Analysis of Stiff Markov Chains

ANDREW REIBMAN
AT&T Bell Laboratories, Crawford's Corner Road, Room 2L-518, Holmdel, NJ 07733.
ARPANET: hoqamlafr@att.com

KISHOR TRIVEDI
Department of Computer Science, Duke University, Durham, NC 27706

SANJAYA KUMAR
Hewlett-Packard, 19420 Homestead Road, Mail Stop 43L.N, Cupertino, CA 95014

GIANFRANCO CIARDO
Software Productivity Consortium, 1880 Campus Commons Drive North, Reston, VA 22091

(Received August, 1988; final revision received October, 1988; accepted November, 1988)

Continuous-time Markov chains (CTMC) are widely used mathematical models. Reliability models, queueing networks, and inventory models all require transient solutions of CTMC. The cost of CTMC transient solution increases with size, stiffness, and mission time. To eliminate stiffness and reduce the cost of solution, approximation techniques have been proposed. In this paper, we describe a software package for the specification and solution of stiff CTMC. As an interface, we use a language for the description of Markov chains. The language also provides facilities for controlling the solution procedure. Both exact and approximate solution techniques are provided. To conclude the paper, we use several examples to show the use of our specification language and the utility of our approximation technique.

Continuous-time, discrete-state Markov chains (CTMC) are mathematical models that are widely used in computer science and operations research. For example, they are the basis of models of computer system reliability, availability, and performance.

This paper describes a software package for the specification and solution of Markov chains. Our efforts have been directed toward efficient and accurate transient analysis, although facilities for steady-state analysis are also provided. For steady-state analysis, we use optimal SOR (Successive Over Relaxation). For exact transient analysis, we use the Uniformization technique. These methods are discussed in Section 1. Unfortunately, many interesting models are stiff; they have important events occurring at widely differing time-scales. Uniformization's performance degrades with stiffness. Exact “stable” techniques deal well with stiffness but incur high overhead on non-stiff problems. Instead of using a stable technique, we can eliminate stiffness and reduce the cost of solution by using an approximation technique. We developed an approximate solution technique for stiff chains in [3]. The approximation technique and its implementation are discussed in Section 2. In Section 3, we describe a language for specifying Markov chains. The language also provides features for controlling Markov chain solution. In Section 4, we show the effectiveness of our approximation technique with a series of numerical examples.

1. Exact CTMC Solution

A CTMC is a stochastic process, \( \{X(t), t \geq 0\} \), with discrete state space \( \Omega \). A CTMC's state at time \( t \) models the state of a system at time \( t \). We assume that the state space is finite and of size \( N \). We let \( q_{ij}, i \neq j \), be the transition rate from state \( i \) to state \( j \). Define \( q_i = -\sum_j q_{ij} \). The matrix \( Q = [q_{ij}] \) is the "infinitesimal generator" of the CTMC. We let \( n \) denote the number of non-zero entries in \( Q \). If \( P_t(t) \) is the probability that the CTMC is in state \( i \) at time \( t \), the row vector \( \hat{P}(t) \) is called the "state probability vector" of the CTMC. The system of Kolmogorov differential equations describes the behavior of the state probability vector as a function of \( t \):

\[
\dot{P}(t) = P(t)Q, \quad P(0) = P_0.
\]  

(1)

Often, the CTMC is analyzed in "steady-state," where the system has reached "equilibrium." In equation (1), this corresponds to the condition \( \dot{P}(t) = 0 \). So to solve for the steady-state probability vector, \( \Pi \), we need only solve the linear algebraic system

\[
\Pi Q = 0, \quad \Pi e^T = 1
\]

(2)

where \( e \) is a row vector of ones and the superscript \( T \) denotes transpose. The problem of steady-state solution has been examined in detail in the literature. For steady-state solution in our package, we use an iterative linear system solver based on optimal Successive-Over-Relaxation (SOR).
Although steady-state analysis is more widely considered, many applications require CTMC transient analysis. In contrast to steady-state solution, transient analysis requires the solution of a system of first order linear differential equations with constant coefficients (1) for a set of time-points. The exact numerical transient solution of (1) is discussed in detail in [11]. We review the pertinent results here.

