Accelerating Mean Time to Failure Computations

Philip Heidelberger
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
berger@watson.ibm.com

Jogesh K. Muppala
Dept. of Computer Science
University of Science and Technology
Clear Water Bay
Kowloon, Hong Kong
muppala@cs.ust.hk

Kishor S. Trivedi
Center for Adv. Comp. & Comm.,
Dept. of Electrical & Computer Engineering,
Duke University,
Durham, NC 27708-0291, USA
kst@ee.duke.edu

March 18, 1996

Abstract

In this paper we consider the problem of numerical computation of the mean time to failure (MTTF) in Markovian dependability and/or performance models. The problem can be cast as a system of linear equations which is solved using an iterative method preserving sparsity of the Markov chain matrix. For highly dependable systems, system failure is a rare event and the above system solution can take an extremely large number of iterations. We propose to solve the problem by dividing the computation in two parts. First, by making some of the high probability states absorbing, we compute the MTTF of the modified Markov chain. In a subsequent step, by solving another system of linear equations, we are able to compute the MTTF of the original model. We prove that for a class of highly dependable systems, the resulting method can speed up computation of the MTTF by orders of magnitude. Experimental results supporting this claim are presented. We also obtain bounds on the convergence rate for computing the mean entrance time of a rare set of states in a class of queueing models.

1 Introduction

This paper is concerned with numerical algorithms that accelerate the speed of computing the mean time to (first) failure (MTTF) in Markovian models of system dependability. Our motivation for studying this problem arose from the slow convergence rates we observed in computing the MTTF using the standard algorithms implemented in SAVE (System AVailability Estimator) [11, 12] and other packages (e.g., SPNP [6]). For example, consider a simple machine repairman model with one repairman, three types of components, and 3 components of each type. The system is considered operational if there is at least one operational component of each type, otherwise it is failed.

The Markov chain corresponding to this model has only 64 states, yet the iterative method used to compute the MTTF was very slow to converge. When the repair rate is fixed (at $\mu = 1.0$) while the failure rate, λ , of each component decreases, the number of iterations and CPU time required to obtain a fixed precision grows dramatically. For example, when $\lambda = 0.005$ the method required 1,680 iterations and took 0.46 seconds on an IBM RS/6000 workstation. However, when λ was decreased by one order of magnitude to 0.0005, the computational requirements increased by about three orders of magnitude to 1.8 million iterations and 468 seconds. By re-writing the MTTF as a ratio (see Equation 10), we are able to solve a slightly different set of equations that has much better convergence properties. In the above example, the improved method took only 20 iterations (the minimum possible in SAVE) and less than 0.03 seconds of CPU time for both $\lambda = 0.005$ and $\lambda = 0.0005$, an improvement of almost six orders of magnitude for the latter case. The ratio formula for the MTTF has been also used in [16] for proving limit laws for rare events in regenerative systems and in [20, 21] as the basis for effective simulation estimates for the MTTF using importance sampling.

In this paper, we prove that for regenerative systems, the method based on the ratio formula always converges faster than the standard method. In addition, we prove that the convergence rate for numerically computing the MTTF using the ratio formula actually improves for “SAVE-like” [20] systems as component failure rates approach zero, whereas the convergence rate degrades using the standard algorithm. For SAVE-like systems, the improved convergence can be attributed to the existence of a single state (the “all components operational” state) which dominates the steady-state distribution of the Markov chain. We call this basic algorithm the single state method. For other systems, there may not be such a dominating state. For those systems, we generalize the ratio formula to a matrix version involving multiple states. This generalization is guaranteed to improve the convergence rate, however, there is a trade-off since now multiple sets of equations must be solved; one for each state. We investigate this trade-off numerically. Our methods also apply to the computation of the derivative of the MTTF with respect to some input parameter. Similar computational savings are realized for such derivatives.

The single state method was implemented in SAVE in 1993 and Carrasco, Calderón and Escribá [2] have also empirically observed improvements by incorporating the single state method within a bounding technique. The multiple state method has been implemented in the Stochastic Petri Net Package (SPNP) [6].

2 Notation and Terminology

We assume that the system behavior can be represented by a homogenous finite-state continuous time Markov chain (CTMC) $\mathbf{Z} = \{Z(t), t \geq 0\}$ with state space Ω . We note that states of a CTMC will most often be vectors. However, the discrete state space of a CTMC can always be mapped into positive integers. Without loss of generality, we will therefore assume that the state space $\Omega = \{1, 2, \dots, M\}$, where M is the number of states in the CTMC. The infinitesimal generator matrix is given by $\mathbf{Q} = [Q(i, j)]$ where $Q(i, j), (i \neq j)$, represents the transition rate from state i to state j and the diagonal elements $Q(i, i) = -Q(i) = -\sum_{j \neq i} Q(i, j)$. Furthermore, $h(i) = 1/Q(i)$ is the mean holding time in state i , and $\mathbf{h} = [h(i)]$. We will assume that the CTMC $\mathbf{Z}$ is irreducible and since it is finite, all its states are positive recurrent.

Given the CTMC $\mathbf{Z}$, we can construct the embedded DTMC $\mathbf{X} = \{X(n), n \geq 0\}$ with the corresponding transition probability matrix $\mathbf{P} = [P(i, j)]$ where $P(i, i) = 0$, and $P(i, j) =$

$Q(i, j)/Q(i)$ for $j \neq i$. Since $\mathbf{Q}$ is irreducible, $\mathbf{P}$ will also be irreducible.

Given a matrix $\mathbf{M} = [M(i, j)]$ where $i, j \in \Omega$, and a set S , let ${}_S\mathbf{M}$ denote the “taboo probabilities” [5] corresponding to taboo set S , i.e., ${}_S\mathbf{M}(i, j) = M(i, j)$ for $j \notin S$ and ${}_S\mathbf{M}(i, j) = 0$ for $j \in S$. Let $\mathbf{M}'$ denote the transpose of $\mathbf{M}$ and let $\rho(\mathbf{M})$ denote the spectral radius of $\mathbf{M}$. Since a DTMC matrix is nonnegative, its spectral radius always corresponds to a positive real eigenvalue [14]. For a random variable Y defined over the state space of the CTMC, $E_i[Y]$ denotes the expectation of Y given that the starting state is i . For any set of states $S \subseteq \Omega$, let T_S denote the first hitting time for the set S , that is, the time at which the CTMC first enters a state in the set S . We shall be interested in computing $E_i[T_S]$ for different starting states i and subsets S ; it is well known [5] that sets of linear equations can be solved to compute $E_i[T_S]$:

$$E_i[T_S] = h(i) + \sum_{j \notin S} P(i, j) E_j[T_S] \quad i \notin S \quad (1)$$

For $i \in S$, we define $E_i[T_S] = 0$. Letting $t(i) = E_i[T_S]$, then in matrix form we have,

$$\mathbf{t} = \mathbf{h} + {}_S\mathbf{P} \mathbf{t}. \quad (2)$$

Here vectors $\mathbf{t}$ and $\mathbf{h}$ are column vectors; their transposes will be respectively denoted by $\mathbf{t}'$ and $\mathbf{h}'$.

