ON THE PROGRAM OF THE SPECTRAL METHOD FOR COMPUTING THE STATIONARY PROBABILITY VECTOR FOR A BMAP/G/1 QUEUE

SHOICHI NISHIMURA Tokyo University of Science
NAOHIKO YATOMI Tokyo University of Science

Abstract—Batch Markovian arrival processes (BMAPs) are useful in modeling IP traffic. In this paper, we release the program of the spectral method for computing the stationary probability vector for a BMAP/G/1 queue and explain how to use it. Using this program, we can efficiently calculate statistical values of a BMAP/G/1 model whose underlying process is estimated from actual IP traffic. And in future, the computational method may be utilized for simulating and optimizing various IP traffic.

I. INTRODUCTION

IP traffic data are transmitted through a queue when they pass through a server or a router. Some queueing models are offered in order to express actual IP traffic. It is impossible to describe IP traffic using a simple Poisson process because it has special properties such as burst and self-similarity. In this paper, we use a batch Markovian arrival process (BMAP) introduced by Neuts [3] to describe it. A BMAP is great flexible and its queueing models are analytically tractable. Using a BMAP, we can express lots of stochastic processes which have burst and self-similarity. Actually, in [4], some models which express self-similarity of actual IP traffic are introduced. It is shown that the simulation results (e.g. the stationary probability) of the analytical model is in good agreement with the numerical values of raw traffic.

When we model IP traffic by a BMAP, we observe the arrival epochs first. We can classify the states which have no arrival using the EM algorithm (See [4]). Finally, we apply them to a BMAP.

Some well-known matrix analytical methods for BMAP/G/1 are studied. However, these methods are inadequate for models which have a large size of batch arrivals. The spectral method in [3] is applicable to these models.

In [5], the spectral method for computing the stationary probability vector for a BMAP/G/1 queue is described. The application of the spectral method to a practical case is developed in [4]. And now, we release the program of the spectral method and the algorithm improved to increase generality, stability, preciseness and computational speed. The computational speed is faster than any other available methods. We believe that it must be important to release all these three pillars - the method, application and program.

When we implement the spectral method, graphical observations and easy treatment of complex numbers are especially required. Decimal BASIC [7] has the ability to realize them. Therefore we write our program in Decimal BASIC and open it in [6]. All calculations are performed in double precision in our program. To our regret, however, there exists a defect. If the size of the underlying Markov process is so large, the computation fails because determinants of matrices are calculated many times in the spectral method. To overcome the defect, we are trying to rewrite the program in C++. If we use C++ and some arithmetic libraries for it, we can carry out the computation in arbitrary precision with high performance. Thus we can treat a wider range of large size models.

When we derive the stationary probability vector, we need two computational steps. The first is to calculate the invariant probability vector g of the stochastic matrix G which is the minimal nonnegative solution of a matrix nonlinear equation. Since the precision of the vector g has a great influence on that of the stationary probability vector, efficient algorithms are needed. The second is to calculate the stationary probability vector from the vector g. In this step, we use the inverse Fast Fourier Transform (FFT) for the vector generating function.

II. BMAP/G/1

We consider a BMAP whose state space of the underlying Markov process is \{i : i = 1, \ldots, M\}. Let \( D_k = \begin{pmatrix} d_{0}^{(k)} & d_{1}^{(k)} & \cdots & d_{i}^{(k)} \end{pmatrix} \) \((i, j = 1, \ldots, M)\) be the \( M \times M \) matrix, where \( d_{i,j}^{(k)} \) is the phase transition rate from \( i \) to \( j \) with an arrival of batch size \( k \). Let \( D_0 = \begin{pmatrix} d_{0}^{(0)} & d_{1}^{(0)} & \cdots & d_{i}^{(0)} \end{pmatrix} \) \((i, j = 1, \ldots, M)\) be the \( M \times M \) matrix, where \( d_{i,j}^{(0)} \) is the phase transition rate from \( i \) to \( j \) without arrival and \( d_{i,j}^{(0)} = -\left( \sum_{k=1}^{\infty} d_{i,j}^{(k)} + \sum_{k=1}^{\infty} \frac{d_{i,j}^{(k)}}{k} \right) \). For instance, for an Interrupted Poisson process (IPP), \( D_0 \) and \( D_1 \) are given by

\[
D_0 = \begin{pmatrix} -a-b & b & 0 \\ c & -c & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad D_1 = \begin{pmatrix} a & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.
\]