The general solution for the entire system (1) is given by

\[ P(t) = P_0 e^{Qt} \]  

where \( e^{Qt} \) is the “matrix exponential” \(^{[6]}\) defined by the Taylor series

\[ e^{Qt} = \sum_{i=0}^{\infty} \frac{(Qt)^i}{i!} \]  

The direct computation of the matrix exponential Taylor series is subject to severe roundoff error and is thus unsuitable as a general solution method. Similarly, classical methods from control theory, based on the eigenanalysis, \(^{[6]}\) can be slow and numerically unstable, particularly for large problems.

A first alternative to this approach is the use of numerical methods for ordinary differential equation solution. Explicit methods require only function evaluations. The simplest of these methods is the Euler method. The most well-known is the classical Runge-Kutta method. Explicit Runge-Kutta is not suitable for the solution of stiff differential equations. \(^{[11]}\) Implicit methods require the solution of a linear algebraic system at each time step. Simple implicit methods include Backwards Euler and the Trapezoid Rule. These methods suffer little performance degradation in the face of increasing stiffness. However, they are less accurate and incur substantial performance penalties on non-stiff problems.

In our package, we use the Uniformization method for the exact numerical solution of non-stiff CTMC. Uniformization is a series technique derived from the Taylor series expansion of the matrix exponential. \(^{[8,11]}\) Uniformization first applies the transformation \( Q^* = Q/q + I \), where \( q \) is the largest magnitude of a diagonal element of \( Q \) (i.e., \( q = \max_{i} |q_{ii}| \)). The solution is then given by the infinite series

\[ P(t) = \sum_{i=0}^{\infty} e^{-q t} P(0)(Q^*)^i \frac{(qt)^i}{i!} \]  

If the series is truncated after term \( k \), the error is bounded by

\[ e \leq \sum_{i=k+1}^{\infty} e^{-q t} \frac{(qt)^i}{i!} = 1 - \sum_{i=0}^{k} e^{-q t} \frac{(qt)^i}{i!} \]  

Because the matrix \( Q^* \) is non-negative, this approach requires no subtractions and is not subject to the severe roundoff error encountered when directly evaluating the matrix exponential series.

On non-stiff problems, Uniformization provides an accurate and economical solution. \(^{[11]}\) We define \( q \) to be the “index of stiffness” for a CTMC solution. As \( q \) increases, Uniformization requires \( O(q t) \) terms of the series to achieve a given accuracy. \(^{[11]}\) Since full matrix-vector multiplications cost \( O(N^2) \) floating-point operations (FLOPS), the total solution time in the full matrix case is \( O(N^2 q t) \) FLOPS. If we use sparse matrix storage, the overall run-time is then \( O(n q t) \) FLOPS, where \( n \) is the number of non-zero elements in the matrix \( Q \). As an alternative to exact solution techniques for stiff problems, we will use approximation techniques to eliminate stiffness. This will allow the efficient use of Uniformization on both stiff and non-stiff problems.

2. An Approximate Solution Technique

Approximation techniques have been widely considered for steady-state solution. \(^{[4]}\) An algorithm for the approximate transient solution of stiff Markov chains was described in [3]. We summarize the theoretical underpinnings of this algorithm in this section. The algorithm has the following main steps:

- Choose threshold \( \alpha \).
- Identify fast states.
- Using depth-first search, identify fast-recurrent classes (FRC).
- For each FRC, solve for the steady-state probability vector.
- Reduce each FRC to a single state.
- Aggregate fast-transient states.
- Solve aggregated system exactly.
- Disaggregate fast-transient states.
- Disaggregate fast-recurrent states.

A run-time analysis of the algorithm is given at the end of this section.

We will assume that matrix \( Q \) is “separably stiff.” Given an interval \([0, t]\) for transient analysis, a matrix is separably stiff if it contains rates of size \( \alpha/1(t) \) (the so-called “slow” rates), and rates of size \( \alpha/(\alpha t) \), with \( \alpha \gg 1 \) (the “fast” rates). Note, we are using \( \alpha \) to informally denote order of magnitude, and to avoid confusion with the more formal big-“O” used in asymptotic analysis. The parameter \( \alpha \) is the “degree of stiffness” of the matrix. Usually, the length of the solution interval is chosen inversely proportional to the smallest transition rates in the system. So the “degree of stiffness” of the matrix and the “index of stiffness” of the problem often have comparable orders of magnitude. Stiffness makes many integration methods
(such as Uniformization and Runge-Kutta) inefficient.\textsuperscript{[10, 11]}

