{k+1} + (I - B_{k,N}F^{(1)}(X_k))(X_k - X^0) \tag{59}$$

Since $B_{k,N}F^{(1)}(x_k) \leq I$, thus $X^0 \geq X_{k+1}$.

Then, by induction, (52) holds for all k, and we have $X_k \to \check{X}$ as $k, N \to \infty$. Now

$$0 = \lim_{k,N\to\infty}(X_{k+1} - X_k + B_{k,N}F(X_k)) = BF(\check{X}) \leq 0 \tag{60}$$

thus $\check{X} = X^0$.

d) Finally we give the detailed steps of the algorithm, and the explanation of why it is faster than the others.

Algorithm of ANK: /* This algorithm is for solving matrix version algebraic equation */

Step 1: Input A_n,
N, /* $n \leq N$, the maximum number of Coefficient Matrices A_n */
d, /* $d \in (0, 1)$, the factor that stops the truncation */
c_1, c_2, /* the factor that decides the maximum number of no-updating allowed */
I, /* the maximum number of iterations allowed */
ϵ, /* the tolerance that stop the iterations */
Step 2: Define $F(X) = X - \sum_{n=0}^{N} A_n X^n$; $X = 0$; $Z = (I - A_1)^{-1}$;
Step 3: For $(i = 1; i \leq I; i++)$ do Step 4–16; /* iteration */
{Step 4: $B = -F(X)$;
Step 5: $\mu_1 = \| B \|$; $J = 0$, $F = ZB$; $K = c_1 + c_2\rho$;
Step 6: For $(k = 1; k \leq K; k--)$ goto Step 14; /* no update */
{Step 7: $E_v = 0, E = 0$;
Step 8: For $(v = 1; v \leq N; v++)$ do Steps 9–11;
{Step 9: $E_v = \sum_{j=0}^{v-1} X^j F X^{v-1-j} A_v$;
Step 10: $E = E + E_v$, $\eta_v = \| E_v \|$;
Step 11: If $\eta_v \leq d\eta_1$ break;}
Step 12: $J = EF^{-1}$;
Step 13: $\mu_2 = \| F \|$;
Step 14: $Z = (I - J)^{-1}$; If $\mu_2 \leq d\mu_1$ break;}
Step 15: $X = X + ZB + E$
Step 16: If $\mu_1 \leq \epsilon$ break;}
Step 17: Output X; /* final solution X^0 */
Step 18: End;

There are three loops in the algorithm, the first loop is for iteration of X; the tolerance ϵ will stop the iteration. The second loop decides how many times not to update. The innermost loop is for truncating (41) up to proper terms, here the factor d decides the relative error of truncation. From (41) we see that the derivative J is an increasing function of iterate X, thus the number of terms in Step 9 will increase with the number of iterations, and thus increasing the accuracy of the derivative.

3 Application to the N-MMBP/D/1/∞ Model

3.1 The Model

In the telecommunication area, there is an increasing interest in the discrete time model as all information is packetized into ATM cell (a packet of 424 bits) and then to be transported through networks. The arrival ATM cells consist of all kinds of traffic such as voice, data and

video which could be coming from multimedia computer users. To analyze the performance of such networks, we use an N-MMBP/D/1/∞ queuing system, which means there are N independent traffic sources modeled as MMBP arriving to the system as shown in Fig.1.

The state of the system is the number of cells in the queue and the phase of the arrival of process is the number of active sources. In the case of each source being Poisson or Bernoulli arrival, the phase does not matter, since the superposition of arrivals is still Poisson or Bernoulli arrival. The simplest increase in complexity is that each source is of two states. We may take them as on and off states, as shown in Fig.2. In the on state, a source transmits a cell with probability p in each data slot. The transition probabilities are q_0 and q_1, respectively. If $p = 1$, we have a constant arrival process. If $q_0 = 0$, we have a Bernoulli arrival process. We can also make N, the number of binary sources, as large as we wish.