As usual, assume that for \(|z| \leq 1\), \( D(z) = \sum_{k=0}^{\infty} D_k z^k \) is analytic and that \( D = D(1) = \sum_{k=0}^{\infty} D_k \) is the irreducible infinitesimal generator of the underlying Markov process. And let \( \pi = (\pi_1, \pi_2, \ldots, \pi_M) \) be the invariant probability vector of \( D \) \((\pi D = 0 \text{ and } \pi e = 1)\). The mean arrival rate of customers is given by \( \lambda = \pi D'(1)e = \pi \sum_{k=1}^{\infty} k D_k e \).
Let $H(t)$ be a distribution function of the service time with mean $1/\mu$. Let $A_{kj}$ be the matrix whose $(i,j)$th element is the phase transition probability from $i$ to $j$ with $k$ arrivals during a service time. Then $A(z) = \sum_{k=0}^{\infty} A_{kj}z^k = \int_0^\infty \rho(t) e^{zt} dH(t)$. Let $G$ and $g$ be the phase transition stochastic matrix of the first passage time from the level $k+1$ to $k$ and its invariant probability vector $(gG = g, \; ge = 1)$, respectively. Then $G$ is the minimal nonnegative solution of a matrix nonlinear equation of

$$G = \sum_{k=0}^{\infty} A_{kj}G^k. \tag{1}$$

Assume that the traffic intensity $\rho = \lambda/\mu < 1$.

Let $p_k = (p_{k,0}, \ldots, p_{k,M})$ $(k \geq 0)$ be the $M$-dimensional row vector whose $i$th entry $p_{k,i}$ is the stationary joint probability that the arrival phase is $i$ and the number of customers in the system is $k$ just after service completion epochs. Then $p = (p_0, p_1, \ldots)$ is the stationary probability vector just after service completion epochs.

The vector generating function $p(z)$ of $p_k$ is given by

$$p(z) = \lambda^{-1}(1 - \rho)gD(z)A(z)(zI - A(z))^{-1}. \tag{2}$$

For more mathematical explanations, see Lucantoni [2] and Neuts [3].

### III. Spectral Method for the Vector $g$

In this section, we consider the spectral method based on eigenvalues for calculating the vector $g$.

Let $\alpha_i(z)$ $(i = 1, \ldots, M)$ be the $i$th eigenvalue of $D(z)$. Let $u_i(z)$ and $v_i(z)$ be the corresponding left and right eigenvectors ($u_i(z)D(z) = \alpha_i(z)u_i(z)$ and $D(z)v_i(z) = \alpha_i(z)v_i(z)$) normalized so that $u_i(z)w_i(z) = 1$ $(i = 1, \ldots, M)$. Particularly, $\alpha_1(1) = 0$, $u_1(1) = \pi$ and $v_1(1) = e$. It is proved in [5] that for $|z| < 1$,

$$\alpha_i(z) \in \mathbb{C} = \{ \alpha : |\alpha + \delta| \leq 1 \},$$

where $\delta = \max_i (-d^{(0)}_i)$. Assume that for $|z| = 1$, all eigenvalues $\alpha_i(z)$ are simple.

Now, let $z_i$ and $\eta_i$ $(i = 1, \ldots, M)$ be the eigenvalues and the corresponding right eigenvectors of $G$, respectively $(z_i = 1$, $\eta_i = e)$. Assume that all $z_i$(i = 1, ... , M) are simple. Then the $M \times M$ matrix $Y = (\eta_1, \ldots, \eta_M)$ is nonsingular. Post-multiplying of (1) by $\eta_i$, we derive the following theorem.

**Theorem 1**: ([5]) $z_i$ $(i = 1, \ldots, M)$ are zeros of $\det(zI - A(z))$ on the unit disk. And there are $M$ zeros $\gamma_i$ $(i = 1, \ldots, M)$ of

$$\det(\alpha I - D(h(\alpha))) \tag{3}$$

in $\Omega$, which satisfy $z_i = h(\gamma_i)$ $(i = 1, \ldots, M)$, where $h(\alpha) = \int_0^\infty e^{zt} dH(t)$.

And post-multiplying of $p(z)(zI - A(z)) = \frac{1 - \rho}{\lambda} gD(z_i)A(z_i)$ by $\eta_i$, we have the following theorem.

