

STABILITY AND CONDITIONING ISSUES ON THE NUMERICAL
SOLUTION OF MARKOV CHAINS

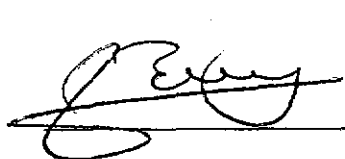
by
TUĞRUL DAYAR

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

COMPUTER SCIENCE

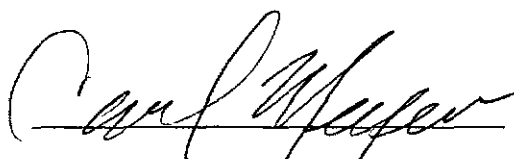
Raleigh
1994

APPROVED BY:



William I. Stewart

Chair of Advisory Committee



H. S. Pervov

ABSTRACT

DAYAR, TUĞRUL. Stability and Conditioning Issues on the Numerical Solution of Markov Chains. (Under the direction of William James Stewart.)

Markovian modeling and analysis is extensively used in many disciplines in evaluating the performance of existing systems and in analyzing and designing systems to be developed. In most cases the systems under consideration are large, and therefore require the employment of numerical solution techniques. With the changing face of computing environments, there are several issues that need to be addressed in the numerical solution of Markov chains. Firstly, stability and conditioning issues in the solution of such large systems has to be of concern, and one should be in the lookout for algorithms that are more efficient and that produce more accurate results. This may be achieved by improving existing algorithms or developing new ones that take advantage of the increased computer power. In this thesis, the effects of using a modified version of Gaussian elimination in the iterative aggregation-disaggregation technique, which is especially suited to the solution of ill-conditioned nearly completely decomposable Markov chains, has been investigated. A second solution technique geared towards multivector computers has been extended to include the systems of interest. A third solution technique, which is useful in handling vast state spaces, is shown to be effective in computing performance measures for systems having highly unbalanced stationary probabilities. Experiments on real-life applications (involving mostly large systems) have been conducted in each case. Finally, the possibility of reducing the state space size by combining states in nearly completely decomposable chains is discussed.

Acknowledgements

I would like to thank Dr. William J. Stewart for his invaluable help, time, persistent encouragement, advice, ideas and suggestions throughout this study on the numerical solution of Markov chains. He was the one to introduce me to the area, and he not only has been a role model but also has provided me with an excellent work environment. Without his continuing support and supervision this work would not have been possible. I also would like to thank Dr. Carl D. Meyer for his guidance, motivation and the insightful discussions we had. Through his presence in my committee, I have acquired a different perspective for analyzing problems. He will be a big influence in the years to come. Special thanks are due to Dr. Harry G. Perros and my minor representative Dr. Ioannis Viniotis for their contribution, helpful suggestions and criticism. I will not forget them. I also would like to extend my gratitude to the other professors I have taken courses with and I have got to know during the past five years. I especially would like to acknowledge Dr. Robert E. Funderlic, Dr. Arne Nilsson and Dr. Oscar Wesler from whom I have learned a lot; they will greatly influence my future work as well. The work on the arithmetic mean method and splittings of singular matrices has been prepared with the help of Dr. Michele Benzi. I thank him for introducing me to the subject, his interesting ideas and his friendship.

I am grateful to the graduate administration of the Department of Computer Science for the financial support provided throughout my studies at North Carolina State University and to the faculty and staff of the same department for creating a very pleasant environment for learning and research.

Lastly, I am forever grateful to my parents and my sister for their unwavering support throughout my student life.

Biography

Tuğrul Dayar was born August 28, 1968 in Ankara, TURKEY. He received his elementary and secondary education in Brussels, BELGIUM, Ankara and İzmir, TURKEY graduating from high school at İzmir Bornova Anadolu Lisesi in 1985.

He received the Bachelor of Science degree in Computer Engineering with high honors from Middle East Technical University, Ankara in 1989. In the same year he was admitted to North Carolina State University to begin studying for the Master of Science in Computer Science and received his degree with a minor in Computer Engineering in 1991. He is currently a candidate in the doctoral program in the same department. During the past five years he has been a teaching assistant for the graduate courses Introduction to Performance Modeling, Architecture of Parallel Computers, Expert Systems and undergraduate courses Numerical Analysis and Database Management Systems.

Contents

List of Tables	vi
List of Figures.	viii
Symbols.	ix
1 Introduction and Overview	1
1.1 Markov Processes	1
1.1.1 Definition	1
1.1.2 Discrete- and Continous-time Markov Chains	3
1.1.3 Linear System Formulation	5
1.1.4 Analysis	9
1.2 Basic Terminology	10
1.2.1 State Classification	10
1.2.2 NCD Markov Chains	15
1.2.3 Lumpability	17
2 Numerical Solution Methods	20
2.1 Direct Methods	21
2.2 Iterative Methods	23
2.3 Large State Space Techniques	31
3 Using GTH in IAD	33
3.1 IAD for NCD Markov Chains	33

3.2	Solving the Coupling Matrix	35
3.3	Using GTH in the Disaggregation Step	39
3.4	Implementation Considerations	43
4	Applications with GTH in IAD	50
4.1	Test Problem 1	52
4.2	Test Problem 2	53
4.3	Test Problem 3	54
4.4	Conclusion	63
5	AM Method for Markov Chains	65
5.1	Arithmetic Mean Method	66
5.2	A Reliability Model	70
5.3	Numerical Experiments	73
5.4	Conclusion	78
6	State Space Exploration	79
6.1	State Exploration in Markov Chains	80
6.2	Bounding Performance Measures	87
6.3	Numerical Experiments	90
6.4	Conclusion	101
7	Quasi-lumpability in NCD Chains	103
7.1	Definition	104
7.2	Construction	104
7.3	A Simple Example	107
8	Concluding Remarks	110
	Bibliography	113

List of Tables

4.1	Notation for parameters of numerical methods	51
4.2	Solvers used	51
4.3	Results for Problem 1	52
4.4	Results for Problem 2(a)	53
4.5	Results for Problem 2(b)	54
4.6	Sizes of aggregates for Problem 3(a)	56
4.7	Results for Problem 3(a)	56
4.8	Results for Problem 3(b)	56
4.9	Results for Problem 3(c)	57
4.10	Results for Problem 3(d)	57
4.11	Sizes of aggregates for Problem 3(e)	58
4.12	Results for Problem 3(e)	58
4.13	Sizes of aggregates for Problem 3(f)	58
4.14	Results for Problem 3(f)	59
4.15	Results for Problem 3(g)	59
4.16	Sizes of aggregates for Problem 3(h)	59
4.17	Results for Problem 3(h)	60
4.18	Sizes of aggregates for Problem 3(i)	60
4.19	Results for Problem 3(i)	60
4.20	Sizes of aggregates for Problem 3(j)	61

4.21	Results for Problem 3(j)	61
4.22	Results for Problem 3(k)	61
5.1	Results for $n = 2,500$ and $n = 10,000$ ($\lambda_1 = 1.0$, $\lambda_2 = 2.0$)	76
5.2	Results for $n = 2,500$ ($\alpha = 1.0$)	76
6.1	Sizes and number of aggregates: load 0.1	92
6.2	Sizes and number of aggregates: load 0.01	92
6.3	Sizes and number of aggregates: load 0.001	93
6.4	Sizes and number of aggregates: load 0.0001	93
6.5	Marginal probability distribution: loads 0.1, 0.01, 0.001, 0.0001 . . .	94
6.6	Number of states that accumulate a given probability mass: loads 0.1, 0.01, 0.001, 0.0001	94
6.7	Average number of cells in the buffer at steady state: loads 0.1, 0.01, 0.001, 0.0001	95

List of Figures

3.1	The generic iterative aggregation-disaggregation (IAD) algorithm . .	34
3.2	Matlab Code for GTH	42
3.3	An example using row-wise sparse storage format	45
3.4	An example using Knuth's sparse storage format	49
4.1	An interactive computer system	54
6.1	Dynamic state exploration algorithm	81
6.2	Bounding the conditional stationary distribution of a subset of states	89

Symbols

Below is a list of symbols used and their definitions:

A	boldface capital letters denote matrices
a	boldface lowercase letters denote column vectors
a, α	lowercase letters denote scalars
$\mathcal{A}$	calligraphic letters denote sets

$\mathcal{S}$	the set of states of a Markov chain
$\#(\mathcal{S})$	the cardinality of $\mathcal{S}$
$\mathcal{S}_i$	a subset of states in $\mathcal{S}$

$\mathbf{A}^{-1}$	the inverse of $\mathbf{A}$
$\mathbf{A}^T$	the transpose of $\mathbf{A}$
$\mathbf{A}^\sharp$	the group inverse of $\mathbf{A}$
$\ \cdot\ $	the norm of a matrix or a vector
$\sigma(\mathbf{A})$	the spectrum of $\mathbf{A}$
$\rho(\mathbf{A})$	the spectral radius of $\mathbf{A}$
$\gamma(\mathbf{A})$	the modulus of the subdominant eigenvalue of $\mathbf{A}$

P	the stochastic transition probability matrix of a Markov chain
$\mathbf{P}_{i,j}$	the part of P corresponding to transitions from $\mathcal{S}_i$ to $\mathcal{S}_j$
Q	the transition rate matrix of a Markov chain
K	the lumped chain corresponding to a Markov chain
C	the coupling matrix associated with a NCD Markov chain
T	the iteration matrix of an iterative method

I the identity matrix
D a diagonal matrix
0 the zero matrix

π the stationary probability distribution (row) vector of the Markov chain
 ξ the stationary probability distribution (row) vector of the coupling matrix
 ξ_i the stationary probability of being in $\mathcal{S}_i$
 e a column vector of all ones
 $\mathbf{o}$ a row or a column vector of all zeros depending on the context

Chapter 1

Introduction and Overview

1.1 Markov Processes

1.1.1 Definition

The class of stochastic processes named after A. A. Markov (1856–1922) has been of considerable interest and use to scientists from many disciplines. What is referred to as a Markov process arises in queueing network analysis, large scale economic modeling, reliability analysis, computer systems performance evaluation and other areas. However, before we formally define a Markov process and mention its uses, we need to provide the definition of a stochastic process. Throughout this section, we mainly follow [22], [30], and [51] in the given order of emphasis.

A *stochastic process* is a family of random variables $X(t)$ where the random variables are indexed by the time parameter t . The evolution of the random variables $X(t)$ is usually characterized by four quantities: the *state space* $\mathcal{S}$, the *time parameter* t , the *statistical dependencies* among the random variables $X(t)$ for different values of t , and the *time homogeneity*.

1. The *state space* $\mathcal{S}$ is the set of values $X(t)$ may assume. If the set of values is finite or countable, then the process is named as a *discrete-state process* (usually called a *chain*). Contrarily, if $\mathcal{S}$ is uncountable, then the process is

named as a *continuous-state process*.

2. As for the *time parameter*, if the instants at which state changes (transitions) occur are finite or countable, then we have a *discrete-time parameter*. Whereas, if state changes may occur anywhere in time, we have a *continuous-time parameter*.
3. *Statistical dependencies* among the random variables $X(t)$ for different values of t is described by the joint distribution function of the random variables (denoted by $X = [X(t_0), X(t_1), \dots, X(t_{n-1})]$) as

$$F_X(\mathbf{x}, \mathbf{t}) = \Pr[X(t_k) \leq x_k, \quad 0 \leq k \leq n-1]$$

for all $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})$, $\mathbf{t} = (t_0, t_1, \dots, t_{n-1})$, and $n \geq 1$.

4. A stochastic process is said to be *time homogeneous* if the system state does not depend on the value of the time parameter.

Throughout this work we concentrate on time homogeneous stochastic processes that are discrete-state, finite in the state space, discrete- or continuous-time, and Markovian.

A discrete-state stochastic process becomes a *Markov chain* if the next state depends only on the current state and not on any previous states. In other words, a Markov chain restricts the time between two consecutive state changes to be *memory-less*. All dependencies on the history of the process are removed. For a continuous-time Markov chain (CTMC), this requires that the time between two consecutive state changes be exponentially distributed. On the other hand, time between two state changes for a discrete-time Markov chain (DTMC) needs to be geometrically distributed. In the next subsection we give the formal definitions of DTMCs and CTMCs.