As an alternative to Equation 2, we can solve a related set of equations to find $v(j) =$ the expected number of visits to state j before entering set S for a given, fixed, initial probability vector $\mathbf{p}_0$:

$$v(j) = \begin{cases} p_0(j) & j \in S \\ p_0(j) + \sum_{i \notin S} v(i) P(i, j) & j \notin S \end{cases} \quad (3)$$

which, in matrix notation, is

$$\mathbf{v}' = \mathbf{p}_0' + \mathbf{v}' {}_S\mathbf{P} \quad (4)$$

$$\mathbf{v} = \mathbf{p}_0 + {}_S\mathbf{P}' \mathbf{v}.$$

In the special case of a fixed starting state i , $p_0(i) = 1$ and

$$E_i[T_S] = \sum_j v(j) h(j). \quad (5)$$

Similarly, the probability of absorption to state j given $p_0(i) = 1$ is given by

$$H(i, j) = P\{Z(T_S) = j | Z(0) = i\} = \sum_k v(k) P(k, j). \quad (6)$$

Equation (3) also has CTMC counterpart [17]:

$$\tau(j) Q(j) = p_0(j) + \sum_{i \neq j} \tau(i) Q(i, j), \quad j \notin S \quad (7)$$

and $\tau(j) = 0$, if $j \in S$. Here $\tau(j)$ is the amount of time spent by the CTMC in state j until absorption.

Either Equation 2 or 4 is suitable for solution via matrix iterative methods (see [23]) which preserve sparsity. Assuming $\mathbf{P}$ is irreducible, then either Jacobi’s method or Gauss-Seidel iteration

applied to Equation 2 is guaranteed to converge, with Gauss-Seidel converging faster (Theorem 9.20 of [18]). The convergence of these algorithms depends on whether or not the spectral radii, say ρ_J and ρ_G of their respective iteration matrices are less than one. It is known that $\rho_G = \rho_J^2$. Successive over-relaxation (SOR) with a properly chosen parameter can further accelerate convergence, although computing bounds on SOR's iteration matrix is difficult. Thus, in this paper, we will usually derive bounds on ρ_J . Our numerical examples do use SOR, when appropriate.

For some power k of ${}_S\mathbf{P}^k$ will have a maximum row sum, $a(k, S)$, strictly less than one, which serves as an upper bound on the spectral radius $\rho({}_S\mathbf{P}^k)$. Thus the asymptotic convergence rate is bounded by $a(k, S)^{1/k}$, i.e., the error after n iterations is bounded by a constant times $a(k, S)^{n/k}$. Furthermore, since $\rho({}_S\mathbf{P}) = \rho({}_S\mathbf{P}')$, the convergence rates of iterative schemes applied to Equations 2 or 4 will be the same. We will usually find it more convenient to use Equations of the form 4.

3 Standard Computation of the MTTF for Highly Dependable Systems

We assume that the state space Ω can be partitioned into two subsets: $\Omega = O \cup F$, where O represents the set of up states in the system, i.e., the set of states in which the system is operational, and F represents the set of failed states. We further assume that the system starts out in the state in which all components are operational, and we label this state as 1. In particular, we are interested in $E_1[T_F]$, the first entrance time into the subset F of failed states, or the MTTF. Setting $S = F$ in Equation 2 yields a direct set of equations for computing the MTTF. However, for SAVE-like systems, we will show that convergence is very slow: the spectral radius approaches one as the failure rates approach zero.

Following [20] we first define SAVE-like systems as follows:

1. There are n components that can fail and be repaired. The maximum rate of any failure transition is ϵ , and failure transitions can cause multiple components to fail simultaneously.
2. At least one repair transition emanates from a state with at least one failed component. The rate of all repair transitions is bounded below by a positive constant $\underline{\mu}$.
3. The CTMC with the generator matrix $\mathbf{Q}$ is irreducible (and thus positive recurrent).
4. In addition to the assumptions in [20], we assume there is not a direct transition from the state 1 to any state in F .

We assume that the system is characterized by a “rarity parameter” ϵ and let $\mathbf{P}(\epsilon)$ denote the transition probability matrix of the corresponding embedded DTMC. We will assume that $\lim_{\epsilon \rightarrow 0} \mathbf{P}(\epsilon) = \mathbf{R}$. $\mathbf{R}$ is a transition probability matrix. Thus we also have $\lim_{\epsilon \rightarrow 0} {}_F\mathbf{P}(\epsilon) = {}_F\mathbf{R}$. By the above assumptions, in particular since there are no direct transitions from 1 into F , $\sum_{j \in O} {}_F\mathbf{R}(i, j) = 1$ for all $i \in O$, i.e., ${}_F\mathbf{R}$ is a transition probability matrix when restricted to operational states. Let C denote the set of states that state 1 communicates with (under matrix ${}_F\mathbf{R}$). Because of the above assumptions, C forms an irreducible class that includes 1 and some subset of the operational states; C contains only those states reachable from 1 (C_1) and states along repair paths back to 1 from states in C_1 . We will further assume that the limiting DTMC is aperiodic on C , and thus it has a steady state solution $\pi(C)$. Extending this solution to all of Ω

by setting $\pi(i) = 0$ for $i \notin C$, we thus have $\pi'_F \mathbf{R} = \pi'$, i.e., $_F \mathbf{R}$ has an eigenvalue equal to one (and left eigenvector π'). Thus, since eigenvalues of a matrix are continuous functions of its elements [7], there is an eigenvalue of $_F \mathbf{P}(\epsilon)$ that approaches one as $\epsilon \rightarrow 0$. Since $\rho(_F \mathbf{P}(\epsilon)) \leq 1$, $\rho(_F \mathbf{P}(\epsilon)) \rightarrow 1$ as $\epsilon \rightarrow 0$. Thus the efficiency of the standard method for computing the MTTF degrades as failure rates approach zero. This will be true whether equations of the form 2 or 4 are used.

4 Accelerated Convergence for Computing the MTTF

We will use Dynkin's Formula (page 297 of [15]) as the basis for accelerating the convergence of the MTTF. Pick a subset of states $A \in O$ that includes 1 and let $B = F \cup A$. We will think of A as being a small, but probabilistically significant, set. For convenience, we will denote $C = \Omega - B$. Then, for any state $i \in O$, $T_B \leq T_F$. Note that $T_B = \min(T_A, T_F)$. Thus we can write $T_F = T_B + (T_F - T_B)1_{\{T_B < T_F\}}$ where $1_{\{E\}}$ is the indicator of the set E . Applying the Markov property at time T_B , we can write a special case of Dynkin's Formula

$$E_i[T_F] = E_i[T_B] + \sum_{j \in A} H(i, j) E_j[T_F] \quad i \in A \quad (8)$$

where for $i \in O$ and $j \in A$, $H(i, j) = P[Z(T_B) = j | Z(0) = i]$. Let $y(i) = E_i[T_F]$, $i \in O$ and $t(i) = E_i[T_B]$, $i \in O$. We are seeking to compute the MTTF $= y(1)$. Then we have the system of equations

$$y(i) = t(i) + \sum_{j \in A} H(i, j) y(j) \quad i \in A. \quad (9)$$

If $A = \{1\}$, then Equation 9 can be reduced to the ratio formula

$$MTTF = E_1[T_F] = \frac{E_1[\min(T_F, T_1)]}{P_1\{T_F < T_1\}}. \quad (10)$$

Restricting Equation 9 to states $i \in A$, we see that the MTTF, $y(1)$ can be found by solving a set of $|A|$ linear equations provided $t(i)$ and $H(i, j)$ are known for each $i, j \in A$.