Following the approach presented in \cite{3}, we classify a transition rate $q_{ij}$ as "fast" if $q_{ij} > \alpha$. A rate that is not "fast" is "slow." Similarly, we classify states of the Markov chain into "fast" states and "slow" states. A state is fast if at least one of its outgoing transitions is fast; otherwise, the state is slow. Next, to obtain the subgraph of fast connections, we delete all the slow rates in the original Markov chain. States in this subgraph are further classified into a nearly transient subset and into nearly recurrent subsets. The nearly transient subset groups the fast states that are transient in the subgraph of the fast connections; these are the fast states that in the original chain were connected to a slow state by a sequence of one or more fast transitions. On the other hand, the nearly recurrent subsets group fast states that form ergodic subchains in the subgraph of the fast connections. In the original graph, only slow transitions connect each nearly recurrent subset to the slow states or to other nearly recurrent subsets. We proceed by first describing the graphical analysis of the CTMC to obtain the fast-recurrent and fast-transient subsets. We then discuss the numerical analysis of the aggregated CTMC.

\subsection*{2.1. Graphical Analysis}

This first step in the aggregation procedure is to identify the fast-recurrent classes, the fast-transient class, and the slow states. First, we delete all the slow transitions; this yields the sub-graph of fast transitions. The fast-recurrent classes are the sets of states that (i) are "strongly connected components" of the sub-graph of the fast transitions, and (ii) have no fast transitions exiting from the set. The strongly connected components can be identified using a "depth-first" search algorithm devised by Tarjan and given in \cite{1}. All fast states that are not in a fast-recurrent class are fast-transient states. All other states are slow states.

\subsection*{2.2. Numerical Analysis}

Once the fast-recurrent and fast-transient classes are identified, the numerical portion of the aggregation procedure can begin.

\subsubsection*{2.2.1. Fast-Reccurrent Subset}

In this subsection, we consider the computation of state probabilities when the state space $\Omega_N$ of the Markov chain is partitioned into a subset $\Omega_s$ of slow states and a single nearly recurrent subset of fast states $\Omega_r$. (Note that $\sigma + \rho = N$.) The approach described is easily generalized to the case of more than one fast-recurrent subset and to the case where recurrent subsets are present together with a transient fast subset.\textsuperscript{[3]}

The transition rate matrix is partitioned as follows:

$$Q = \begin{bmatrix} Q_{ss} & Q_{sr} \\ Q_{rs} & Q_{rr} \end{bmatrix}$$

where the matrix $Q_{sr}$ contains at least one fast rate in each row and column, while all the other matrices contain only slow rates. Because the states inside subset $\Omega_s$ are, by definition, strongly connected through fast rates, and are only weakly connected to states outside $\Omega_s$, the matrix $Q$ is nearly completely decomposable.\textsuperscript{[4]}

The fast-recurrent subset thus forms a nearly ergodic Markov chain.

The transient analysis is based on the Simon and Ando theorems discussed in \cite{4}. These theorems state that system dynamics can be separated in two phases: a short-term phase and a long-term phase. During the short-term phase, the attainment of equilibrium inside each nearly completely decomposable subset is weakly affected by the interactions among aggregates. In the long-term phase, the relative values attained in short-term equilibrium are maintained approximately. System dynamics is dominated by the interaction among the aggregates.

For the case of separably-stiff Markov chains, the fast-recurrent aggregate reaches equilibrium in time that is $\mathcal{O}(1/\alpha)$ shorter than the time to reach equilibrium in the slow states. Since our main concern is transient analysis, we can assume that in time $\mathcal{O}(t)$ the recurrent subsets have already reached the short-term equilibrium.

Approximate values of the short-term equilibrium probabilities can be calculated by considering the fast-recurrent subset in isolation. To convert the fast-recurrent subset into an ergodic Markov chain, the slow rates directed toward states outside the fast-recurrent subset must be redistributed into the states of the subset. The method used for this redistribution affects the final level of accuracy of the algorithm.\textsuperscript{[5]} At present, we simply eliminate the slow rates out of the states and adjust the diagonal of $Q_{ss}$.