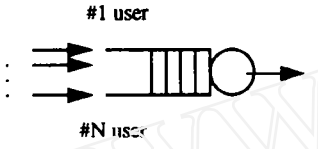

Fig.1. Queuing model of N-MMBP/D/1.

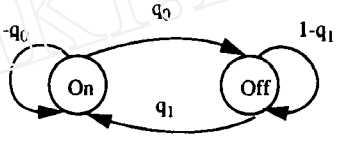

Fig.2. MMBP source model.

We define a state vector, (Q_k, R_k). Q_k is the number of cells in the queue at time k, and R_k is the number of binary sources in the active state at time k. It is not difficult to write the state transition matrix T. By $S_k = (Q_k, R_k)$, we assume lexicographic order. Notice that:

$$Q_k = \max(0, Q_{k-1} - 1) + \tilde{A}_k \tag{61}$$

where $\tilde{A}_k$ is the arrival process. We assume a finite queue with maximum buffer size M, so that $\tilde{A}_k = 0$ if $Q_k = M$. The probability that there are m cells coming in a time slot conditioned on there are n active sources is:

$$P(\tilde{A}_k = m \mid R_k = n) = \binom{n}{m} p^m (1-p)^{n-m} \tag{62}$$

And the transition probability for the number of active sources changes according to the rule:

$$P(R_k = i \mid R_{k-1} = j) = \sum_{l=\max(0,i-j)}^{\min(i,N-j)} \binom{N-j}{l} q_1^l (1-q_1)^{N-j-l} \binom{j}{i-l} q_0^{i-l} (1-q_0)^{j-i+l} \tag{63}$$

Suppose

$$(S_k = s) \Leftrightarrow (Q_k = m_s, R_k = n_s) \text{ or } s = m_s \times N + n_s \tag{64}$$

The transition matrix $T = \{t_{ij}\}$ is the random driving component of the whole system. We can write down the element of the matrix t_{ij} as:

$$\begin{aligned} t_{ij} &= P(S_k = j \mid S_{k-1} = i) \\ &P((Q_k = m_j, R_k = n_j) \mid Q_{k-1} = m_i, R_{k-1} = n_i)) \end{aligned} \tag{65}$$

From above equations (61)–(63), we can get:

$$\begin{aligned} t_{ij} &= P(R_k = n_j \mid R_{k-1} = n_i) \times P(A_k = m \mid R_k = n_j) \\ &= \binom{n_j}{m} p^m (1-p)^{n_j - m} \times \sum_{l=\max(0, n_i - n_j)}^{\min(n_j, N-n_i)} \binom{N-n_i}{l} q_1^l (1-q_1)^{N-l-n_i} \\ &\quad \times \binom{n_i}{n_j - l} q_0^{n_j - l} \times (1-q_0)^{n_i - n_j + l} \end{aligned} \tag{66}$$

Here

$$m = \begin{cases} Q_k - Q_{k-1} + 1 & (Q_{k-1} > 1, Q_k > Q_{k-1}) \\ Q_k & (Q_{k-1} \leq 1) \\ 0 & (Q_k < Q_{k-1}) \end{cases} \tag{67}$$

We can obtain the steady state probability $\bar{P} = (p_0, p_1, \ldots, p_M)$ by solving the equation:

$$\bar{P}T = \bar{P}, \qquad \bar{P}e = 1 \tag{68}$$

However when N is large, such as 50, the size of transition matrix T goes so large that it is difficult to solve the equation using ordinary method. ANK method is a good choice under this situation. And it is particularly useful when the buffer size M is also very large, usually $M \geq N$.

3.2 Numerical Results

Now, we apply matrix method to the problem. Rewrite the transition matrix T as:

$$T = \begin{bmatrix} A_0 & A_1 & A_2 & A_3 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots \\ 0 & 0 & 0 & A_0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix} \tag{69}$$

where A_m is an $N \times N$ matrix, $m = 0, 1, \ldots, N$. And for $a_{ij}^{(m)}$, the ij-th element of A_m, we have:

$$a_{ij}^{(m)} = t_{m \times N + i, m \times N + j} \tag{70}$$

Following the general algorithm mentioned in Subsection 1.2, we can obtain the numerical results about the performance of the system, such as the traffic load and the steady probability of queue length. The key step is to solve the nonlinear matrix equation. It is time consuming. We use the Adaptive Newton-Kantorovich (ANK) algorithm to solve the matrix equation.