The equations to compute $\mathbf{t}$ and $\mathbf{H}$ can be found in Section 2. Although we could work with either Equations 2 or 4 we will find it advantageous to use the transposed form 4 (as the number of sets of equations that need to be solved will be reduced by one set). Set $S = A \cup F$. For each $i \in A$, obtain $t(i)$ and $H(i, j)$ by setting $p_0(i) = 1$, solving Equation 4 and using Equations 5 and 6. Thus in the method we solve

1. $|A|$ sets of equations of the form 4, and
2. one set of equations of the form 9.

Assuming $|A|$ is small, most of the work is in step 1.

As we add states to A , the rate of convergence of the iteration increases since if $A \subset A'$, then $_{A'} \mathbf{P} \leq _A \mathbf{P}$ (with strict inequality for at least one entry) and thus of $\rho(_{A'} \mathbf{P}) < \rho(_A \mathbf{P})$. However, this increased convergence rate must be balanced against the cost of solving an extra set of linear equations for each state added to A . Although simple cost models can be derived to address this question theoretically, these models involve unknown terms such as $\rho(_A \mathbf{P})$ and therefore may not be of much practical use.

If $|A| = 1$, i.e., $A = \{1\}$, we will refer to the method as the single state method. The convergence rate of the single state method is always better than the original method. In this case, only one system of equations needs to be solved, just as in the original method. So the work of the single state and original methods are the same, except for added work of computing $H(1, 1) = P_1(T_1 < T_F)$ in Equation 6; this extra work is only linear in the size of the state space and is thus negligible compared to the cost of applying iterative methods to solve Equation 4.

The method as described above assumes that the initial probability vector of the Markov chain is concentrated on the single state 1. If this is not the case and we are interested in solving $E_{\mathbf{p}_0}[T_F]$, where the subscript $\mathbf{p}_0$ indicates that the initial probability vector is given by an arbitrary probability vector $\mathbf{p}_0$. Then $|A| + 1$ sets of equations of the form 4 need to be solved: one set corresponding to $\mathbf{p}_0$, and an additional set with $p_0(i) = 1$ for each state $i \in A$. Thus, in this case, the single state method requires solving two sets of equations.

4.1 Block Matrix Formulation of the Method

The new method can also be derived from a block-matrix approach. To do this, let $y(i) = E_i[T_F]$ as defined above, and let $\mathbf{y} = [y(i)]$ be the corresponding vector. Further, let $D = C \cup A$, where C and A are defined in the previous section. Let $\mathbf{P}_{DD}$ be the $\mathbf{P}$ matrix restricted to only those states in D . Substituting $\mathbf{y}$ and $\mathbf{P}_{DD}$ into Equation 2 we obtain:

$$\mathbf{y} = \mathbf{h} + \mathbf{P}_{DD}\mathbf{y} \quad (11)$$

We can decompose $\mathbf{P}_{DD}$ into its constituent blocks as follows:

$$\mathbf{P}_{DD} = \left[\begin{array}{c|c} \mathbf{P}_{CC} & \mathbf{P}_{CA} \\ \hline \mathbf{P}_{AC} & \mathbf{P}_{AA} \end{array} \right]. \quad (12)$$

Substituting Equation 12 into Equation 11 we get:

$$[\mathbf{y}_C | \mathbf{y}_A] = [\mathbf{h}_C | \mathbf{h}_A] + \left[\begin{array}{c|c} \mathbf{P}_{CC} & \mathbf{P}_{CA} \\ \hline \mathbf{P}_{AC} & \mathbf{P}_{AA} \end{array} \right] [\mathbf{y}_C | \mathbf{y}_A] \quad (13)$$

or alternately

$$\mathbf{y}_C = \mathbf{h}_C + \mathbf{P}_{CC}\mathbf{y}_C + \mathbf{P}_{CA}\mathbf{y}_A \quad (14)$$

and

$$\mathbf{y}_A = \mathbf{h}_A + \mathbf{P}_{AA}\mathbf{y}_A + \mathbf{P}_{AC}\mathbf{y}_C \quad (15)$$

We can rewrite Equation 14 as:

$$\mathbf{y}_C = (\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{h}_C + (\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{P}_{CA}\mathbf{y}_A \quad (16)$$

Substituting Equation 16 into Equation 15 we get:

$$\mathbf{y}_A = (\mathbf{h}_A + \mathbf{P}_{AC}(\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{h}_C) + (\mathbf{P}_{AA} + \mathbf{P}_{AC}(\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{P}_{CA})\mathbf{y}_A \quad (17)$$

Comparing Equation 17 to Equation 9, we note that,

$$\mathbf{t} = (\mathbf{h}_A + \mathbf{P}_{AC}(\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{h}_C) \quad (18)$$

and

$$\mathbf{H} = (\mathbf{P}_{AA} + \mathbf{P}_{AC}(\mathbf{I} - \mathbf{P}_{CC})^{-1}\mathbf{P}_{CA}) \quad (19)$$

where $\mathbf{t} = [t(i)]$ and $\mathbf{H} = [H(i, j)]$. In the above expression $\mathbf{H}$ is referred to as the Schur complement [8, 10] of $\mathbf{P}_{CC}$ in $\mathbf{P}_{DD}$. This development is also used in [4] to relate conditional MTTF with the unconditional one. Computing the visit counts using Equation 4 is equivalent to computing $\mathbf{P}_{AC}(\mathbf{I} - \mathbf{P}_{CC})^{-1}$ in the above equations. Furthermore, Equations 18 and 19 are equivalent to Equations 5 and 6 respectively.

4.2 Improved Convergence for SAVE-like Systems

In this section, we show that the efficiency of the single state method actually improves as component failure rates go to zero in a SAVE-like system. Let $A = \{1\}$ and $B = A \cup F$ and let ϵ denote the maximum failure rate.

Proposition 1 *For a SAVE-like system defined by assumptions 1-4 of Section 3,*

$$\lim_{\epsilon \rightarrow 0} \rho(B\mathbf{P}(\epsilon)) = 0. \quad (20)$$

Proof: Consider the row sums of $B\mathbf{P}(\epsilon)^k$ for some power k :

$$\sum_{j \notin B} P(\epsilon, i, j)^k = P_i\{T_B > k\} \leq P_i\{T_1 > k\}. \quad (21)$$

Now, for any starting state i , there is a d_i such that if $T_1 > d_i$, there must be at least one failure transition in the first d_i transitions. Since the maximum failure rate is ϵ , $P_i\{T_1 > k\} \leq c_i\epsilon$ for any $k \geq d_i$ and some constant c_i . Letting $d = \max\{d_i\}$ and $c = \max\{c_i\}$, we have $P_i\{T_1 > d\} \leq c\epsilon$ for all i . Since $\rho(B\mathbf{P}(\epsilon)^d)$ is less than the maximum row sum, $\rho(B\mathbf{P}(\epsilon)^d) \leq c\epsilon$. $\square$

A similar, but algebraic, argument applied to a somewhat different matrix was used in [20] to establish the leading term (in powers of ϵ) in the expansion for $P_1\{T_F < T_1\}$.