Let $Q_{ss}$ denote the ergodic Markov transition rate matrix obtained from $Q_{sr}$, and let $\Pi_{ss}$ denote its steady-state probability vector. In our implementation, we use Successive-Over-Relaxation (SOR) to solve for the steady-state probabilities of the modified fast-recurrent class. If we assume that the fast-recurrent class is in equilibrium, it can be condensed into a single slow state. The exit rates from the condensed state to the other slow states are computed by weighting the original transition rates with the fast-recurrent class steady-state probabilities. By this construction, the original CTMC can be reduced to a new CTMC containing $(\sigma + 1)$
slow states, with transition rate matrix given by:

\[ Q^* = \begin{bmatrix} Q_{ee} & \frac{Q_{e\nu}}{\Pi_{\nu}Q_{\nu}} \\ \frac{Q_{\nu e}}{\Pi_{\nu}Q_{\nu}} & -\lambda \end{bmatrix} \]  

(8)

with \( \lambda = \frac{Q_{\nu e}}{\Pi_{\nu}Q_{\nu}} \). Here \( e_{\nu} \) is a length \( \rho \) row-vector of ones.

2.2.2. Fast Transient Subset

Now we consider the aggregation of the set of fast-transient states. Here the state space \( \Omega_N \) of the original Markov chain is partitioned into a subset \( \Omega_e \) (of cardinality \( \sigma \)) containing only slow states and a subset \( \Omega_{\nu} \) (of cardinality \( \nu \)) containing the nearly transient subset of fast states. (Note that \( \sigma + \nu = N \).) Consequently, the transition rate matrix can be partitioned into the following form:

\[ Q = \begin{bmatrix} Q_{ee} & Q_{e\nu} \\ Q_{\nu e} & Q_{\nu\nu} \end{bmatrix}. \]  

(9)

By hypothesis \( Q_{ee} \) and \( Q_{e\nu} \) contain only slow entries; matrix \( Q_{ee} \) contains at least one fast entry, and all the diagonal entries of matrix \( Q_{e\nu} \) have absolute values of the order of the fast rates. Since the presence of fast values in matrix \( Q_{ee} \) implies that exit from subset \( \Omega_{\nu} \) (once entered) occurs in time \( \mathcal{N}(t/\alpha) \), subset \( \Omega_{\nu} \) is called a “fast-transient” subset.

By making the approximation that the fast-transient states have already reached their steady-state value by time \( t \), and using (1) and the partitioned form of \( Q \), we obtain

\[ \frac{dP^*(t)}{dt} = P^*(t)(Q_{ee} - Q_{e\nu}Q_{\nu e}Q_{\nu\nu}) \]  

(10)

\[ P^* = -P^*(t)Q_{ee}Q_{\nu e}Q_{\nu\nu} \]  

(11)

Equation 11 is a set of algebraic equations, while Equation 10 is the set of Kolmogorov equations for a reduced Markov chain defined on the state space \( \Omega_e \). The reduced Markov chain transition rate matrix is given by:

\[ Q^* = Q_{ee} - Q_{e\nu}Q_{\nu e}Q_{\nu\nu}. \]  

(12)

Matrix \( Q^* \) is sometimes referred to as the Schur complement of \( Q_{e\nu} \) in \( Q \). In our implementation, we construct this Schur complement by applying \( \nu \) steps of a standard Gaussian-elimination algorithm to matrix \( Q \) [6].

2.2.3. Solution of the Aggregated Model

Once the fast-recurrent and fast-transient classes have been aggregated, we can solve the reduced CTMC using Uniformization. An approximate solution is obtained for all the original slow states and the slow states that represent condensed fast-recurrent classes.

2.2.4. Disaggregation

Estimates for the probability of being in a state that was eliminated are obtained by “disaggregation.” The approximate state probabilities for the states in the fast-recurrent class are denoted \( P^*_e(t) \). These probabilities are obtained by multiplying the probability of being in the fast-recurrent class \( (P^*_e(t) \) obtained above) by the equilibrium probability of being in the state of interest, given that we are in the class \( (\Pi_{\nu}) \). Thus, we have

\[ P^*_e(t) = \Pi_{\nu}P^*_e(t). \]  

(13)

The disaggregation of fast-transient classes changes all the state probabilities. To disaggregate the fast-transient class, we first use equation (11) to obtain an estimate for the fast-transient state probabilities:

\[ P^*_e(t) = -P^*_e(t)Q_{ee}Q_{\nu e}Q_{\nu\nu} \]  

(14)

Note that the approximate solution for the slow states is already a state probability vector. So, to obtain a state probability vector for the entire system, we normalize both the fast and slow state entries in the approximate state probability vector:

\[ \mathcal{P}(t) = \frac{1}{c} P^*(t), \quad c = 1.0 + P^*_e(t)e^T \]  

(15)

2.3. Execution-Time Analysis

We let \( N \) be the number of states in the original CTMC. We assume that there is only one fast-recurrent class and that it contains \( N_e \) states, while the fast-transient class contains \( N_{\nu} \) states. Using full-matrix Uniformization without aggregation, the exact solution costs \( \mathcal{O}(N^2qt) \) operations.