1.1.2 Discrete- and Continuous-time Markov Chains

Let us now formally define the statistical dependencies among the random variables $X(t)$ for different values of t in DTMCs and CTMCs.

When the time parameter is discrete, $X(t)$ is denoted as X_n , which is called a *stochastic sequence*. For a DTMC, the following Markov property holds:

$$\Pr[X_n = x_n | X_k = x_k, 0 \leq k \leq n-1] = \Pr[X_n = x_n | X_{n-1} = x_{n-1}] \quad (1.1)$$

for all states x_k , $0 \leq k \leq n$ and all time points (steps) $n \geq 1$.

A CTMC may be described by a similar equation:

$$\Pr[X(t_n) = x_n | X(t_k) = x_k, 0 \leq k \leq n-1] = \Pr[X(t_n) = x_n | X(t_{n-1}) = x_{n-1}] \quad (1.2)$$

for all states x_k , $0 \leq k \leq n$ and all time points $n \geq 1$ such that $t_0 \leq t_1 \leq \dots \leq t_n$.

For a time homogeneous DTMC, the probability in (1.1) is independent of the time parameter, and therefore, it may be written as

$$\Pr[X_n = j | X_{n-1} = i] = p_{i,j} \quad \forall i, j \in \mathcal{S}, n \geq 1. \quad (1.3)$$

Following this, the probability of being at a particular state j at step n may be expressed as

$$\begin{aligned} \Pr[X_n = j] &= \sum_{i \in \mathcal{S}} \Pr[X_{n-1} = i] \Pr[X_n = j | X_{n-1} = i] \\ &= \sum_{i \in \mathcal{S}} \Pr[X_{n-1} = i] p_{i,j}, \quad n \geq 1. \end{aligned} \quad (1.4)$$

Let $\pi_j^{(n)} = \Pr[X_n = j]$, $\forall j \in \mathcal{S}$. Note that $\sum_{j \in \mathcal{S}} \pi_j^{(n)} = 1$. Then $\boldsymbol{\pi}^{(n)}$ denotes the state probability vector at step n . At this point, we must stress that, by convention, all probability vectors are row vectors. If we rewrite (1.4) with the new notation, we get

$$\pi_j^{(n)} = \sum_{i \in \mathcal{S}} \pi_i^{(n-1)} p_{i,j}, \quad n \geq 1, \quad (1.5)$$

which, in matrix form, becomes

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(n-1)}\mathbf{P}, \quad n \geq 1. \quad (1.6)$$

Here, $\mathbf{P}$ is the *one-step transition probability matrix* whose i, j^{th} entry is given by $p_{i,j}$. We delay the discussion on the properties of equation (1.6) to the next subsection and proceed with the formulation for the CTMC case.

Observe (1.2), and consider the case in which $t_n - t_{n-1} = \Delta t > 0$ is small. If Δt is sufficiently small, the probability of having more than one transition in the time interval Δt becomes negligible, and one can define the one-step transition probability $p_{i,j}(\Delta t)$. In other words, it is possible to divide the time axis into sufficiently small intervals so that there is at most one transition in each subinterval. Under these assumptions, at the end of each Δt time interval, the system behaves like a DTMC, and we can carry on with the same formulation in (1.4) for the CTMC case. Consequently, the probability of being at a particular state j at time $t + \Delta t$ may be expressed as

$$\begin{aligned} \Pr[X(t + \Delta t) = j] &= \sum_{i \in \mathcal{S}} \Pr[X(t) = i] \Pr[X(t + \Delta t) = j | X(t) = i] \quad (1.7) \\ &= \sum_{i \in \mathcal{S}} \Pr[X(t) = i] p_{i,j}(\Delta t), \quad t \geq 0. \end{aligned}$$

Let $\pi_j(t + \Delta t) = \Pr[X(t + \Delta t) = j]$, $\forall j \in \mathcal{S}$. Note that $\sum_{j \in \mathcal{S}} \pi_j(t + \Delta t) = 1$. Then $\boldsymbol{\pi}(t + \Delta t)$ denotes the state probability vector at time $t + \Delta t$, and (1.7) reduces to

$$\pi_j(t + \Delta t) = \sum_{i \in \mathcal{S}} \pi_i(t) p_{i,j}(\Delta t), \quad t \geq 0, \quad (1.8)$$

which, in matrix form, may be written as

$$\boldsymbol{\pi}(t + \Delta t) = \boldsymbol{\pi}(t)\mathbf{P}(\Delta t), \quad t \geq 0. \quad (1.9)$$

Here, $\mathbf{P}(\Delta t)$ is the *one-step transition probability matrix for time interval Δt* whose i, j^{th} entry is given by $p_{i,j}(\Delta t)$.

By subtracting $\pi(t)$ from both sides of (1.9), we get

$$\begin{aligned}\pi(t + \Delta t) - \pi(t) &= \pi(t)P(\Delta t) - \pi(t) \\ &= \pi(t)(P(\Delta t) - \mathbf{I}),\end{aligned}\tag{1.10}$$

where $\mathbf{I}$ is the identity matrix. Dividing both sides of (1.10) by Δt and taking it to the limit, we have

$$\lim_{\Delta t \rightarrow 0} \frac{\pi(t + \Delta t) - \pi(t)}{\Delta t} = \pi(t) \lim_{\Delta t \rightarrow 0} \frac{P(\Delta t) - \mathbf{I}}{\Delta t},$$

or simply

$$\frac{d}{dt}\pi(t) = \pi(t)Q(t),\tag{1.11}$$

where

$$Q(t) = \lim_{\Delta t \rightarrow 0} \frac{P(\Delta t) - \mathbf{I}}{\Delta t}.\tag{1.12}$$

$Q(t)$ is called the *transition rate matrix* (or the *infinitesimal generator matrix* of the Markov chain). Since we consider time homogeneous Markov chains, the rate of change of state probabilities is constant with respect to time and we can drop the t in $Q(t)$ and rewrite (1.11) as

$$\frac{d}{dt}\pi(t) = \pi(t)Q, \quad t \geq 0,\tag{1.13}$$

which is quite similar to (1.6) in form. In the next subsection, we show how the long-run behavior of a Markov chain may be obtained from (1.6) (or (1.13)).

1.1.3 Linear System Formulation

The state probabilities of interest in equations (1.6) and (1.13) are the *steady state* (*limiting, equilibrium, long-run*) probabilities. For the DTMC case, the steady state probability vector, when it exists, is defined as

$$\pi = \lim_{n \rightarrow \infty} \pi^{(n)},\tag{1.14}$$

whereas for the CTMC case, when the steady state vector exists, it is defined as

$$\boldsymbol{\pi} = \lim_{t \rightarrow \infty} \boldsymbol{\pi}(t). \quad (1.15)$$

That is, whenever the steady state distribution exists, after sufficiently many transitions the process will reach a unique distribution regardless of the initial distribution, and will remain in that distribution. Hence, the steady state distribution (when it exists) is independent of the starting state distribution. There are Markov chains for which a steady state distribution does not exist; we defer the discussion on this matter to Section 1.2 where existence and uniqueness of the steady state distribution is dealt with in more detail.

Since $\lim_{n \rightarrow \infty} \boldsymbol{\pi}^{(n)} = \lim_{n \rightarrow \infty} \boldsymbol{\pi}^{(n-1)}$, using (1.14) in (1.6), we obtain

$$\boldsymbol{\pi} = \boldsymbol{\pi} \mathbf{P}, \quad \sum_{i \in \mathcal{S}} \pi_i = \|\boldsymbol{\pi}\|_1 = 1. \quad (1.16)$$

Similarly, by definition, the rate of change of the steady state vector has to be zero for a CTMC (i.e., $\lim_{t \rightarrow \infty} \frac{d}{dt} \boldsymbol{\pi}(t) = \mathbf{0}$), and by using (1.15) in (1.13), we get

$$\mathbf{0} = \boldsymbol{\pi} \mathbf{Q}, \quad \|\boldsymbol{\pi}\|_1 = 1. \quad (1.17)$$

In the above analysis we assumed that the steady state distribution of a DTMC (CTMC) exists, and proved that it satisfies (1.16) ((1.17)) whenever it exists. On the other hand, any $\boldsymbol{\pi}$ satisfying (1.16) ((1.17)) is called a *stationary probability distribution* of the Markov chain. A Markov chain starting with a stationary distribution follows this distribution at all points in time. A steady state distribution when it exists is always a stationary distribution, but the converse is not true. Subsection 1.2.1 is an overview of basic Markov chain terminology and a summary of the necessary conditions for the existence of a steady state probability distribution. Lastly in this subsection, we wish to discuss the numerical properties of the matrices $\mathbf{P}$ and $\mathbf{Q}$.

By definition, $\mathbf{P}$ has entries that are greater than or equal to 0. Hence, $\mathbf{P} \geq \mathbf{0}$. Secondly, the row sums of $\mathbf{P}$ are 1, and this property can be easily verified:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} \Rightarrow \boldsymbol{\pi}\mathbf{e} = \boldsymbol{\pi}\mathbf{P}\mathbf{e} \Rightarrow 1 = \boldsymbol{\pi}(\mathbf{P}\mathbf{e}) \Rightarrow \mathbf{P}\mathbf{e} = \mathbf{e}.$$

In what follows, $\mathbf{e}$ is a column vector of 1's whose length is determined by the context in which it is used. A matrix satisfying these two properties is referred to as a *stochastic matrix*. The order of $\mathbf{P}$ is equal to the *cardinality* of the set $\mathcal{S}$, that is the number of states in the Markov chain. (1.16) is an eigensystem and we seek the unit left-hand eigenvector of $\mathbf{P}$ corresponding to eigenvalue 1.

At this point we would like to state the definition of the *spectrum* of a matrix $\mathbf{P}$, denoted by $\sigma(\mathbf{P})$, as the set of eigenvalues of $\mathbf{P}$. Mathematically, this implies $\sigma(\mathbf{P}) = \{\lambda \mid \mathbf{P}\mathbf{v} = \lambda\mathbf{v}, \mathbf{v} \neq \mathbf{0}\}$. It is a well-known fact that the spectrum of a matrix and the spectrum of its transpose are equal. Another important definition is that of the spectral radius of a matrix. The *spectral radius* of a matrix $\mathbf{P}$, denoted by $\rho(\mathbf{P})$, is defined as $\rho(\mathbf{P}) = \max\{|\lambda| \mid \lambda \in \sigma(\mathbf{P})\}$; in other words, it is the maximum magnitude of the eigenvalues of $\mathbf{P}$.

Another way of posing the problem is to rewrite it in the form

$$\boldsymbol{\pi}(\mathbf{P} - \mathbf{I}) = \mathbf{0}, \quad \|\boldsymbol{\pi}\|_1 = 1 \quad (1.18)$$

and view it as a homogeneous system of linear equations with coefficient matrix $\mathbf{P} - \mathbf{I}$ and unknown vector $\boldsymbol{\pi}$.