**Theorem 2**: ([5])

$$g = e_1 Y^{-1},$$

where $e_1 = (1, 0, \ldots, 0)$ is the first unit row vector.

If we get distinct $M$ zeros $\gamma_i$ $(i = 1, \ldots, M)$, the corresponding right eigenvectors $\eta_i$ $(i = 1, \ldots, M)$ are derived. Then the vector $g$ is immediately obtained from Theorem 2. To calculate all the $M$ zeros $\gamma_i$, we propose an efficient method in the next section.

### IV. The Modified Durand-Kerner Method

The Durand-Kerner method (D-K method) is known as an iterative method for calculating all zeros of a polynomial simultaneously ([1]). Using it, we consider to obtain all the $M$ zeros $\gamma_i$ $(i = 1, \ldots, M)$ simultaneously.

It should be noted that for the equation (3), in general, there are infinite unnecessary zeros. To solve this problem, in the same way as [5], we propose a modification of the D-K method given by

$$a_i^{(v+1)} = a_i^{(v)} - \frac{\det(\alpha_i^{(v)} I - D(h(\alpha_i^{(v)})))}{\phi(\gamma_i^{(v)}) \prod_{j=1,j \neq i}^{M} (\alpha_i^{(v)} - \alpha_j^{(v)}),} \tag{4}$$

where $\alpha_i^{(v)}$ $(i = 1, \ldots, m)$ is the $v$th iterative approximation of $\gamma_i$ $(i = 1, \ldots, m)$. $\phi(\gamma_i^{(v)})$ functions as a revision term to keep stability of the iteration (4). We derive $\phi(\gamma_i^{(v)})$ in the next proposition.

**Proposition 1**: ([5]) Suppose that $\gamma_i$ is the $l(i)$th eigenvalue of $D(z_i)$, i.e. $\gamma_i = \alpha_l(i)z_i$. Then we have

$$\phi(\gamma_i) = (1 - a_l(i)z_i)h(\gamma_i) \frac{\prod_{j=1,j \neq i}^{M} (\gamma_i - \alpha_l(j)z_i)}{\prod_{j=1,j \neq i}^{M} (\gamma_i - \gamma_j)}, \tag{5}$$

where $a_l(i)z_i = u_l(i)z_iD(z_i)w_l(i)z_i$.

As mentioned above, there exist unnecessary zeros of (3) out of $\Omega$. For unsuitable initial values, an iterative sequence may converge to such an unnecessary zero. To overcome this difficulty, we propose a double for-loop iteration.

First, suppose that $p_l$ form an increasing sequence of traffic intensity $(\rho_0 = 0 < \rho_1 < \ldots < \rho_K = \rho)$. Then $\mu_0 \equiv \lambda/\rho_0$ forms a decreasing sequence. For each $H_s(t) \equiv H(t, \mu_s)$, let $\gamma_s(i) = (i = 1, \ldots, M)$ be the zeros of $\det(\alpha_i - D(h^{(s)}(\alpha)))$ in $\Omega$, where $h^{(s)}(\alpha) = \int_0^\infty e^{zt} dH_s(t)$. Note that $\gamma_s(0) = \alpha_1(i)$, $h^{(s)}(0) = 1$, $h^{(s)}(0)' = 0$ and $\phi^{(s)}(\gamma_s(i)) = 1$.

Next, we calculate $\gamma_s(i)$ $(i = 1, \ldots, M)$ for $s = 1, \ldots, K$ using the modified D-K method with initial values $\gamma_s^{(s-1)}(i = 1, \ldots, M)$. Then the M zeros $\gamma_s(i)$ gradually get closer to $\gamma_i$ which we need and finally $\gamma_i$ are given by $\gamma_i = \gamma_s^{(K)}(i, \ldots, i)$.

The intermediate zeros $\gamma_s(i)$ $(i = 1, \ldots, K)$ are needed not to be numerically precise values. For $s < K$, we set the number $N_1$ of iteration to a small number, e.g. $N_1 = 20$. However, the zeros $\gamma_s(i)$ $(i = 1, \ldots, M)$ should be numerically precise values. For $s = K$, we set the number $N_2$ of iteration to a large number, e.g. $N_2 = 100$. 
V. CALCULATION OF THE STATIONARY PROBABILITY VECTOR

In this part, we discuss the spectral method for calculating \( p_n \). We use the inverse Fast Fourier Transform (FFT) for the vector generating function \( p(z) \) in (2).