4.3 Convergence Rate for a Class of Queueing Systems

In this section, we derive a bound on the convergence rate of the single state method for a class of queueing system models. We also show that, in the time that the single state method has converged, the error in Jacobi's method (with a bounded starting solution) grows arbitrarily large as the MTTF increases. For Gauss-Seidel we show that the error after any fixed number of iterations grows arbitrarily large as the MTTF increases.

We assume that the CTMC is positive recurrent on an infinite state space and we consider a sequence of problems indexed by n , which may be loosely thought of as the total queue length in the system. Let O_n denote a finite subset of "up (operational)" states and let F_n denote the set of failure states. Let 1 denote the initial state and set $B_n = \{1\} \cup F_n$. We are interested in computing $E_1[T_{F_n}]$. In contrast to the SAVE-like systems, here the rate matrix is fixed and the set of failure states changes. We make the following assumptions:

1. $E_i[T_1] \leq cn$ for all $i \in O_n$ and a constant $c < \infty$.
2. $\lim_{n \rightarrow \infty} E_1[T_{F_n}]/n^{1+\delta} = \infty$ for some $\delta > 0$.

These are reasonable assumptions in many situations. Assumption 1 says that the relaxation time to return to 1 from any state in O_n is at most linear. This is true, for example, in stable Jackson networks starting with initial queue lengths proportional to n , since such saturated queues satisfy a fluid limit [3] which tend to the origin. Assumption 2 sets a lower bound on the MTTF; in many systems, the polynomial lower bound of Assumption 2 is too conservative. Often, the MTTF grows exponentially, as it does in a tandem Jackson network of single server queues where T_{F_n} is the first time that the total queue length reaches or exceeds n [9].

Let $t_n(m, i)$ be the estimate of $E_i[T_{F_n}]$ after m iterations of Jacobi's method applied to Equation 2 with starting values $t_n(0, i)$. We are interested in comparing $t_n(m, 1)$ with $E_1[T_{F_n}]$. Recall that $h(i)$ is the mean holding time in state i .

Proposition 2 *Let $m_n = 1/n^{1+\delta}$.*

1. *Under assumption 1 above,*

$$\lim_{n \rightarrow \infty} \rho(B_n \mathbf{P}^{m_n}) = 0. \quad (22)$$

2. *If $h(i) \leq B$ and $|t_n(0, i)| \leq B < \infty$ for all $i \in O_n$, then under assumption 2*

$$\lim_{n \rightarrow \infty} t_n(m_n, i)/E_i[T_{F_n}] = 0. \quad (23)$$

Proof:

To prove Part 1,

$$\begin{aligned} P_i\{\min(T_{F_n}, T_1) > m_n\} &\leq P_i\{T_1 > m_n\} \\ &\leq E_i[T_1]/m_n \leq cn/m_n \rightarrow 0 \end{aligned} \quad (24)$$

by Assumption 1. However the left hand side of Equation 24 is the row sum of $B_n \mathbf{P}^{m_n}$. Since $\rho(B_n \mathbf{P}^{m_n})$ is less than the maximum row sum, this proves Part 1.

To prove Part 2, note that after m iterations of Jacobi's method,

$$\mathbf{t}_n(m) = \sum_{k=0}^{m-1} F_n \mathbf{P}^k \mathbf{h} + F_n \mathbf{P}^m \mathbf{t}_n(0) \leq B(m+1) \quad (25)$$

and the result follows by assumption 2. $\square$

Part 1 of Proposition 2 states that to achieve a given accuracy, the single state method requires only slightly more than a linear (in n) number of iterations, regardless of how rare F_n is. Part 2 states that by the time the single state method has converged, Jacobi's method hasn't even started estimating the MTTF.

Part 2 of Proposition 2 is not as strong as showing that $\rho(F_n \mathbf{P}^{m_n}) \rightarrow 1$. However, to prove such a result appears to require more delicate arguments involving bounds on eigenvalues and eigenvectors associated with ever increasing state spaces, a topic beyond the scope of this paper. If we assume that $\lim_{n \rightarrow \infty} P_1\{T_{F_n} < T_1\} = 0$, i.e., hitting F_n during a regenerative cycle starting at 1 is rare, then Keilson [16] shows that $T_{F_n}/E_1[T_{F_n}] \Rightarrow EXP(1)$ where $\Rightarrow$ denotes convergence in distribution and $EXP(1)$ denotes an exponential random variable with mean 1. Combined with Assumption 2, this implies that $P_1\{T_{F_n}/m_n \geq 1\} \rightarrow 1$, or, for a given ϵ and large enough n ,

$$P_1\{T_{F_n} \geq m_n\} \geq (1 - \epsilon). \quad (26)$$

While this suggests that $\rho(F_n \mathbf{P}^{m_n}) \rightarrow 1$, proving this using, say, spectral expansions requires obtaining bounds on the eigenvectors which cannot be considered as constants since the size of the state space is increasing. Equation 26 does show that the maximum row sum of $F_n \mathbf{P}^{m_n}$ is close to one. Thus an upper bound on $\rho(F_n \mathbf{P}^{m_n})$ is greater than $(1 - \epsilon)$, but what we need is a lower bound.

We close this section by considering the error of Gauss-Seidel iteration applied to the original system of Equations $\mathbf{t} = \mathbf{h} + F_n \mathbf{P} \mathbf{t}$. Let $\mathbf{t}_n(m)$ be the solution after m iterations. Gauss-Seidel iteration can be written in the form

$$\begin{aligned} \mathbf{t}_n(m+1) &= \mathbf{h} + \mathbf{L}_n \mathbf{t}_n(m+1) + \mathbf{U}_n \mathbf{t}_n(m) \\ &= (\mathbf{I} - \mathbf{L}_n)^{-1} [\mathbf{h} + \mathbf{U}_n \mathbf{t}_n(m)] \end{aligned} \quad (27)$$

where $\mathbf{L}_n$ is lower diagonal, $\mathbf{U}_n$ is upper diagonal and $\mathbf{L}_n + \mathbf{U}_n = F_n \mathbf{P}$.

Proposition 3 *If*

1. $\sum_{\{j:j < i\}} P(i, j) \leq \beta < 1$ for all $i \in O_n$,
2. $h(i) \leq B$ and $|t_n(0, i)| \leq B < \infty$ for all $i \in O_n$, and
3. $\lim_{n \rightarrow \infty} E_1[T_{F_n}] = \infty$,

then $\lim_{n \rightarrow \infty} t_n(m, i)/E_1[T_{F_n}] = 0$.

Proof: Since $(\mathbf{I} - \mathbf{L}_n)^{-1} = \sum_{k=0}^{\infty} \mathbf{L}_n^k$, assumption 1 of the proposition ensures that $(\mathbf{I} - \mathbf{L}_n)^{-1} < 1/(1 - \beta)$ (element-wise). Thus from the recursion in Equation 27, a simple induction argument establishes that

$$|t_n(m, i)| \leq \frac{(m+1)B}{(1 - \beta)^m} \quad (28)$$

from which the result follows by letting $n \rightarrow \infty$, keeping m fixed. $\square$

Thus for any finite number of iterations, the error in Gauss-Seidel increases as the MTTF increases. However, the bound in Inequality 28 is rather crude and does not appear to yield useful insight if m increases along with n . Assumption 1 of this proposition merely states that, under the ordering of states implicit in Gauss-Seidel, the probability of going to a higher numbered state is bounded away from zero. This is reasonable in many queueing networks; for example if the total arrival rate is bounded away from zero and the total service rate is bounded from above by a constant.