On the other hand, the matrix $\mathbf{Q}$ has row sums of 0 (i.e., $\mathbf{Q}\mathbf{e} = \mathbf{0}$) because each diagonal element is the negated sum of the corresponding off-diagonal elements (see 1.12). That is,

$$q_{i,i} = - \sum_{j \neq i} q_{i,j}. \quad (1.19)$$

$\mathbf{Q}$ is a singular matrix, $-\mathbf{Q}$ is an M-matrix, and one can think of (1.17) also as a homogeneous system of linear equations with coefficient matrix $\mathbf{Q}$ and unknown

vector π . $\mathbf{Q}$ is a singular matrix since $\pi \neq \mathbf{o}$ in (1.17) implies $\det(\mathbf{Q}) = 0$. Similar to (1.16), (1.17) may be viewed as an eigensystem for which the left-hand eigenvector of $\mathbf{Q}$, this time, corresponding to eigenvalue 0, is sought. Moreover, just as in the definition of an *M-matrix*,

Definition:

- Any finite matrix $\mathbf{A}$ with nonpositive off-diagonal elements and nonnegative diagonal elements which may be written in the form

$$\mathbf{A} = r\mathbf{I} - \mathbf{G}, \quad r > 0, \mathbf{G} \geq \mathbf{0}$$

and for which $r \geq \rho(\mathbf{G})$,

$-\mathbf{Q}$ has nonpositive off-diagonal elements (since $q_{i,j} \geq 0$, $i \neq j$), (from (1.19)) non-negative diagonal elements, and $\mathbf{G} = r\mathbf{I} + \mathbf{Q} \geq \mathbf{0}$, where $r = \max_{i \in S} |q_{i,i}| > 0$. Furthermore, because $(1/r)\mathbf{G}$ is a stochastic matrix, $\rho(\mathbf{G}) \leq r$, and the last condition is satisfied as well.

Observe that it is possible to transform a given problem from a DTMC formulation to a CTMC formulation or the other way around:

DTMC $\implies$ CTMC : $\pi = \pi\mathbf{P} \Rightarrow \mathbf{o} = \pi\mathbf{Q}$, where $\mathbf{Q} = \mathbf{P} - \mathbf{I}$,

CTMC $\implies$ DTMC : $\mathbf{o} = \pi\mathbf{Q} \Rightarrow \pi = \pi\mathbf{P}$, where $\mathbf{P} = (1/\mu)\mathbf{Q} + \mathbf{I}$, $\mu = \max_{i \in S} (q_{i,i})$.

Hence, the transformation

$$\text{DTMC} \iff \text{CTMC}$$

may be used (only) for the purpose of computing the stationary distribution.

1.1.4 Analysis

We are interested in computing steady state performance measures for systems modeled as Markov processes. For queueing systems, these measures may be the average number of customers, the mean waiting time, or the blocking probability for a specific queue. In communication systems, they may be the total packet loss rate, the probability of an empty system, or any other relevant measure. In any case, the measures may be computed exactly if the steady state probability vector of the Markov chain is available. When the steady state distribution is not computable due to the size of the system, or an approximation is sought in a short time, one may use techniques which require only partial information about the chain under investigation.

In this work we aim at improving existing techniques for the solution of finite Markov chains such as iterative aggregation-disaggregation (IAD), which is geared towards nearly completely decomposable (NCD) systems (Chapter 3 and Chapter 4), and at introducing new solution methods like the method of arithmetic mean (AM), which is suitable for parallel platforms that are becoming widespread (Chapter 5). We also call attention to dynamic state exploration as an effective technique for bounding performance measures in sparse, large Markov chains with highly unbalanced stationary probabilities (Chapter 6). Additionally, we introduce the idea of quasi-lumpability in NCD Markov chains (Chapter 7). In this way, we hope to develop a better understanding of stability and conditioning issues related to the numerical solution of Markov chains. By summarizing solution methods of interest, we convey the basic knowledge required to study more complex numerical algorithms (Chapter 2). However, before we go any further, in the next section, we provide an overview of the terminology and information necessary to follow the material in subsequent chapters.

1.2 Basic Terminology

1.2.1 State Classification

Without losing generality, we concentrate on DTMCs with time homogeneity. Classifying states of Markov chains requires some definitions to be made, and we mainly follow [51] in that respect.

Definitions:

- Let

$$p_{i,j}^{(n)} = \Pr[X_{m+n} = j | X_m = i] \quad (1.20)$$

denote the probability that the process goes from state i to state j in n transitions. Then,

$$p_{i,j}^{(n)} = \sum_{k \in \mathcal{S}} p_{i,k} p_{k,j}^{(n-1)}, \quad n \geq 1, \quad (1.21)$$

and

$$p_{i,j}^{(0)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (1.22)$$

If one observes (1.21), it is clear that $p_{i,j}^{(n)}$ are the entries in the matrix $\mathbf{P}^n$, the n^{th} power of $\mathbf{P}$.

- A probability distribution π over $\mathcal{S}$ is said to be a *steady state distribution* if

$$\lim_{n \rightarrow \infty} \mathbf{P}^n = \begin{pmatrix} \pi \\ \pi \\ \vdots \\ \pi \end{pmatrix}. \quad (1.23)$$

This follows from the fact that (see (1.6))

$$\pi^{(n)} = \pi^{(0)} \mathbf{P}^n, \quad n \geq 1. \quad (1.24)$$

Taking the limit of both sides in (1.24), we have

$$\lim_{n \rightarrow \infty} \boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(0)} \lim_{n \rightarrow \infty} \mathbf{P}^n.$$

By definition, if steady state distribution exists,

$$\lim_{n \rightarrow \infty} \boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}.$$

However, this requires (1.23) to hold. In other words,

$$\lim_{n \rightarrow \infty} p_{i,j}^{(n)} = \pi_j, \quad \forall i, j \in \mathcal{S}. \quad (1.25)$$

- – State j is said to be *accessible* from state i if $\exists n \geq 0$ for which $p_{i,j}^{(n)} > 0$.
Alternately, j is accessible from i if there is a positive probability that state j can be reached starting from state i in a finite number of transitions.
- Two states i and j are said to *communicate* if state i is accessible from state j and state j is accessible from state i .
- A nonempty set $\mathcal{C} \subset \mathcal{S}$ is said to be *closed* if $\forall i \in \mathcal{C}$ and $\forall j \notin \mathcal{C}$, j is not accessible from i .
- A Markov chain is said to be *irreducible* if all states communicate with each other.
- – A state is said to be *recurrent* if the process, once in that state, returns to that state with probability 1. If the probability of returning to the state is strictly less than 1, then the state is said to be *transient*. In terms of the entries in $\mathbf{P}^n$, we have

$$\sum_{n=1}^{\infty} p_{i,i}^{(n)} = \infty \iff i \text{ is recurrent} \quad (1.26)$$

$$\sum_{n=1}^{\infty} p_{i,i}^{(n)} < \infty \iff i \text{ is transient} \quad (1.27)$$

- A recurrent state i for which $p_{i,i} = 1$ is said to be an *absorbing state*.
- If the mean time to return to a state is finite, the state is said to be *positive recurrent (recurrent nonnull)*. If the mean time to return to a state is infinite and given that the state is not transient, the state is said to be *null recurrent*.
- If all states of a Markov chain are positive recurrent, null recurrent, or transient, then we respectively have a positive recurrent, null recurrent, or transient Markov chain.
- – The *period* of state i , written $d(i)$, is the greatest common divisor of all integers $n \geq 1$ for which $p_{i,i}^{(n)} > 0$. If $p_{i,i}^{(n)} = 0, \forall n \geq 1$, define $d(i) = 0$.
- A Markov chain in which each state has period 1 is said to be *aperiodic (acyclic)*; whereas, a Markov chain in which each state has the same periodicity and this period is greater than 1, is said to be *periodic (cyclic)*.
- – A positive recurrent, aperiodic state is said to be *ergodic*.
- If a Markov chain is irreducible, positive recurrent, and aperiodic, it is said to be an ergodic Markov chain.
- A Markov chain with a finite state space is said to be *regular* if

$$\lim_{n \rightarrow \infty} \mathbf{P}^n > \mathbf{0}. \quad (1.28)$$

Hence, a steady state distribution exists for a regular Markov chain.

- A Markov chain with a finite state space is said to be *doubly stochastic* if

$$\sum_{k \in S} p_{i,k} = \sum_{k \in S} p_{k,j} = 1, \quad \forall i, j. \quad (1.29)$$

The following are some important and useful consequences of the definitions given in the previous pages. We provide them without proofs. The proofs may be found in [18], [7], [51], and [21].

- If two states communicate, then they are of the same type. That is, they are both positive recurrent, null recurrent, or transient.
- States that communicate have the same periodicity.
- If the state space $\mathcal{S}$ is finite, then at least one of the states is positive recurrent.
- If the state space $\mathcal{S}$ is finite and the Markov chain is irreducible, then every state in $\mathcal{S}$ is positive recurrent.
- No state of a Markov chain with a finite state space can be null recurrent.
- If π is the stationary distribution for a Markov chain, then $\pi_j = 0$ if j is transient or null recurrent.
- If a Markov chain does not have any positive recurrent states, then the Markov chain does not have any stationary distribution. Besides, the state space for such a chain has to be infinite.
- Suppose π and π' are two different stationary distributions for a Markov chain. Then there exists infinitely many stationary distributions for the chain. In essence, any convex combination of π and π' is also a stationary distribution.
- Let $\mathcal{C}$ be any irreducible closed set of positive recurrent states in a Markov chain. Then there exists a unique stationary distribution π for the chain that is concentrated on $\mathcal{C}$. Stationary distribution probabilities for states outside $\mathcal{C}$ are all 0.

- Let $\mathcal{S} = \mathcal{S}_t \cup \mathcal{S}_{nr} \cup \mathcal{S}_{pr}$, where $\mathcal{S}_t$ is the set of transient states, $\mathcal{S}_{nr}$ is the set of null recurrent states, and $\mathcal{S}_{pr}$ is the set of positive recurrent states. The following is a summary regarding stationary distributions of Markov chains:

1. If $\mathcal{S}_{pr} = \emptyset$, then there is no stationary distribution.
2. If $\mathcal{S}_{pr} \neq \emptyset$, then there is at least one stationary distribution.
3. If $\mathcal{S}_{pr} \neq \emptyset$, $\mathcal{S}_{pr}$ is irreducible and closed, then there is a unique stationary distribution concentrated on $\mathcal{S}_{pr}$.
4. If $\mathcal{S}_{pr} = \bigcup_i \mathcal{C}_i$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $\forall i, j$, then there is a unique stationary distribution π_i concentrated on $\mathcal{C}_i$, $\forall i$. Furthermore,

$$\pi = \sum_i \alpha_i \pi_i, \quad (1.30)$$

where $\alpha_i \geq 0$, $\forall i$ and $\sum_i \alpha_i = 1$, is also a stationary distribution.

- An irreducible, positive recurrent Markov chain has a unique stationary distribution.
- If π is a steady state distribution for a Markov chain, then π is the only stationary distribution.
- A regular Markov chain is irreducible, positive recurrent, aperiodic.
- A steady state distribution exists for an ergodic Markov chain.

Since we consider only finite Markov chains in this thesis, the existence of at least one stationary distribution is guaranteed. Moreover, we restrict our attention to irreducible finite Markov chains, for which the following theorem by Perron-Frobenius (see [24], pp.532–538) proves to be quite useful:

Theorem 1.1 *If $\mathbf{G} \in \mathbb{R}^{n \times n}$ is nonnegative (i.e., $\mathbf{G} \geq 0$) and irreducible, then*

- *The matrix $\mathbf{G}$ has a positive real eigenvalue, r , equal to $\rho(\mathbf{G})$,*
- *The eigenvalue r has algebraic multiplicity 1 (i.e., it is simple),*
- *There is a positive eigenvector associated with the eigenvalue r ; there is no other linearly independent positive eigenvector.*
- *If $\lambda_1 = r, \lambda_2, \dots, \lambda_n$ are the eigenvalues of $\mathbf{G}$, and if $|\lambda_1| = |\lambda_2| = \dots = |\lambda_k|$, and if $\omega_1, \omega_2, \dots, \omega_k$ are the k^{th} roots of unity, then $\lambda_j = \omega_j r$, $j = 1, 2, \dots, k$. Moreover, the n points of the complex plane corresponding to $\lambda_1, \lambda_2, \dots, \lambda_n$ are invariant under rotations about the origin through $2\pi/k$.*

In the language of matrices, an irreducible matrix is said to be *primitive* or *imprimitive* based on whether $k = 1$ or $k > 1$, respectively. The number k is called the *index of imprimitivity*. In the theory of finite Markov chains, primitivity corresponds to aperiodicity and imprimitivity to periodicity. A consequence of the Perron-Frobenius theorem is that, since $\rho(\mathbf{P}) = 1$ for a stochastic matrix, if $\mathbf{P}$ is periodic with periodicity k , then there are k eigenvalues all equally spaced around the unit circle.