To find the strongly connected components of the fast subgraph requires \( \mathcal{O}(n) \) operations, where \( n \) is the number of fast transitions in the chain. One should note that this analysis is purely graphical and requires no floating-point operations. If we assume that there is one fast-recurrent class of \( N_e \) states, aggregating the fast-recurrent states requires the steady-state evaluation of an \( N_e \) state Markov chain. This evaluation requires \( \mathcal{O}(N_e^3) \) floating-point operations using an iterative solver (e.g., SOR), or \( \mathcal{O}(N_e^3) \) floating-point operations using a direct solver (e.g., Gaussian elimination). Aggregating the fast-transient states requires \( N_{\nu} \) Gaussian-elimination steps on \( Q \), an \( N \times N \) matrix, for a total of \( \mathcal{O}(N_{\nu}N^3) \) floating-point operations. Even though the aggregation steps could be correctly applied in either order, because the Gaussian elimination steps in the fast-transient aggregation step are applied to the entire matrix, it is worthwhile to aggregate the fast-recurrent states first.
Once the CTMC has been aggregated, the cost of the exact solution is reduced in two ways. First the model is reduced by about \( N_b + N_f - 1 \) states. Each fast-recurrent class is transformed to a single slow state. The largest transition rate in the model is reduced from \( q \), to \( \mathcal{M}(q/\alpha) \). The run-time of full-matrix Uniformization on the aggregated model is reduced to \( \mathcal{M}\left( (N - (N_b + N_f))q/\alpha \right) \). Thus, the total execution time for approximate solution is given by \( \mathcal{M}(N^2 + N_b^2 + N_f^2 + (N - (N_b + N_f))q/\alpha) \).

Note that in the sparse case, fill may result from the approximation procedure. Provably efficient implementation of the sparse version of the aggregation algorithm is an open problem.

3. A Markov Chain Language

High level system modeling packages often require CTMC solution to analyze their models.\(^2\) Modelers may also use a CTMC solver directly. To provide several modeling packages with a simple interface to a common CTMC solver, we have implemented MCL, a language for specifying CTMC and controlling their solution. A BNF (Backus-Naur Form) grammar for MCL can be found in Figure 1, together with the definition of the meta-symbols used. Figure 2 contains the legal statements of the language. Because MCL is a regular language, its parser is small and efficient. Little overhead is required for a program to interpret an MCL file and build the corresponding numerical data structures.

The first half of an MCL file defines a CTMC, using the statements defined in Figure 2. For each row

- \( \text{extstates} \)
- \( \text{firstindex} \)
- \( \text{nstates} \)
- \( \text{order} \)
- \( \text{matrix} \)
- \( \text{option} \)
- \( \text{method} \)
- \( \text{initstate} \)
- \( \text{totaltime} \)
- \( \text{stepsize} \)
- \( \text{precision} \)
- \( \text{alpha} \)
- \( \text{solver} \)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2}
\caption{The statements in MCL.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3}
\caption{The BNF for MCL.}
\end{figure}

index (numbered firstindex to firstindex+nstates), column indices and the transition rate values are specified in pairs. If needed, the diagonal of the infinitesimal transition rate matrix is computed internally. The second half of an MCL file specifies the solution algorithm and control parameters that guide the solution.

Several extensions to MCL are being considered. A block or subchain definition capability would simplify the specification of large models with regular structure. Symbolic variables and expressions could be provided for specifying transition rates. Values could then be bound to symbolic parameters before solution. This would allow several related models to be specified almost identically. (This approach is already used in the SHARPE package.\(^3\)) Note that both these extensions could be implemented by defining a higher level Markov chain language that could be parsed into MCL.

4. A Numerical Example

To illustrate the use of our solution package, we present a numerical example. As an example, we use a model of an M/M/1/k queue with breakdowns and repairs. The first "M" denotes a Poisson arrival process. The second "M" denotes an exponentially distributed service time. The system has 1 server and 3 buffers to hold customers waiting for or receiving service. We denote the states of the corresponding Markov chain by the pair \((i,j)\), where \(i \leq k\) is the number of customers in the system, and \(j = 0, 1\) is the number of failed servers. Let us further denote the arrival and service rates by \(\lambda\) and \(\mu\), respectively, and the server failure and repair rates by \(\gamma\) and \(\tau\). The Markov transition graph for the system is shown in Figure 3. An MCL specification for the CTMC is given in Figure 4. The states in
the upper row form a fast-recurrent class. After the fast-
recurrent class is aggregated, the first \( k \) states of the
bottom row are fast transient states that can be elimi-
nated by aggregation. The final aggregated chain has
two states.