4.4 Sensitivity Analysis

It is frequently of interest to compute the sensitivity of output performance measures with respect to changes in input parameters, e.g., to compute the derivative of the MTTF with respect to a component failure rate [1]. Suppose the system is parameterized by θ (say a failure rate), and we wish to compute $\frac{\partial}{\partial \theta} E_1[T_F]$. Differentiating Equation 2, we obtain

$$\begin{aligned} \frac{\partial}{\partial \theta} \mathbf{t} &= \frac{\partial}{\partial \theta} \mathbf{h} + S \mathbf{P} \left(\frac{\partial}{\partial \theta} \mathbf{t} \right) + \left(\frac{\partial}{\partial \theta} S \right) \mathbf{P} \mathbf{t} \\ &= \mathbf{x} + S \mathbf{P} \frac{\partial}{\partial \theta} \mathbf{t} \end{aligned} \quad (29)$$

where $\mathbf{x} = \frac{\partial}{\partial \theta} \mathbf{h} + \left(\frac{\partial}{\partial \theta} {}_S\mathbf{P} \right) \mathbf{t}$. Thus, once $\mathbf{x}$ is computed (which involves solving Equation 2 for $\mathbf{t}$ and differentiating $\mathbf{h}$ and ${}_S\mathbf{P}$), the equations for the derivatives have the same form as the equations for the MTTF, with the same matrix ${}_S\mathbf{P}$. Thus the same acceleration techniques can be applied to computing the derivative of the MTTF and the same improvement in efficiency is realized for SAVE-like systems.

4.5 Multiple State Heuristics

Selecting the appropriate set of states to include in A is still an open problem. In our experience, we found that for SAVE-like systems where the system behavior is mainly dominated by a single state, the convergence obtained by using the single state in A is fast. Further improvement is rarely obtained by adding more states to A . However, when no single state is dominating the steady-state distribution, as we will see in some examples, addition of states to A yields faster convergence. In general, we found that adding those states to A which have the highest steady-state probability tends to yield faster convergence. (Such information is available provided the steady-state equations are solved first. Since users are generally also interesting in steady-state behavior, this will often be the case.) However, this gain in convergence is often offset by the added time required for solving multiple sets of linear equations. In both the examples that we present in section 5.2, adding one state to A reduces the computation time by an order of magnitude, but adding two states to A (i.e., $|A| = 3$) does not yield any better performance, than adding only one state.

5 Numerical Results

We use either near-optimal successive over-relaxation (SOR) [6, 22] or Gauss-Seidel (GASE) [10] to solve the $|A|$ systems of linear equations. For some problems, near-optimal SOR provides faster convergence than GASE, but for cases where GASE is faster, we have chosen to use GASE. To solve for $\mathbf{y}$ in Equation 9 we implemented the Gaussian elimination method. We note that in general the $\mathbf{H}$ matrix tends to be dense, but the matrix is usually of small dimensions ($|A| \times |A|$). In this case, it is beneficial to solve this matrix using Gaussian elimination rather than using an iterative method.

We test for convergence in the iterative methods by examining successive iterates. If $\mathbf{x}^i$ is the iteration vector at the end of the i th iteration, then given a precision ε to be attained, we check if $(\max_j |x^i(j) - x^{i-1}(j)| / \max_j |x^i(j)|) < \varepsilon$. The iterative method exits when this condition is satisfied.

5.1 The Single State Method

As the first example, we consider a machine repairman model of a system consisting of m components. The components suffer failure where the time to failure is exponentially distributed with rate λ . A single repair person attends to the repair of the failed components, where the time to repair is exponentially distributed with rate μ . The structure of the underlying CTMC is illustrated in Figure 1. (Note that we have deviated from our earlier notation and let m be the fully up state rather than state 1.) In this system, we consider the state with all components failed as the system failure state, and compute the mean time to (first) failure. While this model is simple, it serves to illustrate the gain in speed accrued by using the new method. We compare the performance of

Figure 1: CTMC for the machine repairman model.

both the original and the single-state method in Table 1. These results were computed using the SPNP package in which both the original method and the single-state method are implemented. In this table, $\mu = 1.0 \text{ sec}^{-1}$. All execution times shown in the table are in seconds, measured on a SUN SPARC IPX workstation. We used near-optimal SOR for iterative solution in this example. Notice that for $m = 2$, the single state method always converges in two iterations. This is because after two transitions, the Markov chain either enters the starting state (state 2) or the failure state (state 0). For $m = 4$, the iteration count of the single state method decreases as λ decreases, in accordance with theory. The iteration count for the original method increases as λ decreases, also in accordance with the theory.

Parameters			Original Method		Single State Method	
m (States)	λ	MTTF	Iterations	CPU Time	Iterations	CPU Time
2 (3)	0.1	6.50×10^1	107	0.25	2	0.19
	0.01	5.15×10^3	300	0.23	2	0.16
	0.001	5.01×10^5	1740	0.24	2	0.18
	0.0001	5.00×10^7	17,224	0.59	2	0.20
	0.00001	5.00×10^9	190,431	4.06	2	0.19
4 (5)	0.1	7.20×10^2	579	0.28	16	0.19
	0.01	4.38×10^6	311,289	11.9	7	0.16
	0.001	4.19×10^{10}	49,147,284	1840	4	0.16

Table 1: Convergence properties in the machine repairman problem.

For the second example, we extend the machine repairman model of Example 1 by assuming that the time to failure of the components has a two stage Erlang distribution with parameter λ . The structure of the underlying CTMC is illustrated in Figure 2. In this figure, the state label (i, j) corresponds to the system state in which i components are still functioning, and j components are in the second stage of their failure. The performance of both the original and the single-state method are shown in Table 2. In this table, $\mu = 1.0 \text{ sec}^{-1}$. All execution times shown in the table are in seconds, measured on a SUN SPARC IPX workstation. We used the SPNP package and near-optimal SOR as the iterative solution method. In this example, the initial state is $(m, 0)$ which corresponds to all components up and in the first phase of service. For $m = 2$, $\pi(m, 0) = 0.25$ for $\lambda = 0.00001$, while for $m = 4$, $\pi(m, 0) = 0.06$ for $\lambda = 0.01$. Thus the steady state probability vector is not dominated by a single state as is the case for SAVE-like systems. Yet the single-state

Figure 2: CTMC of the machine repairman model with 2 stage Erlang failure distribution.