1.2.2 NCD Markov Chains

Nearly completely decomposable (NCD) Markov chains are irreducible stochastic matrices that can be ordered so that the matrix of transition probabilities has a block structure in which the nonzero elements of the off-diagonal blocks are small compared to those of the diagonal blocks.

$$\mathbf{P}_{n \times n} = \begin{pmatrix} n_1 & n_2 & \cdots & n_N \\ \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,N} \\ \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N,1} & \mathbf{P}_{N,2} & \cdots & \mathbf{P}_{N,N} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{matrix} \quad (1.31)$$

The subblocks $\mathbf{P}_{i,i}$ are square, of order n_i , with $n = \sum_{i=1}^N n_i$. Let the stationary distribution of $\mathbf{P}$, $\boldsymbol{\pi}$ (i.e., $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$, $\|\boldsymbol{\pi}\|_1 = 1$), be partitioned conformally with $\mathbf{P}$ such that $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \dots, \boldsymbol{\pi}_N)$. Each $\boldsymbol{\pi}_i$, $i = 1, 2, \dots, N$, is a row vector having n_i elements. Let $\mathbf{P} = \text{diag}(\mathbf{P}_{1,1}, \mathbf{P}_{2,2}, \dots, \mathbf{P}_{N,N}) + \mathbf{E}$. We shall refer to $\|\mathbf{E}\|_\infty$ as the *degree of coupling* and take it to be a measure of the decomposability of the matrix (see [16]). If it were zero, then $\mathbf{P}$ would be reducible.

NCD Markov chains arise in almost all areas where Markovian modeling is utilized. Small perturbations in the transition probabilities of these chains may lead to considerable changes in the stationary probabilities; therefore, they are known to be ill-conditioned. For the transition probability matrix $\mathbf{P}$, the *stochastic complement* of $\mathbf{P}_{i,i}$ [27], is

$$\mathbf{S}_{i,i} = \mathbf{P}_{i,i} + \mathbf{P}_{i,*}(\mathbf{I} - \mathbf{P}_i)^{-1}\mathbf{P}_{*,i}, \quad (1.32)$$

where

$\mathbf{P}_{i,*} : n_i \times (n - n_i)$ matrix composed of the i^{th} row of blocks of $\mathbf{P}$ with $\mathbf{P}_{i,i}$ removed,

$\mathbf{P}_{*,i} : (n - n_i) \times n_i$ matrix composed of the i^{th} column of blocks of $\mathbf{P}$ with $\mathbf{P}_{i,i}$ removed,

$\mathbf{P}_i : (n - n_i) \times (n - n_i)$ principal submatrix of $\mathbf{P}$ with i^{th} row and i^{th} column of blocks removed.

Each stochastic complement is the stochastic transition probability matrix of a smaller irreducible Markov chain obtained by observing the original process in the corresponding block of states. The conditional stationary probability vector of the i^{th} block is $\boldsymbol{\pi}_i / \|\boldsymbol{\pi}_i\|_1$ and it may be computed by solving $(\boldsymbol{\pi}_i / \|\boldsymbol{\pi}_i\|_1)\mathbf{S}_{i,i} = \boldsymbol{\pi}_i / \|\boldsymbol{\pi}_i\|_1$ (see [27] for details). However, each stochastic complement has an embedded matrix inversion which may require excessive computation. An alternative solution technique is to approximate $\mathbf{S}_{i,i}$ by accumulating the off-diagonal mass $\mathbf{P}_{i,*}$ into the diagonal

block $\mathbf{P}_{i,:}$ on a row by row basis. There are various ways in which this can be done ([48]). Thereafter, an approximation to the conditional stationary vector of the block of states may be found by solving the corresponding linear system as before.

To determine the probability of being in a certain block of states, one needs to construct the so called *coupling matrix* which shrinks each block down to a single element, forming an $N \times N$ irreducible stochastic matrix. This is accomplished by first replacing each row of each block by the sum of elements in that block row. The sum of elements of row k of $\mathbf{P}_{i,j}$ gives the probability of leaving state k of block i and entering into block j . Therefore, the operation to be performed for each block is $\mathbf{P}_{i,j}\mathbf{e}$. Moreover, each column vector, $\mathbf{P}_{i,j}\mathbf{e}$, must be reduced to a scalar. The total probability of leaving block i to enter into block j may be determined by summing the elements of $\mathbf{P}_{i,j}\mathbf{e}$ after each element of this vector has been multiplied by the probability of being in that state (given that the system is in block i). These multiplicative constants may be obtained from the stationary vector elements; they are the components of $\boldsymbol{\pi}_i/\|\boldsymbol{\pi}_i\|_1$. Hence, the i, j^{th} element of the coupling matrix is given by $c_{i,j} = (\boldsymbol{\pi}_i/\|\boldsymbol{\pi}_i\|_1)\mathbf{P}_{i,j}\mathbf{e}$. The stationary vector of the coupling matrix gives the stationary probability of being in each one of the block of states. However, the stationary vector itself is needed to form the coupling matrix. Since it is our aim to compute the stationary vector, one can approximate the coupling matrix by using an approximate stationary vector, and improve on the approximate solution iteratively ([48]). In Chapter 3 we present iterative aggregation-disaggregation (IAD) which is an effective technique for solving finite NCD chains.

1.2.3 Lumpability

Lumpability is a property of some Markov chains which, if conditions are met, may be used to reduce a large state space to a smaller one. The idea is to find a partition

of the original state space such that, when the states in each partition are combined to form a single state, the resulting Markov chain described by the combined states has equivalent behavior to the original chain, only at a coarser level of detail. Given that the conditions for lumpability are satisfied, it is mostly useful in systems which require the computation of performance measures dependent on the coarser analysis specified by the lumped chain (see [21], p.123).

Definition: A DTMC is said to be *lumpable* with respect to a given state space partition $\mathcal{S} = \bigcup_i \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $\forall i, j$ if its transition probability matrix $\mathbf{P}$ satisfies the lumpability conditions

$$\forall \mathcal{S}_i, \mathcal{S}_j \subset \mathcal{S}, \forall s \in \mathcal{S}_i : \sum_{s' \in \mathcal{S}_j} p_{s,s'} = k_{i,j}, \quad (1.33)$$

where $k_{i,j}$ is a constant value that depends only on i and j , and $p_{s,s'}$ is the one-step transition probability of going from state s to state s' . Here, $k_{i,j}$ is the i, j^{th} entry of the lumped chain. A similar definition applies to a CTMC, where the probability matrix is substituted with the infinitesimal generator $\mathbf{Q}$.

To put it in another way, the lumpability condition requires the transition probability from each state in a given partition to another partition to be the same. For a given state, the probability of making a transition to a partition is the sum of the transition probabilities from the given state to each state in that partition. At this point, we should stress that not all Markov chains are lumpable. In fact, only a small percentage of Markov chains arising in real-life applications is expected to be lumpable. However, in Chapter 7 we show that NCD Markov chains are quasi-lumpable; that is, almost lumpable [10]. Let us conclude this section with an informative example.

Example: Let

$$\mathbf{P} = \begin{array}{c} \begin{array}{cccc} & 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array} \left(\begin{array}{cccc} 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.1 & 0.4 & 0.2 \\ 0.5 & 0.1 & 0.1 & 0.3 \\ 0.5 & 0.3 & 0.2 & 0 \end{array} \right) \end{array}.$$

We take partition $\mathcal{S} = \{1, 3\} \cup \{2, 4\}$. For this partition the lumpability condition is satisfied with $k_{1,1} = 0.6$, $k_{1,2} = 0.4$, $k_{2,1} = 0.7$, $k_{2,2} = 0.3$, where $\mathcal{S}_1 = \{1, 3\}$, $\mathcal{S}_2 = \{2, 4\}$. And the lumped chain is given by

$$\mathbf{K} = \begin{array}{c} \begin{array}{cc} \mathcal{S}_1 & \mathcal{S}_2 \end{array} \\ \begin{array}{l} \mathcal{S}_1 \\ \mathcal{S}_2 \end{array} \left(\begin{array}{cc} 0.6 & 0.4 \\ 0.7 & 0.3 \end{array} \right) \end{array}.$$

Chapter 2

Numerical Solution Methods

One of the problems for which numerical methods prove to be useful is the computation of the stationary distribution vector for a stochastic matrix. The problem amounts to finding a (nontrivial) solution to a homogeneous system of linear algebraic equations

$$\mathbf{A}\mathbf{x} = \mathbf{0} \tag{2.1}$$

where $\mathbf{A}$ is a $(n \times n)$ singular, irreducible M-matrix and $\mathbf{x}$ is the unknown $(n \times 1)$ vector to be determined; in many important applications $\mathbf{A}$ can be quite large. The goal is to compute a solution to (2.1) with positive entries; that is, a positive spanning vector for $N(\mathbf{A})$, the null space of $\mathbf{A}$. Indeed a singular, irreducible M-matrix of order n has rank $n - 1$, hence $N(\mathbf{A})$ is one-dimensional (see [2], p.156). The Perron-Frobenius theorem guarantees the existence of a positive spanning vector, which can be normalized (in the 1-norm) to obtain the stationary probability distribution vector of the Markov process generated by $\mathbf{A}$. In our problem formulation, $\mathbf{A} = -\mathbf{Q}^T$ or $\mathbf{A} = -(\mathbf{P} - \mathbf{I})^T$ depending on whether the CTMC or the DTMC is supplied. Then the unknown vector $\mathbf{x}$ clearly corresponds to $\boldsymbol{\pi}^T$.

2.1 Direct Methods

Numerical methods which compute solutions to mathematical problems in a fixed number of floating-point operations are called *direct methods*. Each direct method is essentially an algorithm over which a single pass provides the solution to a given problem. A thorough discussion on direct methods for Markov chains may be found in [49]. For our purposes, the classical *Gaussian elimination* (GE) method is quite suitable. GE consists of a *forward elimination* phase in which the coefficient matrix is reduced to upper-triangular form and a *back substitution* phase through which the solution vector is obtained. The total number of operations performed in GE is $O(n^3)$, and the space requirement is $O(n^2)$. However, there are a few problems with GE and its variations. Firstly, time and space complexity increases with the size of the problem. Secondly, the elimination of one nonzero element during the reduction phase often creates several nonzero elements in positions which previously contained zeros. This undesirable situation is referred to as *fill-in*, and it is frequently encountered, especially when the coefficient matrix in (2.1) is *sparse*, meaning the ratio of the number of nonzero elements to the total number of elements in the matrix is small. The latter problem resulted in the design and employment of *sparse storage formats* which store solely the nonzero elements in the coefficient matrix. Implementation issues for direct methods are dealt with in Section 3.4. Lastly, even if the coefficient matrix may not be overwritten, the matrix (or its copy) is altered in the forward elimination phase resulting in the buildup of rounding errors [44].

Returning to the description of GE, we must point out that GE is simply a decomposition of the coefficient matrix into a lower-triangular matrix, $\mathbf{L}$, and an upper-triangular matrix, $\mathbf{U}$, i.e.,

$$\mathbf{A} = \mathbf{LU}. \quad (2.2)$$

Clearly,

$$l_{i,j} = 0, \quad i < j \quad \text{and} \quad u_{i,j} = 0, \quad i > j. \quad (2.3)$$

If $l_{i,i} = 1, \forall i$, then we have a *Doolittle decomposition*; whereas, if $u_{i,i} = 1, \forall i$, we have a *Crout decomposition*. Since $\mathbf{A}$ is a singular matrix, either $\mathbf{L}$ or $\mathbf{U}$ has to be singular.