Let \( N \) be a sufficiently large integer such that the tail probability \( \sum_{n=N}^{\infty} p_n \) is negligible. Let \( \omega = e^{2\pi i/N} \) be the \( N \)th root of the unity. From assumption and the diagonal representation of \( D(z)A(z)(I - A(z))^{-1} \) in (2), we have the following proposition.

Proposition 2: \((5)\) \( p_n \) is given by

\[
\frac{1}{N} \sum_{k=0}^{N-1} p(o^k) \omega^{-kn} = \frac{\lambda - 1 - (1 - \rho)g}{N} \tag{6}
\]

\[
\times \sum_{k=0}^{N-1} M_{i,j} \alpha_i(o^k) \nu_i(o^k) \omega^{-kn},
\]

where for \( k = 0 \) and \( i = 1 \), \( \alpha_i(o^k) \nu_i(o^k) \omega^{-kn} = \frac{\lambda - 1 - \rho}{1 - \rho} \).

Note that \( \alpha_i(o^k) \) \((i = 1, \ldots, M)\) are eigenvalues of \( D(o^k) \) and they are derived by the conventional D-K method with initial values \( \alpha_i(o^k) \) \((i = 1, \ldots, M)\).

VI. ALGORITHM

In this section, computational algorithms corresponding to the above theories are described.

There are some functions to realize various purposes of researchers in our program. First, we can define a service time distribution as a constant service or a gamma distribution. There are some functions to realize various purposes of researchers in our program. First, we can define a service time distribution as a constant service or a gamma distribution.

At the beginning of this section, there is a list of variables used in the program. Next, input, output and concrete computation are mentioned. If necessary, they are described with the index number of sub-routines in the program. At the end, some ideas to make the computation faster, more precise and more stable are stated.

A. Variables

The variables used in our program are as follows. The corresponding symbols in this paper and explanations are described with them. Now, \( i = 1, \ldots, m; j = 1, \ldots, m; l = 0, \ldots, b \text{ num}; n = 0, \ldots, N - 1. \)

Modeling parameters

- \( m \): the size of the underlying Markov process
- \( b \text{ num} \): the number of types of batch sizes
- \( \rho \): the traffic intensity \( \rho \)
- \( D(x, j, l) \), \( batch(l) \): \( d_{i,j}^{(k)} = D(x, j, l) \) if \( k = batch(l) \) (See Idea 1.)
- \( sp \): \( sp = 1 \) (constant service, \( h(\alpha) = \exp(\alpha) \)); \( sp > 0 \) (gamma distribution with parameter \( sp, h(\alpha) = (\frac{\alpha}{\alpha - \sigma})^{sp} \))

Computational parameters

- \( NN \): the parameter in FFT. \( N = 2^{NN} \) in (6), which is the number of partition in FFT.
- \( \epsilon \): a fixed sufficiently small number \( \epsilon > 0 \) (See Ideas 2-3.)
- \( slice \): the number of partition in the modified D-K method (See \( K \) in the last paragraph of Section IV.)
- \( it(1) \): the number of iteration in the modified D-K method (when \( s < slice \))
- \( it(2) \): the number of iteration in the modified D-K method (when \( s = slice \))
- \( it(3) \): the number of iteration in the D-K method
- \( x \text{ width} \): the width of the graph for the stationary probability

Internal parameters

- \( \text{distance} \): the distance in which we rotate the eigenvalues (See Idea 2.);
- \( \text{distance} = \max (\max(\alpha_j(1) - \alpha_j(1)) \epsilon, \delta, 50) \)

B. Input

We input these parameters in the beginning and Sub-routine[1] of our program.

\( m, b \text{ num}, \rho, D(x, j, l), batch(l), NN, \epsilon, slice, sp, x \text{ width}, \text{distance} \) \((i = 1, \ldots, m; j = 1, \ldots, m; l = 0, \ldots, b \text{ num}; t = 1, \ldots, 3) \).

C. Output

If we carry out the program, these values are printed and plotted.

\( \alpha_j(1), \) evaluation results of \( ||(\alpha_j(1)I - D)v||_\infty \), \( \gamma, \) evaluation results of \( ||(\gamma I - D(\gamma_j))\xi||_\infty \), \( Y, g, p_n, p_r \), the cumulative distribution \((n = 0, \ldots, N - 1) \).