Parameters			Original Method		Single State Method	
m (States)	λ	MTTF	Iterations	CPU Time	Iterations	CPU Time
2 (6)	0.1	2.21×10^2	110	0.22	19	0.20
	0.01	2.02×10^4	515	0.25	20	0.20
	0.001	2.00×10^6	3,930	0.38	20	0.20
	0.0001	2.00×10^8	43,651	2.18	20	0.20
4 (15)	0.1	8.29×10^3	3,378	0.67	74	0.25
	0.05	1.18×10^5	26,492	3.55	64	0.22
	0.01	6.81×10^7	1,731,956	218	69	0.29

Table 2: Convergence properties for computing the MTTF in the machine repairman problem with Erlang(2) failure distributions and m components of one type.

method still produces significant cost savings.

As a third example, we extend the machine repairman model of Example 1 by considering three types of components, each type exhibiting behavior similar to the ones in Example 1. Further we consider m components of each type. The system is considered operational as long as at least one component of each type is operational. The initial state is the all components operational state. Thus states in which all components of any type are failed, are designated as system failure states. The performance of both the original and the single-state method are shown in Table 3. In this table, $\mu = 1.0 \text{ sec}^{-1}$. All execution times shown in the table are in seconds, measured

Parameters			Original Method		Single State Method	
m (States)	λ	MTTF	Iterations	CPU Time	Iterations	CPU Time
2 (27)	0.001	1.67×10^5	1738	0.38	5	0.28
	0.0001	1.67×10^7	17,219	1.24	4	0.24
	0.00001	1.67×10^9	190,425	18.2	3	0.23
	0.000001	1.67×10^{11}	1,391,088	131	3	0.27
3 (64)	0.01	5.14×10^4	9,020	3.25	11	0.38
	0.005	4.27×10^5	42,899	15.0	9	0.32
	0.001	5.51×10^7	769,035	248	6	0.35
	0.0005	4.43×10^8	2,456,708	774	5	0.34

Table 3: Convergence properties for computing the MTTF in the machine repairman problem with three types of components and m components of each type.

on a SUN SPARC IPX workstation. We used the SPNP package and near-optimal SOR as the iterative solution method. In agreement with the theory for SAVE-like systems, the performance of the original method degrades whereas the performance of the single state method improves as the failure rates approach zero.

As mentioned in Section 4.4, the method can also be applied to compute derivatives of the MTTF. In accordance with theory, we have observed similar savings for computing derivatives. For example, to compute the derivative of the MTTF with respect to λ for the above model in Table 3 with $m = 3$ and $\lambda = 0.0005$ (the last row) required close to 2 million iterations and 528 seconds of CPU time using the original method (not including the time to compute the MTTF). The single state method took 20 iterations and 0.04 seconds. These derivative results were obtained using the SAVE package running on an IBM RS/6000.

Our fourth example is from the performance modeling domain. We consider a system consisting of N terminals connected to computing system consisting of multiple CPUs and multiple disks. Jobs are submitted from the terminals and they require some processing on a CPU and an access to a disk before completing execution. Upon completion of a job, the user at the terminal submits a new job after some time has elapsed (the think time). We assume that the think time is exponentially distributed with mean $1/\lambda$ seconds. The service rates of a CPU and a disk are μ_1 and μ_2 respectively. We assume that the system contains S CPUs and S disks respectively. This system is modeled by the closed queueing network shown in Figure 3. The system starts with all the terminals in the thinking (free) state. For this system, we compute the mean time until there are no terminals in

Figure 3: The closed queueing network model of the computer system.

the thinking state. The performance of both the original and the single-state method are shown in Table 4. The results in this table were computed by setting $S = 3$, $\mu_1 = 10 \text{ sec}^{-1}$ and $\mu_2 = 1$

Parameters			Original Method		Single State Method	
N (States)	λ	MTTF	Iterations	CPU Time	Iterations	CPU Time
5 (27)	0.5	6.35×10^1	2124	0.34	189	0.10
	0.2	3.11×10^3	45,526	4.58	60	0.10
	0.1	1.08×10^5	701,450	69	32	0.10
10 (77)	0.5	1.60×10^2	7,525	2.51	2,047	0.79
	0.2	1.22×10^5	2,306,330	723	202	0.24
	0.1	7.96×10^7	9,416,577	2,959	69	0.22

Table 4: Convergence properties for computing the MTTF for the closed queueing system.

sec^{-1} . All execution times shown in the table are in seconds, measured on a SUN SPARC 1000 workstation. We used the SPNP package and GASE as the iterative solution method. We again see substantial improvements as λ decreases, although this is not a “SAVE-like” system.

A modification of the above model has been studied in the context of rare event simulations [9, 19]. We consider an open tandem network with two single server queueing stations with service rate μ_i at station i and arrival rate λ . The rare event of interest is the total number of jobs in the system reaching or exceeding n . The initial state is the system empty state. This model satisfies the assumptions of Proposition 2. For this model, with utilization $\rho_i = \lambda/\mu_i = 0.5$ at each, we also observe large improvements from using the single state method. For $n = 5, 10, 15$, and 20 , the original method takes 6.7, 52, 701, and 11,714, respectively, times as many iterations as the single state method. For example, with $n = 20$ the iteration counts are 404 and 4,732,462.

The next example is the fault tolerant database system model adapted from [13]. The model consists of 2 sets each of replicated processors, and disk controllers, and six RAID disk arrays. Each disk array consists of four disks. The data on each disk are replicated, such that one third of the data resides on each of the other three disks in the disk array. The disk array can withstand

Figure 4: A block diagram of the database system.

failure of one disk. The disk arrays are divided into two groups of 3 disk arrays each. Each group is connected to one set of disk controllers. The database system is considered to be operational if all the data is accessible to both the processor sets. This implies that the system is operational if at least one processor in each processor set, one disk controller in each disk controller set and at least three disks in each disk array are operational. A processor suffers failure at the rate λ_P . When the processor in one set fails, it has a small probability of p of causing an operating processor in the other set to fail. The disk controllers suffer failure at the rate λ_C , and a disk suffers failure at the rate λ_D respectively. Failed components are repaired at the rate μ by a single repair person who selects any one of the failed components with equal probability. This model does contain “failure propagation”, i.e., simultaneously failing components, so there are rather short paths to the failure set. The initial state has all components operational. We first compute the results for a balanced system [13] which consists of two processors per processor set, and two disk controllers per disk controller set. A balanced system is one in which each type of component has the same amount of redundancy (i.e., the same number of components of a type must fail in order for the system to fail) and all components have failure rates of the same order of magnitude [13]. We assume $\lambda_P = 1/2000$ per hour, $\lambda_C = 1/2000$ per hour, and $\lambda_D = 1/6000$ per hour, $p = 0.01$, and $\mu = 1.0$ per hour. The CTMC had several hundred thousand states, with about 1000 non-failed states. The mean time to failure was 2.5×10^5 hours. The original method required 4,628 iterations and 89.6 seconds, while the single state method converged in 6 iterations in 15.92 seconds. All times are measured on a SUN SPARC 1000 workstation using the SPNP package and near-optimal SOR as the iterative solution method. These times include a constant time overhead used by SPNP in generating the large Markov state space from a stochastic Petri net description of the system behavior. Similar savings were also observed in computing the derivative of the MTTF.