When one mentions Gaussian elimination, it is usually the Doolittle decomposition that is referred to. Hence, GE amounts to multiplying the coefficient matrix, $\mathbf{A}$, by a lower-triangular multiplier matrix to give $\mathbf{U}$. In other words,

$$\mathbf{L}^{-1}\mathbf{A} = \mathbf{U}, \quad (2.4)$$

where $\mathbf{L}^{-1}$ is the *multiplier matrix*. Since $\mathbf{L}$ is nonsingular ($\det(\mathbf{L}) = 1$), $\mathbf{L}^{-1}$ exists. Remember that the inverse of a lower(upper)-triangular matrix is lower(upper)-triangular as well. Therefore, the i^{th} column of $\mathbf{L}^{-1}$ is composed of the multipliers that reduce the elements of the i^{th} column below the main diagonal of $\mathbf{A}$ to zero (see [14], pp.92–104). Because $\mathbf{U}$ is singular and we assumed the existence of a stationary vector (i.e., $\mathbf{A}$ is irreducible), the last row of $\mathbf{U}$ has to be zero. Consequently, the elimination needs to be carried out only for the first $n - 1$ steps. GE may be viewed as follows:

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{x} = \mathbf{U}\mathbf{x} = \mathbf{0}, \quad \|\mathbf{x}\|_1 \quad (\text{forward elimination}),$$

$$\mathbf{U}\mathbf{x} = \mathbf{0}, \quad \|\mathbf{x}\|_1 = 1 \quad (\text{back substitution}).$$

Assuming $\mathbf{A}$ is overwritten and $\mathbf{A}^{(k)}$ represents the altered coefficient matrix at the k^{th} step of forward elimination, we have

$$a_{i,j}^{(k)} = \begin{cases} a_{i,j}^{(k-1)} & i \leq k, \forall j \\ a_{i,j}^{(k-1)} + \mu_{i,k}a_{k,j}^{(k-1)} & i > k, \forall j \end{cases} \quad (2.5)$$

where the *multipliers* are given by

$$\mu_{i,k} = -a_{i,k}^{(k-1)} / a_{k,k}^{(k-1)}. \quad (2.6)$$

When forward elimination terminates, $\mathbf{U}$ is available as

$$u_{i,j} = a_{i,j}^{(n-1)}, \quad i \leq j. \quad (2.7)$$

The elements $a_{k,k}^{(k)}$ are called *pivots*, and they should not be allowed to assume a zero value. Fortunately, the *growth factor* of $\mathbf{A}$, given by

$$g_A = \max_{i,j,k} |a_{i,j}^{(k)}| / \max_{i,j} |a_{i,j}|, \quad (2.8)$$

is 1. Additionally, the multipliers, $\mu_{i,j}$, $i > j$, are bounded by 1. However, there are circumstances under which GE breaks down due to a zero pivot. In Section 3.2 we explain an alternate way of computing pivot elements so that stability is ensured throughout the forward elimination phase.

2.2 Iterative Methods

In this section, we present standard matrix iterative methods and discuss their convergence properties for the case of singular, irreducible M-matrices. Iterative methods start with an initial approximation to the solution of a given problem and compute a new approximation at each iteration. The aim is to obtain a good approximation to the solution in a small number of iterations. Hence, the number of floating-point operations in iterative methods depends on the number of iterations taken. The iterative methods to be discussed are in the form of splittings, and for that reason it is appropriate to first review some basic concepts and results related to iterative methods for singular, homogeneous linear systems. Our discussion follows [26] and [31]; however, the irreducibility assumption is not present in such works. We adhere to it for the sake of simplicity, and because it is widely fulfilled in practice.

For a square matrix $\mathbf{A}$, we define

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \quad (2.9)$$

to be a *splitting* for $\mathbf{A}$ whenever $\mathbf{M}$ is nonsingular. The splitting is said to be *regular* if $\mathbf{M}^{-1}$ and $\mathbf{N}$ are nonnegative matrices (we write $\mathbf{M}^{-1} \geq \mathbf{0}$ and $\mathbf{N} \geq \mathbf{0}$); it is *weak regular* if $\mathbf{M}^{-1} \geq \mathbf{0}$ and $\mathbf{M}^{-1}\mathbf{N} \geq \mathbf{0}$.

The matrix

$$\mathbf{T} = \mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A} \quad (2.10)$$

is the *iteration matrix* for the iterative scheme

$$\mathbf{x}^{(k+1)} = \mathbf{T}\mathbf{x}^{(k)}, \quad (2.11)$$

where $k = 0, 1, \dots$ and $\mathbf{x}^{(0)}$ is the initial guess. Since $\mathbf{M}$ is invertible, problems (2.1) and $\mathbf{x} = \mathbf{T}\mathbf{x}$ have the same solution set.

As in the nonsingular case, the convergence of iteration (2.11) depends on the behavior of the powers $\mathbf{T}^k$ of the iteration matrix, as $k \rightarrow \infty$. Let $\rho(\mathbf{T})$ denote the spectral radius of $\mathbf{T}$. We adopt the following terminology : we say that $\mathbf{T}$ is *weakly semiconvergent* if (i) $\rho(\mathbf{T}) \leq 1$ and (ii) $(\mathbf{I} - \mathbf{T})^\sharp$, the *group inverse* of $\mathbf{I} - \mathbf{T}$, exists (for a definition of the group inverse, see [3]). Regarding condition (ii), observe that the singularity of $\mathbf{A}$ implies $\rho(\mathbf{T}) = 1$, and it may happen that $1 \in \sigma(\mathbf{T})$, the spectrum of $\mathbf{T}$. By the Perron-Frobenius theorem, this will certainly happen if $\mathbf{T}$ is nonnegative, as it is the case if $\mathbf{T} = \mathbf{M}^{-1}\mathbf{N}$ where $\mathbf{A} = \mathbf{M} - \mathbf{N}$ is a (weak) regular splitting. When $1 \in \sigma(\mathbf{T})$, condition (ii) is equivalent to requiring that $\mathbf{T}$ has only linear elementary divisors corresponding to the eigenvalue 1. In other terms, the Jordan blocks associated with the eigenvalue 1 are 1×1 .

We say that (2.9) is a *weakly semiconvergent splitting* if $\mathbf{T} = \mathbf{M}^{-1}\mathbf{N}$ is weakly semiconvergent. Now let

$$\gamma(\mathbf{T}) = \max\{|\lambda| \mid \lambda \in \sigma(\mathbf{T}), \lambda \neq 1\}. \quad (2.12)$$

It may well happen that $\mathbf{T}$ is weakly semiconvergent and yet $\gamma(\mathbf{T}) = 1$. If this is not the case, we say that $\mathbf{T}$ is *semiconvergent*, and (2.9) is a *semiconvergent splitting* if

$\mathbf{T} = \mathbf{M}^{-1}\mathbf{N}$ is semiconvergent. In other terms, $\mathbf{T}$ is semiconvergent if it is weakly semiconvergent and does not have eigenvalues other than (possibly) 1 on the unit circle.

If $\mathbf{T}$ is semiconvergent, it is similar to a matrix of the form

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{J} \end{pmatrix}, \quad (2.13)$$

where $\rho(\mathbf{J}) < 1$. It is easy to see that if $\mathbf{T}$ is semiconvergent, then

$$\lim_{k \rightarrow \infty} \mathbf{T}^k = \mathbf{I} - \mathbf{F}, \quad (2.14)$$

where $\mathbf{F} = (\mathbf{I} - \mathbf{T})(\mathbf{I} - \mathbf{T})^\sharp$. Notice that $\mathbf{I} - \mathbf{F}$ is the projector onto $N(\mathbf{I} - \mathbf{T})$ along $R(\mathbf{I} - \mathbf{T})$ (see [3]).

Consider now the solution of problem (2.1), and let $\mathbf{A} = \mathbf{M} - \mathbf{N}$ be a *semiconvergent* splitting. Then the sequence $\{\mathbf{x}^{(k)}\}$ defined by (2.11) is convergent for each initial vector $\mathbf{x}^{(0)}$, with

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = (\mathbf{I} - \mathbf{F})\mathbf{x}^{(0)}. \quad (2.15)$$

The limit (2.15) will depend in general on $\mathbf{x}^{(0)}$, unless $\mathbf{x}^{(0)} \in R(\mathbf{I} - \mathbf{T}) \equiv R(\mathbf{M}^{-1}\mathbf{A})$, in which case $(\mathbf{I} - \mathbf{F})\mathbf{x}^{(0)} = \mathbf{0}$. The solution approximated by (2.11) is nontrivial if and only if $\mathbf{x}^{(0)} \notin R(\mathbf{M}^{-1}\mathbf{A})$; in this case $\mathbf{x}^* = \lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = (\mathbf{I} - \mathbf{F})\mathbf{x}^{(0)}$ is nonzero, and $\mathbf{A}\mathbf{x}^* = \mathbf{0}$ as desired, since $\mathbf{I} - \mathbf{F}$ is a projector onto $N(\mathbf{I} - \mathbf{T}) = N(\mathbf{M}^{-1}\mathbf{A}) = N(\mathbf{A})$. The *asymptotic rate of convergence* is given by

$$R_\infty(\mathbf{T}) = -\ln \gamma(\mathbf{T}). \quad (2.16)$$

The next step is to state conditions under which a splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ of a singular, irreducible M-matrix $\mathbf{A}$ is a semiconvergent splitting. Unfortunately, there are examples of regular splittings of singular irreducible M-matrices which are not semiconvergent (see [31]). This will be true, *a fortiori*, of weak regular splittings.

However, it follows from the work in [31] that a weak regular splitting of a singular, irreducible M-matrix is weakly semiconvergent. The iteration matrix $\mathbf{T} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$ is nonnegative, $\rho(\mathbf{T}) = 1$ since $0 \in \sigma(\mathbf{A})$, and it is possible that $\gamma(\mathbf{T}) = 1$. In this case, the iteration (2.11) is *not* convergent to a solution of (2.1) for each starting vector $\mathbf{x}^{(0)}$, generally speaking.

In the theory of finite Markov chains, this situation arises when the chain is irreducible but not regular, i.e., cyclic; in the language of matrices, the transition probability matrix $\mathbf{P}$ is irreducible but imprimitive. To remedy this situation, one replaces $\mathbf{P}$ by

$$\mathbf{P}_\alpha = (1 - \alpha)\mathbf{I} + \alpha\mathbf{P}, \quad (2.17)$$

where $0 < \alpha < 1$. This matrix is now regular (i.e., primitive) and its powers converge to a limiting matrix, each row of which is equal to the stationary probability distribution vector of the original chain. The standard terminology is that $\mathbf{P}$ is *Euler summable* to this limiting matrix (see [21], p.99).

This same transformation can be applied, more generally, to the iteration matrix $\mathbf{T}$ arising from a weak regular splitting of an irreducible, singular M-matrix $\mathbf{A}$. Thus, we can consider the transformed matrix

$$\mathbf{T}_\alpha = (1 - \alpha)\mathbf{I} + \alpha\mathbf{T}, \quad (2.18)$$

which is now a semiconvergent matrix for all $\alpha \in (0, 1)$. Indeed, the eigenvalues of $\mathbf{T}_\alpha$ are of the form $1 - \alpha + \alpha\lambda$ where λ is the generic eigenvalue of $\mathbf{T}$; and since $\mathbf{T}$ is weakly semiconvergent, and $0 < \alpha < 1$, the matrix $\mathbf{T}_\alpha$ has no eigenvalues other than 1 on the unit circle. This approach was considered in [31].

Notice that $\mathbf{T}_\alpha = (1 - \alpha)\mathbf{I} + \alpha(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) = \mathbf{I} - \alpha\mathbf{M}^{-1}\mathbf{A}$. Therefore $\mathbf{T}_\alpha$ can be regarded as the iteration matrix associated with the splitting $\alpha\mathbf{A} = \mathbf{M} - (\mathbf{M} - \alpha\mathbf{A})$ for the solution of the system $\alpha\mathbf{A}\mathbf{x} = \mathbf{o}$, equivalent to (2.1). The iteration is now

$$\mathbf{x}^{(k+1)} = \mathbf{T}_\alpha \mathbf{x}^{(k)} \quad (2.19)$$

($k = 0, 1, \dots$) and since $\mathbf{T}_\alpha$ is semiconvergent, the sequence $\{\mathbf{x}^{(k)}\}$ is convergent for each starting vector $\mathbf{x}^{(0)}$, the limit being given, again, by (2.15).