D. Computation

We show each computational step with the corresponding index number of sub-routines in our program.

Step 1

1) Computation of the eigenvalues of \( D \) (cf. Sub-routine[2]):

a) Transform \( D \) to upper Hessenberg form.
b) Compute all eigenvalues \( \alpha_j(1) \) \((i = 1, \ldots, m) \) of a real upper Hessenberg matrix \( D \). We use the original \( D \) in later computations.
c) For \( i = 1, \ldots, m \), set \( \alpha_i(1) = \Re(\alpha_i(1)) \) if \( |\Im(\alpha_i(1))| < \epsilon \).
d) Set \( \alpha_i(1) = 0 \) where \( i \) is the index such that \( \alpha_i(1) = \min(\alpha_j(1)) \) and swap \( \alpha_i(1) \) for \( \alpha_j(1) \).
e) If \( \alpha_i(1) \) \((i = 1, \ldots, m) \) are not simple then indicate error and stop the computation.

2) Evaluation of the precision of the eigenvalues \( \alpha_j(1) \) \((i = 1, \ldots, m) \) (cf. Sub-routine[2]):

a) For \( i = 1, \ldots, m \), compute the right eigenvector \( v \) of \( \alpha_j(1) \), compute \( ||(\alpha_j(1)I - D)v||_\infty \) and display its exponential part.
3) Computation of $\gamma_i(i=1,\ldots,m)$ (cf. Sub-routine[3]):
   a) Set $\gamma_i^{(0)} = \alpha_i(1)$ and $\alpha_j(z_i^{(0)}) = \alpha_j(1)$.
   b) For $s = 1,\ldots,$ slice,
      i) If $s < \text{slice}$ then set iteration $= \text{it}(1)$, else set iteration $= \text{it}(2)$.
      ii) If $s = 1$ then set $\phi(s^{-1})(\gamma_i^{(s^{-1})}) = 1$, else compute
          
          \[
          \phi(s^{-1})(\gamma_i^{(s^{-1})}) = (1-\alpha_i^j(z_i^{(s)})\alpha_j(z_i^{(s)})) \prod_{j=1,j\neq i}^{M}(\gamma_j^{(s^{-1})}-\alpha_j(z_i^{(s)}))\cdot
          \]
          where $\alpha_i^j(z_i^{(s)}) = h(s)(\gamma_i^{(s)})$, $\gamma_i^{(s)} = \alpha_i^j(z_i^{(s)})$ and $\alpha_j(z_i^{(s)}) = u_i^j(z_i^{(s)})v_i(z_i^{(s)})$.
      iii) Set $\alpha_j^{(s+1)} = \gamma_j^{(s)}$ and $\rho_j = \rho \times \frac{s}{\text{slice}}$.
      iv) For $v = 1,\ldots,$ iteration,
      A) For $i = 2,\ldots,m-1; \quad j = i+1,\ldots,m$,
         if $v \leq \text{it}(1)$ and $\alpha_i^{(s,v-1)} < \alpha_i^{(s,v-1)}$ distance then compute $\beta_i^{(s,v-1)}$ (res.
         $\beta_i^{(s,v-1)}$ with rotation angle $\theta = \frac{\alpha_i^{(s,v-1)}}{\alpha_i^{(s,v-1)}}$ and set $\alpha_i^{(s,v-1)} = \beta_i^{(s,v-1)}$ (res.
         $\alpha_i^{(s,v-1)} = \beta_i^{(s,v-1)}$).
      B) Compute
          \[
          \alpha_i^{(s,v)} = \alpha_i^{(s,v-1)} - \frac{\det(\alpha_i^{(s,v-1)} - D(h(s)(\alpha_i^{(s,v-1)})))}{\prod_{j=1,j\neq i}^{M}(\alpha_i^{(s,v-1)} - \alpha_j(z_i^{(s)}))} \times \phi(s^{-1})(\gamma_i^{(s)})^{-1}.
          \]
          If $|\alpha_i^{(s,v)} - \alpha_i^{(s,v-1)}| < \text{epsilon}$ for $\forall i$ and slice then go to (3(b)iv)
      C) If a computation error (e.g. overflow) occurs in (8) then
         a) Set $\alpha_j(z_i^{(s)}) = \alpha_j(z_i^{(s-2)})$ if $s \geq 2$.
         b) If we have this error at that $s$ for the first time then set $\phi(s^{-1})(\gamma_i^{(s)}) = 1$ and return to
            (3(b)ii), else set $\gamma_i^{(s)} = \gamma_i^{(s-1)}$ and go to the next $s$.
      v) Set last $v = \max(v \mid \alpha_i^{(s,v)} \exists)$.
      vi) If $|2(\alpha_i^{(s,last,v)}) - \alpha_i^{(s,v)}| < \text{epsilon}$ then set $\alpha_i^{(s,last,v)} = \mathcal{N}(\alpha_i^{(s,last,v)})$.
      vii) If $|\alpha_i^{(s,last,v)} + \delta > \delta$ then
         A) Set $\alpha_i^{(s,v-1)} = \alpha_i^{(s,v-2)}$ if $s \geq 2$.
         B) If we have this error at that $s$ for the first time then set $\phi(s^{-1})(\gamma_i^{(s)}) = 1$ and return to
            (3(b)ii), else set $\gamma_i^{(s)} = \gamma_i^{(s-1)}$ and go to the next $s$.
      viii) Set $\gamma_i^{(s)} = \gamma_i^{(s,last,v)}$.
      c) Set $\gamma_i^{(s)} = \gamma_i^{(s)}$.
      d) If $\gamma_i(i=1,\ldots,m)$ are not simple then indicate error and stop the computation.
   4) Evaluation of the precision of $\gamma_i(i=1,\ldots,m)$ and computation of $Y = (\eta_1,\ldots,\eta_m)$ (cf. Sub-routine[3]):
      a) For $i = 1,\ldots,m$, compute the right null vector $\eta_i$ of $(\gamma_iI - D(h(\gamma_i)))$, compute $||\gamma_iI - D(h(\gamma_i))\eta_i||_\infty$ and display its exponential part.
      b) Reset $\eta_1 = e$. Then the $m \times m$ matrix $(\eta_1,\ldots,\eta_m) \text{expresses } Y$.
   5) Computation of $g$ (cf. Sub-routine[5]):
      a) Compute $g = e_iY^{-1}$.
      b) For each element of $g$, the imaginary part of the element is replaced by 0.
      c) If there exists a negative element then indicate error and stop the computation.