Fig.3 shows some results about the distribution of steady state probability of queue length, under the assumption that: the dimension of the system N is 30, the traffic density of each source p is 0.05, the transition probability are $q_0 = 0.4$, $q_1 = 0.8$, respectively. The traffic load $\rho = 0.8286$ which is calculated from Eq.(18). Fig.4 shows the steady state probability distribution with both the queue length and the phase information.

In the next section, we compare ANK and MNK algorithm when they are used to solve the equation for different cases of this N-MMBP/D/1/∞ queue model.

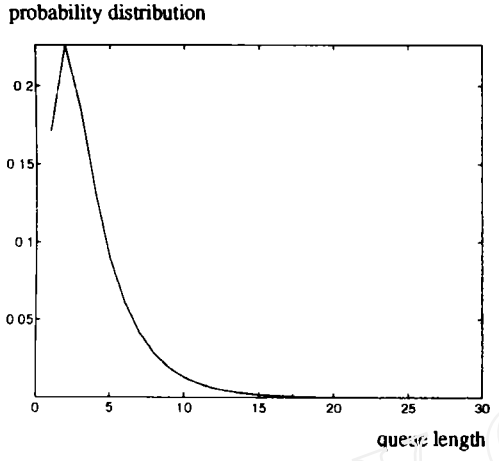

Fig.3. Probability distribution of queue length.

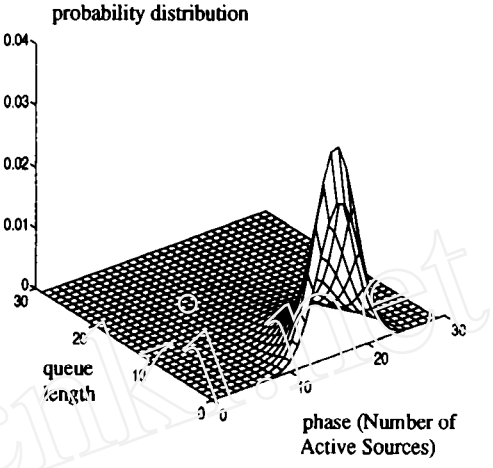

Fig.4. Joint probability distribution of queue length and phase.

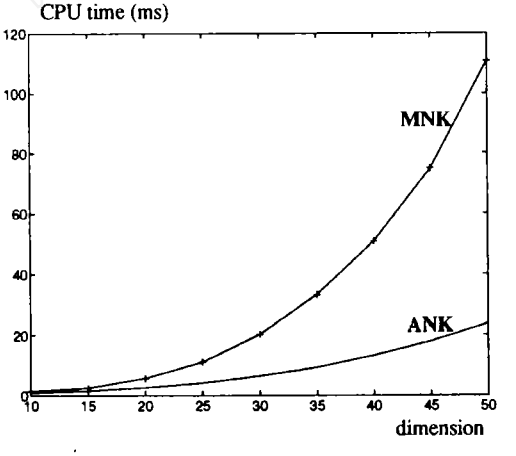

Fig.5. Comparison of CPU time versus dimension between ANK and MNK.

3.3 Comparisons of ANK with MNK

The major limitation in the applicability of the matrix analytic techniques is the "curse of dimensionality"[8], which may incur intolerably long CPU times for computation. In Fig.5, the CPU time required to solve the N-MMBP/D/1 queue is shown as a function of dimensionality for both MNK and ANK, these curves show that the larger the dimension is, the better ANK is over MNK. This can be explained by the fact that the larger the dimension, the relatively smaller the "overhead" of the program for producing adaptability of ANK. If we plot Fig.5 in logarithmically scale, we can find that the CPU time increases almost cubically for MNK, while the CPU time increases almost quadratically for ANK. The parameter for the case is $q_0 = 0.4$, $q_1 = 0.8$, $\rho = 0.85$.