We may now summarize our discussion of splitting methods for singular linear systems in the following theorem.

Theorem 2. 1 *Let $\mathbf{A}$ be a singular, irreducible M -matrix and let $\mathbf{A} = \mathbf{M} - \mathbf{N}$ be a weak regular splitting. Then*

- (i) *The matrix $\mathbf{T} = \mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$ is weakly semiconvergent;*
- (ii) *For all $\alpha \in (0, 1)$ the matrix $\mathbf{T}_\alpha = (1 - \alpha)\mathbf{I} + \alpha\mathbf{T} = \mathbf{I} - \alpha\mathbf{M}^{-1}\mathbf{N}$ is semiconvergent;*
- (iii) *The iterative scheme (2.19) is convergent to a solution $\mathbf{x}^*$ of (2.1) for arbitrary initial vectors $\mathbf{x}^{(0)}$, and $\mathbf{x}^* = P\mathbf{x}^{(0)}$, where P is the projector onto $N(\mathbf{A})$ along $R(\mathbf{M}^{-1}\mathbf{A})$. In particular, $\mathbf{x}^* \neq \mathbf{0}$ if and only if $\mathbf{x}^{(0)} \notin R(\mathbf{M}^{-1}\mathbf{A})$.*

The asymptotic rate of convergence of the iterative scheme is

$$R_\infty(\mathbf{T}_\alpha) = -\ln \gamma(\mathbf{T}_\alpha), \quad (2.20)$$

where $\gamma(\mathbf{T}_\alpha)$ is the maximum magnitude over all elements in the spectrum of $\mathbf{T}_\alpha$ different from 1.

We are now in a position to present the method of Jacobi, method of Gauss-Seidel, and the SOR method. In the following, $\mathbf{D}, \mathbf{L}, \mathbf{U}$ ¹ represent respectively diagonal (i.e., $d_{i,j} = 0, i \neq j$), strictly lower-triangular (i.e., $l_{i,j} = 0, j > i$), strictly upper-triangular (i.e., $u_{i,j} = 0, i > j$) matrices. Note that, $d_{i,i} \neq 0, \forall i$ in our case. This fact is used to show the existence of $\mathbf{M}^{-1}$ in (2.10) for our choice of splittings.

¹ $\mathbf{L}$ and $\mathbf{U}$ should not be confused with the lower and upper-triangular matrices in the LU decomposition.

Jacobi:

The splitting for the method of *Jacobi* is

$$\mathbf{A} = \mathbf{D} - (\mathbf{L} + \mathbf{U}). \quad (2.21)$$

$\mathbf{D}^{-1}$ exists. Since $\mathbf{A}$ is an M-matrix, $\mathbf{D}^{-1} \geq \mathbf{0}$ and $(\mathbf{L} + \mathbf{U}) \geq \mathbf{0}$. Hence, we have a regular splitting. The iteration matrix for the Jacobi method is given by

$$\mathbf{T}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}). \quad (2.22)$$

Note that $\boldsymbol{\pi}^T$ is the right eigenvector corresponding to the unit eigenvalue of $\mathbf{T}_J$ and it may be shown that the method of Jacobi is identical to the power method applied to $\mathbf{T}_J$ (see [49]). Furthermore,

$$\|\mathbf{T}_J\|_\infty = \|\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\|_\infty \leq \|\mathbf{D}^{-1}\|_\infty \|\mathbf{L} + \mathbf{U}\|_\infty.$$

Since $d_{i,i} = \sum_{j \neq i} (l_{i,j} + u_{i,j})$, $\|\mathbf{T}_J\|_\infty \leq 1$ implying $\rho(\mathbf{T}_J) \leq 1$. The Jacobi iteration will converge to the stationary vector if $\mathbf{T}_J$ is primitive.

Gauss-Seidel:

The splitting for the method of *Gauss-Seidel* is

$$\mathbf{A} = (\mathbf{D} - \mathbf{L}) - \mathbf{U}. \quad (2.23)$$

$(\mathbf{D} - \mathbf{L})^{-1}$ exists. Since $\mathbf{A}$ is an M-matrix, $(\mathbf{D} - \mathbf{L})^{-1} \geq \mathbf{0}$ and $\mathbf{U} \geq \mathbf{0}$. In order to prove the former, one can use $(\mathbf{D} - \mathbf{L})(\mathbf{D} - \mathbf{L})^{-1} = \mathbf{I}$. We have a regular splitting in this case as well. The iteration matrix is given by

$$\mathbf{T}_{GS} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}. \quad (2.24)$$

Again $\boldsymbol{\pi}^T$ is the right eigenvector corresponding to the unit eigenvalue of $\mathbf{T}_{GS}$ and the method of Gauss-Seidel is equivalent to the power method applied to $\mathbf{T}_{GS}$. As it

is indicated in [49], $\rho(\mathbf{T}_{GS}) \leq 1$. Usually, the closer $(\mathbf{D} - \mathbf{L})$ is to $\mathbf{A}$, the faster the convergence will be. However, there are counterexamples for which this rule of thumb does not hold (see [19], p.540 for instance) since comparison theorems for splittings of nonsingular M-matrices do not extend to the singular case. Nevertheless, $\mathbf{M}$ generally should be chosen as close to $\mathbf{A}$ (see (2.9)) as possible subject to the constraint that $\mathbf{M}^{-1}$ is easy to find.

Similar to the iteration matrix $\mathbf{T}_{GS}$, which is for *forward Gauss-Seidel*, one can define a *backward Gauss-Seidel* iteration with iteration matrix,

$$\mathbf{T}_{BGS} = (\mathbf{D} - \mathbf{U})^{-1}\mathbf{L}. \quad (2.25)$$

Generally speaking, this should be preferred when most of the elemental mass of $\mathbf{A}$ is concentrated in $\mathbf{U}$; that is, above the main diagonal of $\mathbf{A}$. In any case, the Gauss-Seidel iteration will converge to the stationary vector if the iteration matrix is primitive.

SOR:

SOR stands for *successive overrelaxation*. The splitting for the SOR method is

$$\mathbf{A} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{L}\right) - \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{U}\right), \quad (2.26)$$

where ω is the *relaxation parameter*. For $\omega > 1$, the method is said to be *overrelaxation*; for $\omega < 1$, it is said to be *underrelaxation*. For $\omega = 1$, the method reduces to Gauss-Seidel. Clearly, $(\frac{1}{\omega}\mathbf{D} - \mathbf{L})^{-1}$ exists. Since $\mathbf{A}$ is an M-matrix, $(\frac{1}{\omega}\mathbf{D} - \mathbf{L})^{-1} \geq \mathbf{0}$ regardless of the value for ω . However, only for $0 < \omega < 1$ is $(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{U}) \geq \mathbf{0}$ true and (2.26) becomes a regular splitting. The iteration matrix for forward SOR is given by

$$\mathbf{T}_{SOR} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{L}\right)^{-1}\left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{U}\right). \quad (2.27)$$

A *backward SOR* iteration may also be written.

π^T is the right eigenvector corresponding to the unit eigenvalue of $\mathbf{T}_{SOR}$, but this time it is *not* necessarily true that the unit eigenvalue is the dominant eigenvalue. The magnitude of the dominant eigenvalue depends on the choice of the relaxation parameter, ω [49]. When ω is within a certain range, the unit eigenvalue is the dominant eigenvalue (i.e., $\rho(\mathbf{T}_{SOR}) \leq 1$) and the SOR method is identical to the power method applied to $\mathbf{T}_{SOR}$. The SOR method converges only if $0 < \omega < 2$ (see [2]), and it will converge to the stationary vector if $\mathbf{T}_{SOR}$ is primitive in the range of convergence for ω .

The eigenvalues of matrices $\mathbf{T}_J, \mathbf{T}_{GS}, \mathbf{T}_{SOR}$, with the exception of the unit eigenvalue, will generally be different. The iteration matrix that yields convergence in the fewest iterations should be employed in (2.11). In other words, the iteration matrix whose maximum subdominant eigenvalue in magnitude is furthest from unity should be chosen. However, no iterative scheme can be *a priori* proved to be faster than others. Hence, $\gamma(\mathbf{T})$ must be evaluated in a particular problem for each iteration matrix. Obviously, this evaluation must be cheaper than solving the problem itself.

There is a matrix-vector multiply at each iteration in (2.11); therefore, the cost per iteration is $O(n^2)$. However, this is an overestimate, especially for sparse matrices. One problem with standard matrix iterative methods is that their rate of convergence for nearly completely decomposable (NCD) Markov chains is slow. Accordingly, different techniques, one of which is iterative aggregation-disaggregation, have been developed. We discuss this technique in Section 3.1. We now turn to state space exploration techniques which may be employed to cope with vast state spaces.

2.3 Large State Space Techniques

In most real-life applications, the Markov chains we come across are sparse and they more or less possess some structure; that is, the magnitude and location of the non-zero elements is not random. Depending on the size of the system under investigation, the chain to be solved can be quite large implying an implementation problem on computers even for iterative methods. Indeed, sparse storage schemes may not be adequate to handle some chains (see [52]).

One way of getting around such a problem is to try to locate any inherent structure in the chain for which there are existing memory efficient solution techniques. For instance, if the Markov chain is NCD, then the conditional stationary probability vector of each NCD block and consequently the stationary probability vector of the entire chain may be bounded with the bounded aggregation method discussed in [6] and [40]. On the other hand, if there are lumpable (or quasi-lumpable, [10]) sets of states in the Markov chain, one can cut down the amount of computation drastically by lumping such states and solving for a smaller Markov chain (see [21]). Note that, in both of the above methods, one essentially solves for smaller Markov chains, and then weighs, or as in the second case, extends these individual solutions to the complete Markov chain. In order to benefit from these approaches, one first has to make sure that the Markov chain satisfies either one of the characteristics. If the given problem does not conform to these structural properties, bounded aggregation and lumping will not be of much help.

Another way of undertaking the problem is to explore only the states that will be used in the computation of the desired performance measures. Techniques of this type have been addressed in [8], [25], [41], and [42]. The property of sparse Markov chains that makes such a notion feasible is that each state has a few neighboring states (states within one-step transition). Secondly, in many cases, real systems have most

of the probability mass concentrated on a subset of states with the result that the system very rarely reaches the other states. The idea then is to start at a particular state and explore states in the desired direction. The search direction may be the most probable states, the least probable states, or another set of states identified in advance or for which a different criteria has to be satisfied at each iteration of the exploration procedure. However, we must stress that, so far bounding techniques have been successfully applied only to performance measures that depend upon the most probable states or upon the conditional stationary vector of a NCD block.

Under the light of the above discussion, our aim is to come up with a computationally efficient algorithm that locates states of interest rapidly. This is achieved by considering more than one starting state simultaneously and exploring fewer states from each one of these starting states. In other words, we wish to get a feeling for the overall behavior of the system by conducting a few local searches in a parallel manner. Depending on the stopping criterion chosen, the number of states explored from each starting state may vary. After states of interest are located, bounds on performance measures may be obtained either exactly through Courtois and Semal's technique discussed in [6] and [40] (see Section 6.2), or approximately through the help of the ideas presented in [41], [42], and [25]. In Chapter 6, we describe how dynamic state exploration may be effectively used in sparse, large Markov chains having highly unbalanced stationary probabilities.

Chapter 3

Using GTH in IAD

3.1 IAD for NCD Markov Chains

Nearly completely decomposable (NCD) Markov chains that appear in applications are quite large and sparse, possibly having more than thousands of states. For such large chains, direct methods can generate immense fill-in during the triangularization phase of the solution process, and due to storage limitations they become impractical. Moreover, when the system is nearly completely decomposable, there are eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods ([35]). Iterative aggregation-disaggregation (IAD) does not suffer from these limitations ([23, 50]).