We also consider an unbalanced system which contains four processors per processor set and a processor failure rate $\lambda_P = 1/1000$ per hour. The failure rates of the remaining components are decreased by a factor of ten. In this system, although the processors are more likely to fail, they are less likely to cause system failure due to the redundancy. The Markov chain contained several hundred thousand states, and about 4200 non-failed states. The mean time to failure was 3.2×10^7 hours. The original method required 182,561 iterations and 12,370 seconds, while the single state method converged in 6 iterations in 138.05 seconds (including loading). All times are measured on a SUN SPARC 1000 workstation using the SPNP package and near-optimal SOR as the iterative solution method.

For all the examples considered so far, we have shown that the single state method performs much better than the original method. Indeed, we notice that while the performance of the original method tends to degrade with decreasing failure rates, the performance of the single state method improves as expected. In most situations where the single state method yields rapid convergence, we do not see any need for using the multiple state method. In some situations, the multiple state method yields better results, as will be illustrated in the next subsection.

5.2 The Multiple State Method

As the first example of the multiple state method, we reconsider Example 2 above (m processors with two stage Erlang failure distributions). We repeat the experiments for the system by consecutively adding states to the set A based on their steady-state probabilities. The results for the model are given in Table 5. All execution times shown in the table are in seconds, measured on a SUN SPARC IPX workstation. We used GASE as the iterative solution method, and the SPNP package. For the cases presented, the original method required at least a few million iterations, and hence

Parameters			Single State Method		Multiple State Method			
			Case (1)		Case (2)		Case (3)	
m (States)	λ	MTTF	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$
6 (28)	0.1	1.25×10^5	1,502 (0.57)	(6,0) 0.0112	54 (0.27)	(3,3) 0.2356	42 (0.28)	(2,4) 0.4039
	0.01	9.16×10^{10}	1,281 (0.55)	(6,0) 0.0152	47 (0.29)	(3,3) 0.3184	32 (0.29)	(2,4) 0.5458
8 (45)	0.1	1.02×10^6	6,444 (2.77)	(8,0) 0.0024	84 (0.36)	(4,4) 0.1738	70 (0.34)	(3,5) 0.3110
	0.01	6.61×10^{13}	4,782 (2.15)	(8,0) 0.0038	65 (0.34)	(4,4) 0.2664	55 (0.31)	(3,5) 0.4764

Table 5: Convergence properties for computing the MTTF in the machine repairman problem with Erlang(2) failure distributions and m components of one type (multiple state method).

is not included. For the multiple state method we give only the maximum number of iterations that are required by any one of the $|A|$ systems of equations. In this table, case (1) corresponds to the single state method where $A = \{(m, 0)\}$. Here state (i, j) represents the system in which i components are non-failed and j components are in the second Erlang stage of failure. We note that the initial state does not dominate the steady-state behavior. Case (2) includes the state with the

highest steady-state probability into set A while case (3) includes the two states with the highest steady-state probabilities into A . We have indicated in the table, the corresponding state that is added to set A for the two cases. For example, when $m = 6$ and $\lambda = 0.1$, set $A = \{(6,0), (3,3)\}$ for case (2) and $A = \{(6,0), (3,3), (2,4)\}$ for case (3) respectively. We also give $\pi(A)$ the sum of the steady-state probabilities of all the states included in the set A . Table 5 does show that the iteration count decreases as $|A|$ increases. Also, case (2) shows a marked decrease in the CPU time compared to case (1), but the CPU times of case (2) and case (3) are comparable.

We next consider the closed queueing network model from the earlier subsection and evaluate the performance of the multiple state method. The results for this model are presented in Table 6. All execution times shown in the table are in seconds, measured on a SUN SPARC IPX workstation. We used GASE as the iterative solution method, and the SPNP package. For the cases presented,

Parameters			Single State Method		Multiple State Method			
			Case (1)		Case (2)		Case (3)	
N (States)	λ	MTTF	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$
20 (253)	0.5	1.74×10^2	13,232 (29.1)	(0,0) 1.75×10^{-8}	270 (1.78)	(0,15) 0.1190	270 (2.21)	(0,16) 0.2380
	0.2	1.05×10^7	4,610 (10.6)	(0,0) 0.0054	281 (1.85)	(0,7) 0.0901	276 (2.35)	(0,6) 0.1748
	0.1	1.15×10^{11}	369 (1.44)	(0,0) 0.1008	153 (1.23)	(0,2) 0.3124	153 (1.56)	(0,1) 0.5240
40 (903)	0.5	1.76×10^2	13,247 (113)	(0,0) ≈ 0	277 (6.69)	(0,35) 0.1190	277 (8.41)	(0,36) 0.2380
	0.2	1.17×10^6	15,283 (129)	(0,0) 2.5×10^{-9}	694 (15.4)	(0,27) 0.0759	583 (17.8)	(0,26) 0.1518
	0.1	3.22×10^{12}	16,007 (143)	(0,0) 0.0017	667 (13.7)	(0,12) 0.0577	621 (18.2)	(0,11) 0.1137

Table 6: Convergence properties for computing the MTTF in the closed queueing network model of a computer system.

the original method requires at least a few million iterations, and hence is not included. In this table, for the multiple state method we give only the maximum number of iterations required by any one of the $|A|$ systems of equations. In this table, case (1) corresponds to the single state method. Even for this model, the initial state does not dominate the steady-state behavior. For all the cases considered, the system starts out in the state $(0,0)$. Hence $A = \{(0,0)\}$ for case (1). For this model, state (i,j) represents the system in which i jobs are at the CPU, and j jobs are at the disks. Case (2) includes the state with the highest steady-state probability into set A , while case (3) includes the two states with the highest steady-state probabilities. We indicate the corresponding state that is added to set A for each of the cases. For example, when $N = 20$ and $\lambda = 0.5$, $A = \{(0,0), (0,15)\}$ for case (2) and $A = \{(0,0), (0,15), (0,16)\}$ for case (3) respectively. We also list $\pi(A)$ the sum of the steady-state probabilities of all the states included in the set A . For a fixed population N , the performance of the single state method degrades as λ increases. The reason for this degradation is that the initial state $(0,0)$ becomes rare (under the steady state distribution) as λ increases. The convergence improves dramatically by adding a single (probable) state to A . However, when two states are added to A , the CPU time increases somewhat.

The final example we consider is one for which performance of the single state method degrades seriously and use of the multiple state method can result in orders of magnitude improvement over the single state method. The model is an open tandem network with two multiple server queueing stations with service rate μ_i at station i , S servers at each station, and arrival rate λ . The rare event of interest is the total number of jobs in the system reaching or exceeding n . A state is denoted by (i, j) where the number of jobs at stations 1 and 2 are i and j respectively. For this model, we set the average utilization $= \rho_i = \lambda/(\mu_i S) = 0.5$ at each station, and we set $n = 50$. (The Markov chain of this model contains 1326 states.) Thus the average number of busy servers at each station is $0.5S$. For a fixed ρ_i , as S increases, the steady-state probability of state $(0,0)$ converges to zero. Thus, when the initial state is $(0,0)$, the system moves with high probability to a neighborhood around $(\rho_1 S, \rho_2 S)$ and tends to stay there for a long time. As S and N increase, keeping ρ_i fixed, we expect that the assumptions for Proposition 2 will be violated. Thus we expect performance of the single state method, starting in $(0,0)$ to degrade as S increases. The results for the multiple state method are given in Table 7. All execution times shown in the table are in seconds, measured on a SUN SPARC 1000 workstation. We used GASE as the iterative solution method, and the SPNP package. Just as in the previous examples, case (1) corresponds to the single state method