Another shortcoming of previous methods is when the traffic intensity goes up, the number of iteration increase rapidly. The results in Fig.6 and Fig.7 show that ANK is not as good as MNK when traffic intensity is low, but it becomes better than MNK when $\rho > 0.45$ for CPU time, and $\rho > 0.7$ for number of iterations. This can be explained by the same reason of "overhead" as previous. The parameter for the case is $q_0 = 0.4$, $q_1 = 0.8$,

$N = 30$.

In the process of analyzing the performance of the ATM network, we often need to calculate quantities up to 10^{-14} accuracy, so we compare the ANK with MNK by changing the error tolerance ϵ in Fig.8 for CPU time and Fig.9 for iterations, from which we see that ANK still has an advantage over MNK. The parameter for this case is $q_0 = 0.4$, $q_1 = 0.8$, $N = 20$, $\rho = 0.85$.

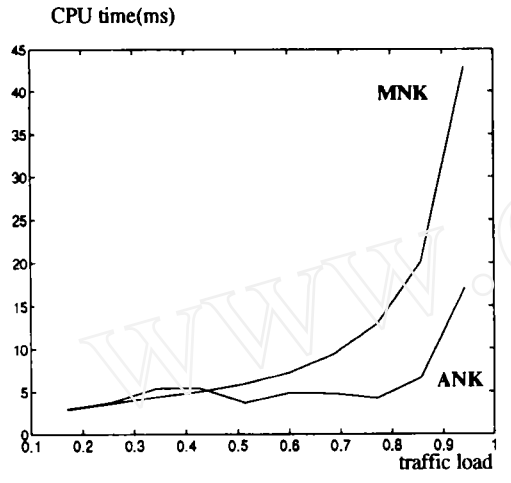

Fig.6. Comparison of CPU time versus traffic load between ANK and MNK.

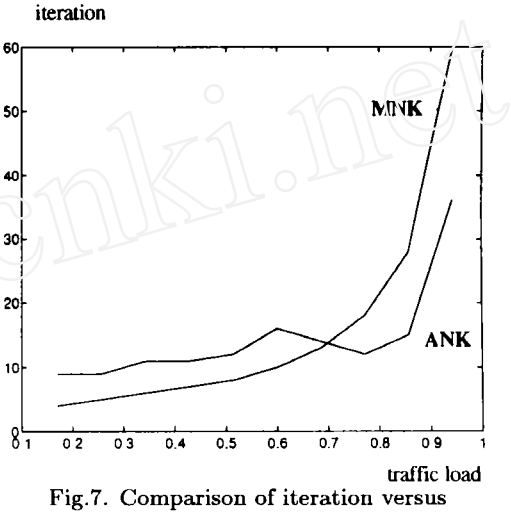

Fig.7. Comparison of iteration versus traffic load between ANK and MNK.

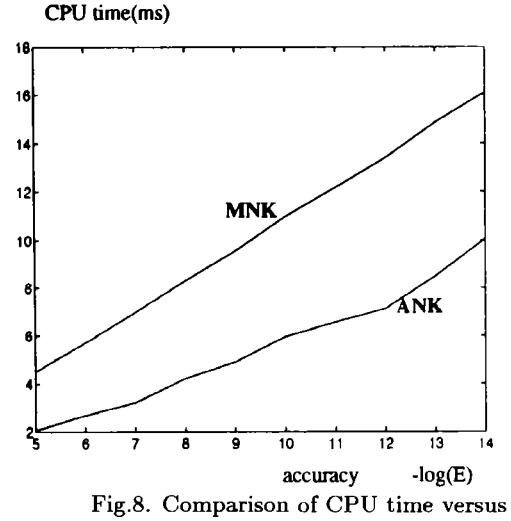

Fig.8. Comparison of CPU time versus accuracy between ANK and MNK.