The idea in IAD methods is to observe the system in isolation in each of the diagonal blocks as if the system is completely decomposable (see [43]), and to compute the stationary probability distribution of each diagonal block. However, there are two problems with this approach. First of all, since the diagonal blocks are substochastic, the off-diagonal probability mass must somehow be incorporated into the diagonal blocks. Secondly, the probabilities obtained by this approach are conditional probabilities, and this condition has to be removed by weighing each probability subvector by the probability of being in that group of states (see Subsection 1.2.2). Only if these

two problems are overcome, can one form the stationary distribution of the Markov chain by weighing the subvectors and pasting them together. The IAD algorithm is a two-step algorithm that handles these two problems in an iterative framework. Without further ado, we provide the IAD algorithm. Convergence analysis of the algorithm appears in [4]. Additional discussion may be found in [48].

Figure 3.1: The generic iterative aggregation-disaggregation (IAD) algorithm

1. Let $\boldsymbol{\pi}^{(0)} = (\boldsymbol{\pi}_1^{(0)}, \boldsymbol{\pi}_2^{(0)}, \dots, \boldsymbol{\pi}_N^{(0)})$ be a given initial approximation to the solution.
Set $k = 1$.
2. Construct the coupling matrix $\mathbf{C}^{(k-1)}$

$$c_{i,j}^{(k-1)} = \frac{\pi_i^{(k-1)}}{\|\boldsymbol{\pi}_i^{(k-1)}\|_1} \mathbf{P}_{i,j} \mathbf{e}.$$

3. Solve the eigenvector problem

$$\boldsymbol{\xi}^{(k-1)} \mathbf{C}^{(k-1)} = \boldsymbol{\xi}^{(k-1)}, \quad \|\boldsymbol{\xi}^{(k-1)}\|_1 = 1$$

for $\boldsymbol{\xi}^{(k-1)} = (\xi_1^{(k-1)}, \xi_2^{(k-1)}, \dots, \xi_N^{(k-1)})$.

4. (a) Compute the row vector

$$\mathbf{z}^{(k)} = (\xi_1^{(k-1)} \frac{\pi_1^{(k-1)}}{\|\boldsymbol{\pi}_1^{(k-1)}\|_1}, \xi_2^{(k-1)} \frac{\pi_2^{(k-1)}}{\|\boldsymbol{\pi}_2^{(k-1)}\|_1}, \dots, \xi_N^{(k-1)} \frac{\pi_N^{(k-1)}}{\|\boldsymbol{\pi}_N^{(k-1)}\|_1}).$$

- (b) Solve the N systems of equations

$$\boldsymbol{\pi}_i^{(k)} = \boldsymbol{\pi}_i^{(k)} \mathbf{P}_{i,i} + \sum_{j>i} \mathbf{z}_j^{(k)} \mathbf{P}_{j,i} + \sum_{j<i} \boldsymbol{\pi}_j^{(k)} \mathbf{P}_{j,i}$$

for $\boldsymbol{\pi}_i^{(k)}$, $i = 1, 2, \dots, N$.

5. Test $\boldsymbol{\pi}^{(k)}$ for convergence. If the desired accuracy is attained, then stop and take $\boldsymbol{\pi}^{(k)}$ as the stationary probability vector of $\mathbf{P}$. Else set $k = k + 1$ and go to step 2.

In the IAD algorithm, (2) and (3) form the aggregation step and (4)(b), which is essentially a block Gauss-Seidel iteration, forms the disaggregation step. In step (2), $\pi_i^{(k-1)} / \|\pi_i^{(k-1)}\|_1$ is an approximation to the stationary distribution of the stochastic complement of $\mathbf{P}_{i,i}$. The weighing factors, $(\|\pi_1\|_1, \|\pi_2\|_1, \dots, \|\pi_N\|_1)$, are approximated by $\xi^{(k-1)}$ in step (3). Each iteration of the IAD algorithm reduces the residual error (i.e., $\|\pi(\mathbf{I} - \mathbf{P})\|$) by a factor of $\|\mathbf{E}\|$ (see [39], p.80).

In Section 3.2, we discuss how a modified version of Gaussian elimination may be used to enforce stability in the solution of the coupling matrix. In Section 3.3, we extend the idea so that it can be used in a nonsingular system of equations with a substochastic coefficient matrix. In Section 3.4, we discuss certain implementation issues.

3.2 Solving the Coupling Matrix

The coupling matrix is an irreducible stochastic matrix of order N ; that is, each of its two dimensions is equal to the number of blocks in the NCD chain, and all states form a single communicating class. The goal is to solve the singular system $\xi^{(k)} \mathbf{C}^{(k)} = \xi^{(k)}$ subject to $\|\xi^{(k)}\|_1 = 1$, which is termed the normalization equation. Essentially, each row of $\mathbf{C}^{(k)}$ is a linear combination of the others, and inclusion of the normalization equation enables us to replace the redundant equation, thus achieving full rank in the set of equations. Being an irreducible stochastic matrix, the coupling matrix has a unique unit eigenvalue. All other $N - 1$ eigenvalues are close to 1. The distance of these other eigenvalues to 1 depends on the degree of coupling.

A careful inspection reveals that one can solve the equivalent transposed system

$$(\mathbf{I} - \mathbf{C}^{(k)})^T (\xi^{(k)})^T = \mathbf{o}, \quad \|\xi^{(k)}\|_1 = 1. \quad (3.1)$$

The justification for considering this form of the problem is the following. (3.1) is the conventional form in which a (homogeneous) linear system of equations is expressed :

the coefficient matrix postmultiplied by the unknown vector equals the (zero) right-hand side. Besides, there are already existing algorithms that may be used with this form of a linear system. But most importantly, as it is explained later in Section 3.4, a row-wise sparse storage implementation will spend an additional amount of time in the substitution phase of the non-transposed system of equations.

$(\mathbf{I} - \mathbf{C}^{(k)})^T$ is a singular M-matrix (see [2]) with 0 column sums, and we seek the unique null vector of unit 1-norm. For such a matrix, Gaussian elimination (GE) preserves column diagonal dominance throughout its computation so that the multiplier element at each step is bounded by 1 thereby eliminating the need for pivoting. This follows from the fact that the pivot element at a given step has the largest magnitude among all elements that lie in the unreduced part of its respective column. Note that the same argument would be valid for the original system of equations if $\mathbf{C}^{(k)}$ were doubly-stochastic.

It is well-known that since $\mathbf{C}^{(k)}$ is a perturbation of the identity matrix, all its nonunit eigenvalues are close to 1. Due to this fact, iterative methods tend to converge slowly. On the other hand, certain stability issues need to be addressed if direct methods are used. As we next show, ordinary GE is not stable in the presence of rounding errors on a coupling matrix whose diagonal elements are close to 1.

Example: Consider the following irreducible stochastic matrix,

$$\mathbf{P} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} - \alpha & \alpha \\ \frac{3}{4} - \alpha & \frac{1}{4} & \alpha \\ \alpha & \alpha & 1 - 2\alpha \end{pmatrix}.$$

Assuming α is small, $\mathbf{P}$ is clearly NCD. Additionally, the degree of coupling is 2α . Now, suppose 2α is less than *machine epsilon* (smallest representable positive floating-point number ϵ such that $1 + \epsilon > 1$). Let $\pi^{(k)} > \alpha$. Then, the coupling matrix is rounded to

$$\mathbf{C}^{(k)} = \begin{pmatrix} 1 & \alpha \\ 2\alpha & 1 \end{pmatrix}.$$

Note that the coupling matrix is not stochastic. Therefore, the matrix $(\mathbf{I} - \mathbf{C}^{(k)})^T$ is rounded to

$$(\mathbf{I} - \mathbf{C}^{(k)})^T = \begin{pmatrix} 0 & -2\alpha \\ -\alpha & 0 \end{pmatrix}.$$

Hence, it is necessary to apply some sort of pivoting strategy in order to proceed with GE. Otherwise, the algorithm breaks down contrary to the general belief that GE applied to stochastic matrices is always stable. Before going any further, let us elaborate on this issue some more.

It is possible for an irreducible NCD Markov chain ordered as described in (1.31) to have zero blocks. Consequently, the coupling matrix of such a chain at any iteration has zeros in locations corresponding to zero blocks in the transition probability matrix. However, the zeros in the coupling matrix occur in just the right places so that the coupling matrix is still irreducible ([27]). Keeping this in mind, the situation that causes GE to break down occurs when the pivot element at a given step of GE is zero in the matrix $(\mathbf{I} - \mathbf{C}^{(k)})^T$ during the k^{th} iteration of the IAD algorithm. A sufficient condition for the failure of GE during the aggregation step is to have a transition probability matrix with a degree of coupling less than machine epsilon. In this case, all diagonal elements in $(\mathbf{I} - \mathbf{C}^{(k)})^T$ will be zero, as in the example, and GE will fail in the first step.

Our approach to finding a remedy for this situation is the *Grassmann-Taksar-Heyman* (GTH) algorithm described in [15] and the direct method discussed in [45]. The original GTH algorithm emerges from probabilistic arguments, and it is shown that the stationary distribution of a Markov chain can be calculated using only nonnegative numbers and avoiding subtraction operations. This algorithm achieves significantly greater accuracy than other algorithms described in the literature since

there is no loss of significant digits due to cancellation ([17]). Interestingly, the inspiration for the algorithm presented in [45], which is specifically for the solution of NCD Markov chains, is the GTH algorithm. In a recent paper ([32]), it is shown that the stationary probability vector of an $N \times N$ irreducible Markov chain stored in floating-point form is close to its exact stationary vector. In other words, even if we had an algorithm that does not introduce any errors by itself, the best we can do is to compute the stationary vector of an $N \times N$ Markov chain with an entrywise relative error of only about $2Nu$ of the exact stationary vector. In fact, the relative error incurred by each element of the floating-point stationary vector is about $2Nu$, where u is the unit roundoff (i.e., the maximum relative error in approximating a real number by its nearest floating-point number). This result follows from the fact that the perturbation introduced by storing the matrix in floating-point form is of order unit roundoff. Moreover, in the same paper it is proven that, if GTH is employed in the solution process of this $N \times N$ Markov chain, the relative error in each entry of the stationary vector will be of order N^3u . We also remark that the entrywise relative error in the stationary vector for the GTH algorithm is independent of the structure of the matrix and the magnitude of its elements.

The following lemma shows that the GTH way of calculating the pivot element may also be applied to the transposed system of equations (3.1). It is this form of the equations in which multipliers are bounded by 1, and we wish to demonstrate by numerical experiments that the type of implementation chosen (i.e., transposed versus nontransposed system of equations) has no effect on the accuracy of the results obtained.

Lemma 3. 1 *Let $\mathbf{A} = \mathbf{I} - \mathbf{P}^T$, where $\mathbf{P}$ is a stochastic matrix, and*

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & \mathbf{v}^T \\ \mathbf{u} & \mathbf{A}_{2,2} \end{pmatrix}.$$

Then $e^T \mathbf{A} = \mathbf{0}$ and $e^T \mathbf{u} = -a_{1,1}$. If

$$\mathbf{L}_1 \mathbf{A} = \begin{pmatrix} a_{1,1} & \mathbf{v}^T \\ \mathbf{0} & \tilde{\mathbf{A}}_{2,2} \end{pmatrix},$$

where

$$\mathbf{L}_1 = \begin{pmatrix} 1 & \mathbf{0} \\ -\mathbf{u}/a_{1,1} & \mathbf{I} \end{pmatrix} \quad \mathbf{L}_1^{-1} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{u}/a_{1,1} & \mathbf{I} \end{pmatrix},$$

then $\tilde{\mathbf{A}}_{2,2}$ (like $\mathbf{A}$) is a singular M-matrix having 0 column sums.

Proof:

$$\begin{aligned} e^T \mathbf{A} &= e^T (\mathbf{L}_1^{-1} \mathbf{L}_1) \mathbf{A} = \mathbf{0} \\ e^T \mathbf{L}_1^{-1} &= (1, 1, \dots, 1) \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{u}/a_{1,1} & \mathbf{I} \end{pmatrix} = (0, 1, \dots, 1) \\ (0, 1, \dots, 1) (\mathbf{L}_1 \mathbf{A}) &= \mathbf{0} \Rightarrow e^T \tilde{\mathbf{A}}_{2,2} = \mathbf{0} \end{aligned}$$

Other steps may be shown similarly. Note that the lemma is also valid for the negation of any generator matrix. $\square$

The properties of a singular M-matrix coupled with the GTH idea of avoiding subtractions and negative numbers, suggests the following modification to GE. At each step of GE, rather than calculating the pivot element in the usual way, one can correct the pivot by replacing it with the negated sum of the off-diagonal elements in the unreduced part of the same column as the pivot. When one mentions the GTH method, it is this approach used in calculating the pivot elements that is implied.