Step 2

6) Computation of $p(\omega^k)$ (cf. Sub-routine[6]):
   a) Set $\alpha_i(\omega^{(0)}) = \alpha_i(1)$.
   b) For $k = 0,\ldots,N/2$,
      i) Compute $\omega^k = \exp(i2\pi k/N)$ and $D(\omega^k)$.
      ii) If $k > 0$ then compute $\alpha_i(\omega^k)(i=1,\ldots,m)$ by the D-K method with initial values
          $\alpha_i(\omega^{(k-1)})$.
      iii) Compute
          \[
          p(\omega^k) = \lambda^{-1}(1-\rho)g \times \sum_{i=1}^{m} \frac{\alpha_i(\omega^k)h(\alpha_i(\omega^k))}{\omega^k-h(\alpha_i(\omega^k))}v_i(\omega^k)u_i(\omega^k),
          \]
          where $\alpha_i(\omega^k)h(\alpha_i(\omega^k)) = \lambda \frac{\alpha_i(\omega^k)}{1-\rho}$ for $k = 0$ and
          $i = 1$.
      iv) If $1 \leq k \leq \frac{N}{2}$ then set $p(\omega^{N-k}) = p(\omega^k)$.
   7) Computation of $p_n$ (cf. Sub-routine[7]):
      a) For $n = 0,\ldots,N-1$, compute $p_n = \frac{1}{N} \sum_{k=0}^{N-1} p(\omega^k)\omega^{-kn}$. We use the inverse
         Fast Fourier Transform (FFT) to compute the above values.

8) End.

We came up with various ideas to make the algorithm faster, more precise and more stable. Some of these ideas are as follows.

Idea 1. (cf. Modeling parameters)
If we set $d_{ij}^{(k)} = Dx(i,j,k)$, we require a lot of computational time and a large amount of memory when $k$ is large. This may be wasteful if there exist some rate matrices whose all the elements equal to 0. Therefore, we define $Dx(i,j,l), \text{batch}(l)$ such that $d_{ij}^{(k)} = Dx(i,j,l)$ if $k = \text{batch}(l)$. Thus we can eliminate the waste.