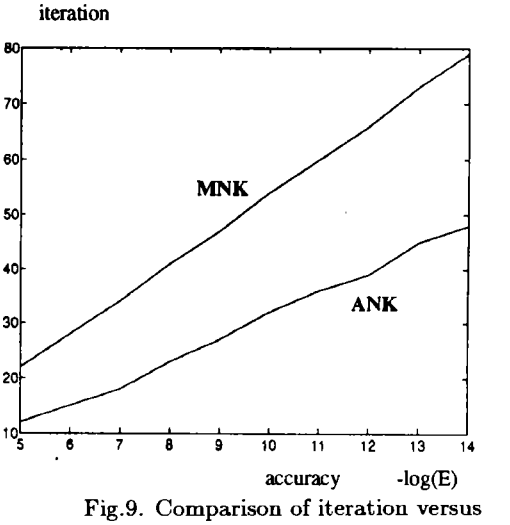

Fig.9. Comparison of iteration versus accuracy between ANK and MNK.

4 Conclusion

We have made use of the adaptive technique improved the traditional second-order matrix-version Newton iteration method. The ANK method achieved quadratically convergence rate regarding the dimension of the system. The similar results for finite queues can also be established. The extension to the level-dependent[7] case, where each row of transition probability matrix of embedded Markov chain is no longer the same, is possible. Especially for M/M/1 type, it is quite ready, because the matrix is of "diagonal strip", not

of triangular. The general algorithm has been developed into a systematic software package.

References

[1] Lucantoni D M. New results on the single server queue with a batch Markovian arrival process. *Stochastic Models.*, 1991, 7(1): 1-46.

[2] Neuts M F. Structured Stochastic Matrices of M/G/1 Type and Their Applications. New York & Basel: Marcel Dekker Inc., 1989.

[3] Lucantoni D M, Meier-Hellstern K S, Neuts M F. A single server queue with server vacations and a class of non-renewal arrival processes. *Advanced Applied Probability*, 1990, 22: 676-705.

[4] Huang J, Hayes J F. A study of the matrix analytic method and its application in performance evaluation of broadband and related system. In *Int'l Symp. on Operations Research with Applications in Engineering, Technology and Management (ISORA)*, Beijing, China, August 19-22, 1995.

[5] Bruneel H, Kim B G. Discrete Time Models for Communication Systems Including ATM. London: Kluwer Academic Publisher, 1993.

[6] Ramaswami V. A stable recursion for the steady state vector in Markov chains of M/G/1 type. *Stochastic Models*, 1988, 4(1): 183-188.

[7] Beuerman S L, Coyle E J. State space expansions and the limiting behavior of quasi-birth-and-dearth processes. *Advanced Applied Probability*, 1989, 21: 284-314.

[8] Gun L. Experimental results on matrix-analytical solution techniques-extensions and comparisons. *Stochastic Models*, 1989, 5(4): 669-682.

[9] Huang J, Chen Y, Hayes J F, Ali M M. Performance Analaysis of Tunable Leaky Bucket in ATM Networks. ITC International Teletraffic Seminar, Russia, 1995.

[10] Ramaswami V. Nonlinear matrix equations in applied probability-solution techniques and open problems. *SIAM Review*, 1988, 30(2): 256-263.

HUANG Jun obtained his Ph.D. degree in electrical engineering from Southeast University in 1992. He is currently a Research Associate in the Department of Electrical Engineering at Concordia University. His research interests are in the area of queuing theory and performance evaluation of telecommunication networks.

ZHU Tao received her Ph.D. degree in electrical engineering from Southeast University in 1993. She is now working for BNR/NT. Her research interests include performance analysis and simulation study with AI technique, traffic modelling and congestion control in BISDN.

J.F. HAYES is a tenure Professor of Department of Electrical Engineering at Concordia University in Montreal. He received his Ph.D. from Berkeley University. He is the senior editor of IEEE Journal of Selected Areas in Communication and the Fellow of IEEE. He published two books and a number of papers in digital and computer communications.