Parameters		Single State Method		Multiple State Method			
		Case (1)		Case (2)		Case (3)	
S	MTTF	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$	Iterations (CPU Time)	(i, j) $\pi(A)$
1	9.05×10^{13}	309 (3.53)	(0,0) 0.2499	293 (3.38)	(1,0) 0.3750	288 (5.43)	(0,1) 0.5000
3	2.69×10^{13}	944 (8.32)	(0,0) 0.0443	340 (6.29)	(1,1) 0.1440	299 (7.91)	(2,1) 0.2188
5	5.87×10^{12}	5,581 (42.76)	(0,0) 0.0064	413 (7.45)	(2,2) 0.0691	334 (8.74)	(3,2) 0.1213
7	1.15×10^{12}	35,533 (264.8)	(0,0) 0.0009	504 (8.79)	(3,3) 0.0464	387 (9.94)	(4,3) 0.0862
9	2.14×10^{11}	234,508 (1739)	(0,0) 0.0001	621 (10.6)	(4,4) 0.0357	463 (11.8)	(5,4) 0.0677

Table 7: Convergence properties for computing the MTTF in tandem Jackson queue example.

with $A = \{(0,0)\}$. For the multiple state method we give only the maximum number of iterations required by any one of the $|A|$ systems of equations. When $S = 5$, $A = \{(0,0), (3,3)\}$ for case (2) and $A = \{(0,0), (3,3), (4,3)\}$ for case (3) respectively. We also list $\pi(A)$ the sum of the steady-state probabilities of all the states included in the set A . As expected, for a fixed ρ_i , the performance of the single state method degrades as S increases. Notice how $\pi\{(0,0)\}$ decreases as S increases. The convergence improves dramatically by adding a single (probable) state to A . However, when two states are added to A , the CPU time increases.

6 Conclusions

We have described a method to significantly speed up the computation of mean time to failure in Markovian models. The method, which is based on Dynkin's formula, artificially makes one or

more frequently visited states into absorbing states. We then compute the mean time to failure in the modified Markov model and provide equations to recover the mean time to failure of the original Markov model. For a class of highly dependable systems, we proved that the convergence rate of the single state method actually improves as the system becomes more reliable, in contrast to the original method. We also obtained bounds on the convergence rate for computing the mean entrance time of a rare set of states in a class of queueing models. For any Markov chain, the single state method was shown to always converge faster than the original method. The new algorithm is implemented in both the SAVE and SPNP packages and shown to reduce computation time on several example problems by many orders of magnitude.

For highly reliable systems, there appears to be little need to employ the multiple state method. In fact, for some examples, use of the two state method increases CPU time compared to the single state method. However, the multiple state method does have advantages if the initial state is itself rare. We experimented with selecting states for the multiple state method according to the steady state distribution. Further work is needed in automating the decision as to which, if any, states should be used for the multiple state method.

Acknowledgment

Perwez Shahabuddin suggested using a continuity of eigenvalues argument.

References

- [1] Blake, J.T., A. L. Reibman, and K. S. Trivedi. 1988. Sensitivity Analysis of Reliability and Performability Measures for Multiprocessor System, *Proc. 1988 ACM SIGMETRICS Int. Conf. on Measurement and Modelling of Computer Systems*, Santa Fe, NM, 177–186.
- [2] Carrasco, J.A., A. Calderón and J. Escribá. 1996. Two new algorithms to compute steady-state bounds for Markov models with slow forward and fast backward transitions. In *Proceedings of the Fourth International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '96)*, 89-95, IEEE Computer Society Press.
- [3] Chen, H., and A. Mandelbaum. 1991. Discrete flow networks: bottleneck analysis and fluid approximations. *Mathematics of Operations Research* 16: 408-446.
- [4] Choi, H. and K. S. Trivedi, 1993. Conditional MTTF and its Computation in Markov Reliability Models. In *Annual Reliability and Maintainability Symposium*, Atlanta, GA, January 25-28.
- [5] Chung, K.L. 1967. *Markov Chains with Stationary Transition Probabilities*, Second Edition. New York, NY: Springer-Verlag.
- [6] Ciardo, G., A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of Markov reward models using Stochastic Reward Nets. In C. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, volume 48. Springer-Verlag, Heidelberg, Germany, 1993.
- [7] Coolidge, J. L., *A Treatise on Algebraic Plane Curves*, Dover, New York, 1959.

- [8] Cottle, R. W. 1974. Manifestation of the Schur complement. *Linear Algebra Applications* 8, 189-211.
- [9] Glasserman, P. and S.G. Kou. 1995. Analysis of an importance sampling estimator for tandem queues. *ACM Transactions on Modeling and Computer Simulation* 5, 1: 22-42.
- [10] Golub, G. H., and C. F. Van Loan. 1989, *Matrix Computations*, 2nd ed.. Baltimore, MD: Johns Hopkins University Press.
- [11] Goyal, A., W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, and K.S. Trivedi. 1986. The System Availability Estimator. In *Proceedings of the Sixteenth International Symposium on Fault-Tolerant Computing*, 84-89, IEEE Computer Society Press.
- [12] Goyal, A., and S.S. Lavenberg. 1987. Modeling and analysis of computer system availability. *IBM Journal of Research and Development* 31, 6: 651-664.
- [13] Goyal, A., P. Shahabuddin, P. Heidelberger, V.F. Nicola, and P.W. Glynn. 1992. A unified framework for simulating Markovian models of highly reliable systems. *IEEE Transactions on Computers* C-41: 36-51.
- [14] Karlin, S. and H.M. Taylor. 1975. *A First Course in Stochastic Processes*, Second Edition. New York, NY: Academic Press.
- [15] Karlin, S. and H.M. Taylor. 1981. *A Second Course in Stochastic Processes*. New York, NY: Academic Press.
- [16] Keilson, J. 1979. *Markov Chain Models - Rarity and Exponentiality*. New York, NY: Springer Verlag.
- [17] Muppala, J. K., M. Malhotra and K. S. Trivedi, Markov dependability models of complex systems: Analysis techniques, *NATO ASI Workshop, Lecture Notes in Computer Science*, S. Ozekici (ed.), Springer-Verlag, Heidelberg, 1996, to appear.
- [18] Noble, B. 1969. *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- [19] Parekh, S., and J. Walrand. 1989. A quick simulation method for excessive backlogs in networks of queues. *IEEE Transactions on Automatic Control* 34: 54-56.
- [20] Shahabuddin, P. 1994. Importance sampling for the simulation of highly reliable Markovian systems. *Management Science* 40: 333-352.
- [21] Shahabuddin, P., V.F. Nicola, P. Heidelberger, A. Goyal, and P.W. Glynn. 1988. Variance reduction in mean time to failure simulations. In *1988 Winter Simulation Conference Proceedings*, 491-499, IEEE Press.
- [22] W. Stewart and A. Goyal. Matrix methods in large dependability models. Technical Report RC-11485, IBM T.J. Watson Res. Center, Yorktown Heights, NY, 10598, Nov. 1985.
- [23] Varga, R.S. 1962. *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall, Inc.