Idea 2. (cf. 3(b)ivA)
In calculating $\alpha_i^{(s,v)}(i=1,\ldots,M)$ by the modified D-K method, there can exist some points at which two intermediate zeroes cause a head-on collision. Let us call them turning points because two real eigenvalues turn into two complex conjugate eigenvalues and the exact opposite phenomenon also occurs there. We found only the above two types of collision. The computation become unstable...
in neighborhoods of turning points, so we should add a device to our algorithm as follows.

Suppose that for \( j > i, |\gamma_i^{(s,v-1)} - \gamma_j^{(s,v-1)}| < \text{epsilon} \).

To overcome the difficulty, we rotate \( \gamma_i^{(s,v-1)} \) and \( \gamma_j^{(s,v-1)} \) with angle \( \theta \) at the center \( \gamma \equiv (\gamma_i^{(s,v-1)} - \gamma_j^{(s,v-1)})/2 \) in the complex plane, i.e.,

\[
\beta_i^{(s,v-1)} = (\alpha_i^{(s,v-1)} - \gamma)e^{i\theta} + \gamma, \\
\beta_j^{(s,v-1)} = (\alpha_j^{(s,v-1)} - \gamma)e^{i\theta} + \gamma.
\]

And we newly set \( \alpha_i^{(s,v-1)} = \beta_i^{(s,v-1)} \) and \( \alpha_j^{(s,v-1)} = \beta_j^{(s,v-1)} \). Graphical explanations of these concepts are given in Fig. 1.

**Idea 3.** (cf. 3(b)iVB)

As stated in the last paragraph of Section IV, we do not need numerically precise values in computing the intermediate zeros \( \gamma_i^{(s)} \) \((s = 1, \ldots, \text{slice}; i = 1, \ldots, m)\).

So if \( |\gamma_i^{(s,v)} - \gamma_j^{(s,v)}| < \text{epsilon} \) for \( \forall i(j = 1, \ldots, m) \), we escape from the loop. This leads to a reduction of computational time. Similar devices are used in 3(b)ii and 6(b)ii.

**Idea 4.** (cf. 3(b)ivC and 3(b)vi)

\( \phi^{(s-1)}(\gamma_j^{(s-1)}) \) in (7) is a revision term to keep stability of the iteration (8). But in some \( s \), this can indicate an unnecessary zero and cause a computational error (e.g., overflow). To deal efficiently with it, if an error occurs or intermediate zeros get out of \( \Omega \), we try computing again with \( \phi^{(s-1)}(\gamma_j^{(s-1)}) = 1 \). If the troubles occur again at that \( s \), we ignore all the computation at that \( s \) and go to the next \( s \).

Thus we can mostly overcome the difficulties in computing \( \gamma_i \) \((i = 1, \ldots, m)\). This brings us a great advantage with respect to efficiency and the number of models which we can adopt.

**VII. A NUMERICAL EXAMPLE**

In this section, we explain a numerical example on our program according to the variables mentioned in Section VI. All computations are performed in double precision using Decimal BASIC on the personal computer (CPU:2.0GHz, Memory:768MB).

We consider the sample example in [6] whose BMAP is estimated in [4] by modeling actual IP traffic measured by Wide Project. In [4], we divide the IP traffic into three levels and efficiently model it considering short-term and long-term dynamics. The parameters are set as follows: \( m = 9, b_{\text{num}} = 15, \rho = 0.3, \epsilon_{\text{epsilon}} = 10^{-15}, NN = 14, \text{slice} = 2, sp = -1, it(1) = 20, it(2) = 100, it(3) = 50 \). We carry out this model according to the concrete algorithms stated in the previous section. We required 4.4 seconds for computations in Step 1 and 649 seconds in Step 2.

The graph of stationary probability is given in Fig. 2 in order to compare the computed result with raw IP traffic. And the statistics of queue length are shown in Table I. The simulation results approximately agree with the numerical values of raw IP traffic. If we apply this method, we can efficiently estimate statistical values of various IP traffic and utilize them for constructing servers and networks which have high availability.

**Acknowledgement**

This research is strongly inspired by H.Tominaga and T.Shigeta. We refer to their way of thinking and some sub-routines to write our program. We would like to express our deep thanks to them.

**REFERENCES**