In the baseline model, \( \mu = 2\lambda, \gamma = 10^{-4}, \) and \( \tau =
10^{-2} \). Note that the degree of stiffness is \( \alpha = \lambda/\tau \).
We solve the model for values of \( \lambda \) ranging from 0.1 to
100.0, time values \( t = 1, 5, 10, 50, 100, 500, \) and 1000,
and an accuracy requirement of \( 10^{-9} \). The runs were
executed on a DEC VAX 8600 using a code written in
the C language. In Figure 5, we plot the CPU time
required to solve the model as a function of \( \lambda \). The
amount of CPU time required by Uniformization in-
creases linearly with \( \lambda \). For the same series of models,

Figure 3. Model for M/M/1/k queue with breakdown and
repair.

Analysis of Stiff Markov Chains

Figure 4. M/M/1/k model CTMC specification.

Figure 5. CPU time needed for exact solution of M/M/1/k example as a function of stiffness.
Figure 6. M/M/1/k exact solution.

Figure 7. Accuracy of M/M/1/k approximate solution as a function of stiffness.
the approximate solution required less than 5 CPU
seconds for each \( \lambda \) value. Next, we compare the exact
and approximate solutions for \( P_{22}(t) \), the probability of
having all buffers full and the server down. In Figure 6,
we plot the exact solutions as a function of \( t \) for various
values of \( \lambda \). Note that, because the values of the param-
eters \( \gamma \) and \( \tau \) are fixed, varying \( \lambda \) also varies the stiffness
ratio \( \alpha \) as well. Because the quantities we are interested
in are small (corresponding to a system failure proba-
bility), we are primarily interested in their order of
magnitude. In Figure 7, we plot the difference between
the log of the exact solutions and the log of the corre-
sponding approximate solutions. For each curve, a
different value of \( \lambda \) was used. We observe that the
approximate solutions' accuracy generally increase
when \( \lambda \) and \( t \). In general, we have observed that the
approximation algorithm works well with \( \alpha \gg 10^4 \). In
fault-tolerant system models, \( \alpha \gg 10^9 \) is common.

5. Conclusion

We have described a software package for the specifi-
cation and solution of Markov chains. The user inter-
face is a Markov chain language that is suitable for both
human and automatic model description. The language
also provides facilities for controlling the solution of
the input chain.

For exact steady-state solution, we use an iterative
linear system solver. For exact transient solution, we
use the Uniformization technique. However, the main
focus of the package is to provide access to an approx-
imation technique for stiff chains. The approximation
technique allows us to eliminate stiffness and reduce
solution cost.

Acknowledgments

This work was supported in part by the Army Research
Office under contract DAAG29-84-0045, and by the Air
Force Office of Scientific Research under grant AFOSR-84-
0132. The work of the fourth author was supported in part
by an IBM graduate fellowship.

References

and Analysis of Computer Algorithms, Addison-Wesley,
Reading, MA.
W. Smith, 1987, Analysis of Some Fault-Tolerant Arc-
chitectures Using HARP, IEEE Transactions on Reli-
bility R-36: 2, 176–185.
 technique for the transient analysis of stiff Markov
 chains, IEEE Transactions on Computers C-35: 9,
803–814.
Computer System Applications, Academic Press, New
York.
of Complex Structures, Communications of the ACM 28,
590–603.
putations, Johns Hopkins University Press, Baltimore.
abilistic Modeling of Computer System Availability,
vian Queuing Systems, Computers and Operations Re-
search 4, 47–56.
Equations and Singular Perturbation Problems. D. Rei-
del, Dordrecht, Holland.
sient Analysis of Markov Models, Computers and Op-
erations Research 15:1, 19–36.
ing Using SHARPE, IEEE Transactions on Reli-
bility R-36:2, 186–193.
niques in Markov Modeling, Communications of the
ACM 21:2, 144–152.
Large Dependability Models, Research Report,
IBM T. J. Watson Research Center (submitted for
publication).
iability, Queueing, and Computer Science Applica-
tions, Prentice-Hall, Englewood Cliffs, NJ.