3.3 Using GTH in the Disaggregation Step

In a given iteration, the disaggregation step of the IAD method uncouples the NCD chain to obtain a new estimate for the stationary distribution. As indicated in [48], in order to achieve an even better approximation to $\boldsymbol{\pi}$, at the $(k+1)^{st}$ iteration of the IAD algorithm, one solves for $(\boldsymbol{\pi}_i^{(k+1)})^T$, in

$$(\mathbf{I} - \mathbf{P}_{i,i}^T)(\boldsymbol{\pi}_i^{(k+1)})^T = \mathbf{b}_i, \quad (3.2)$$

where

$$\mathbf{b}_i^T = \sum_{j < i} \pi_j^{(k+1)} \mathbf{P}_{j,i} + \sum_{j > i} \mathbf{z}_j^{(k+1)} \mathbf{P}_{j,i}, \quad \text{for } i = 1, 2, \dots, N \quad (3.3)$$

(see step 4(b) of the IAD algorithm) and $\pi_i^{(k+1)}$ is the $(k+1)^{st}$ approximation to π_i .

$\mathbf{P}_{i,i}$ is a strictly substochastic matrix of order n_i , and $\mathbf{b}_i \neq \mathbf{0}$, which gives a non-homogeneous system of linear equations with nonsingular coefficient matrix. Had the coefficient matrix been stochastic, we could clearly employ the same GTH technique that we utilized in solving the coupling matrix. In order to be able to use GTH in solving the diagonal block system, we must first make certain modifications. These modifications involve adding one more equation and augmenting the matrix with $\mathbf{b}_i$ to put the system into the form

$$\mathbf{W}_i^T(\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)})^T = \mathbf{0}, \quad (3.4)$$

where

$$\mathbf{W}_i^T = \begin{pmatrix} \mathbf{I} - \mathbf{P}_{i,i}^T & \mathbf{b}_i \\ \mathbf{w}_i^T & \hat{w}_i \end{pmatrix} \begin{matrix} n_i \\ 1 \end{matrix}, \quad (3.5)$$

$\mathbf{w}_i = \mathbf{P}_{i,*} \mathbf{e}$, $\hat{w}_i = -\mathbf{b}_i^T \mathbf{e}$, and $\hat{\pi}_i^{(k+1)}$ is introduced so that the solution vector has as many columns as the coefficient matrix. In other words, $(\mathbf{w}_i^T, \hat{w}_i)$ sums up the columns of $\mathbf{W}_i^T$ to 0. The values of $\hat{w}_i$ and $\hat{\pi}_i^{(k+1)}$ are irrelevant, because just as in (3.1), we have a singular system with $n_i + 1$ equations, and after $\mathbf{W}_i^T$ is reduced to upper-triangular form, the last row will be all 0's. Hence, the reduction needs to be carried out for only n_i steps. What needs to be done for the computation of $(\pi_i^{(k+1)})^T$ is, to use the first n_i elements of column $n_i + 1$ in the upper-triangular matrix as the right-hand side in the back substitution phase.

It is not possible to put GE to use in the disaggregation step in the previous example in Section 3.2 or in similar problems, on the diagonal blocks for which $\|\mathbf{P}_{i,*}\|_\infty < \epsilon$. Under the given condition, $\mathbf{I} - \mathbf{P}_{i,i}^T$ is a singular matrix in floating-point

form; however, the vector $\mathbf{b}_i$ is still nonzero, and therefore the system of equations in (3.4) is inconsistent. On the other hand, GTH computes $\mathbf{w}_i^T$ in $\mathbf{W}_i^T$ by using the nonzero elements in $\mathbf{P}_{i,*}$ and forms the pivot by summing offdiagonal entries in the unreduced part of the same column as the pivot. Hence, GTH may be applied to solve such blocks.

In the next page we present a unified version of the GTH algorithm written for Matlab that may be used both in the aggregation step and the disaggregation step.

1. Aggregation Step:

The function is called with $\mathbf{P}$ as the coupling matrix, $\mathbf{b}$ and $\mathbf{w}$ as zero vectors with appropriate size. It returns the stationary distribution of the coupling matrix in (3.1).

2. Disaggregation Step:

The function is called with $\mathbf{P}$ as a diagonal block, $\mathbf{w}$ as the vector that sums up the rows of $\mathbf{P}$ to 1, and $\mathbf{b}$ as the corresponding right-hand side. It returns the solution to the nonsingular linear system as discussed in this section.

Note that, part of the second statement in the initialization which computes the diagonal elements of $\mathbf{Q}$ by subtracting 1 from the diagonal elements of $\mathbf{P}$ has no effect. Once $\mathbf{P}$ is transposed, each pivot in the upper-triangular matrix is computed by summing the off-diagonal elements in the unreduced part of the pivot column.

Figure 3.2: Matlab code for GTH

```
function [soln] = gth(P,w,b);

%%%%% The GTH Algorithm for  $(P'-I)$  soln = b %%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[n,n] = size(P);
Q = [P'-eye(n),b;w',-sum(b)];
Q(1,1) = sum(Q(2:n,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GTH Reduction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:n-1,
    sigma = 0;
    for j = i+1:n+1,
        mu = Q(j,i)/Q(i,i);
        Q(j,i+1:n+1) = Q(j,i+1:n+1)+mu*Q(i,i+1:n+1);
        if j>i+1
            sigma = sigma+Q(j,i+1);
        end;
    end;
    Q(i+1,i+1) = sigma;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Set up for Backsub. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

d = Q(1:n,n+1); Q = Q(1:n,1:n);
if b == 0
    Q(n,n) = 1; d(n) = 1;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Back Substitution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = n:-1:1,
    soln(i) = -d(i);
    for j = n:-1:i+1,
        soln(i) = soln(i)+Q(i,j)*soln(j);
    end;
    soln(i) = soln(i)/Q(i,i);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Normalization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if b==0
    soln = soln/sum(soln);
end;
```

3.4 Implementation Considerations

As mentioned before, NCD Markov chains that arise in real-life applications are generally large and sparse. This necessitates the design and employment of sparse storage schemes, which essentially store only the nonzero elements of the transition probability/rate matrix.

So far, we considered the application of the GTH (and GE, for that matter) algorithm to the systems (3.1) and (3.2). We wrote (3.2) in the form of (3.4) so that it has a singular M-matrix with 0 column sums as its coefficient matrix just as in (3.1). Now, let us write the alternative nontransposed systems of equations. The system that corresponds to (3.1) is given by

$$\boldsymbol{\xi}^{(k)}(\mathbf{I} - \mathbf{C}^{(k)}) = \mathbf{o}, \quad \|\boldsymbol{\xi}^{(k)}\|_1 = 1. \quad (3.6)$$

Similarly, the equivalent of (3.4) is

$$(\boldsymbol{\pi}_i^{(k+1)}, \hat{\pi}_i^{(k+1)})\mathbf{W}_i = \mathbf{o}. \quad (3.7)$$

Define:

Scheme 1: The transposed systems of equations in (3.1) and (3.4).

Scheme 2: The nontransposed systems of equations in (3.6) and (3.7).

Unless otherwise specified, by reductions we mean row-reductions (i.e., the addition of a multiple of a lower-indexed row to a higher-indexed row) in a given system of equations. Our last assumption is that the coefficient matrices are supplied to both schemes in the nontransposed version. This is a fairly reasonable assumption, since the matrices are usually generated by following the possible transitions from a given state, implying a row-wise generation.

The advantage of reducing the coefficient matrices in scheme 1 rather than the ones in scheme 2 to upper-triangular form is twofold:

- The multiplier elements at each step of the reduction are bounded by 1.
- Only the upper-triangular matrix needs to be stored during the reduction process.

Although the multipliers in scheme 2 are not necessarily bounded by 1, the growth factor still cannot be greater than 1 as indicated in [49]. However, if row-reductions are carried out in scheme 2, both the upper-triangular matrix and the lower-triangular matrix, which contains the multipliers, have to be stored during the triangularization process. The reason being the fact that, if row-reductions are performed on the nontransposed coefficient matrix, the **LU** decomposition will provide a nonsingular lower-triangular matrix and a singular upper-triangular matrix that has 0's in the last row. Although the decomposition is premultiplied by the stationary vector, one only has to carry out a single substitution phase (see [28], p.724) just as in the transposed system of equations (but this time involving the lower-triangular matrix), thereby making it necessary to store both the lower and the upper-triangular matrix during the reduction. One cure that comes to mind is to store only the upper-triangular matrix during the reduction process and to reconstruct the multiplier matrix after the reduction is over. Though quite straightforward, this solution is inefficient due to the superfluous operations performed in calculating the multipliers for the second time. On the other hand, scheme 1 calls for the transposition of the coefficient matrix before executing GE or GTH.

Now let us pause for a moment and consider the effects of the two implementation schemes on a *semi-systematic row-wise sparse storage format* (see [36]). By a row-wise sparse storage format, we mean a data structure that stores the nonzero elements in the coefficient matrix row by row. By semi-systematic, we mean the elements of row i precede those of row $i + 1$, but the elements within a given row need not be ordered. Together with the real value of each nonzero element, the column index

of the element is stored as well. Hence, two arrays of the same length, one real and the other integer, are needed. Lastly, in order to have access to the rows, the starting location of each row is stored in an integer array. To facilitate the computation of the number of nonzero elements per row, an extra element whose value is equal to the total number of nonzero elements plus 1 is appended to this array.

Row-wise sparse storage format : Three arrays are needed.

a : stores real value of each nonzero element (real array)

ja : stores column index of each nonzero element in **a** (integer array)

ia : stores starting location of each row in **a** (integer array)

Example:

$$A = \begin{pmatrix} -2.1 & 0.0 & 1.7 & 0.4 \\ 0.8 & -0.8 & 0.0 & 0.0 \\ 0.2 & 1.5 & -1.7 & 0.0 \\ 0.0 & 0.3 & 0.2 & -0.5 \end{pmatrix}.$$

Figure 3.3: An example using row-wise sparse storage format

	1	2	3	4	5	6	7	8	9	10	11
a :	-2.1	1.7	0.4	0.8	-0.8	0.2	1.5	-1.7	0.3	0.2	-0.5
ja:	1	3	4	1	2	1	2	3	2	3	4
ia:	1	4	6	9	12						

The GE algorithm, no matter which scheme is chosen, may be implemented using *delayed updates*, meaning, at step k , all reductions on row $k + 1$ may be carried out and the row compacted and stored, before the algorithm continues in the next step with the update of the next row. The property of the GE algorithm which makes this efficient implementation possible is that, reduction on a given row requires the addition of appropriate multiples of lower-indexed rows to it. Hence, there is no data dependency between a row to be reduced and higher indexed rows in GE. The

importance of this idea lies in the fact that, it is sufficient to fully expand only the row to be updated at a given step, whereas all other rows with higher indices can still be held in compact form. However, this is no longer applicable when GTH is used in scheme 1, since all updates due to a row must be finished before the algorithm proceeds with the next step (see the Matlab code in Section 3.3). This situation is dictated by the new way of calculating the pivot elements. In the GTH algorithm, the pivot element is formed by summing the most recently updated elements that lie below the pivot which implies a dependency of data between the pivot row and the higher indexed rows. Since storage is limited, one possibility is to expand each row that is operated on, update it, and finally store it back in the appropriate location. The drawback is that, this sequence of operations on a row, depending on the fill-in, most probably causes other elements in sparse storage to be shifted around the data structure (extra nonzero entries are likely to occur after a row update). Consequently, it is safe to assume that GTH, with such an implementation, spends more time doing memory reads and writes for larger chains. As pointed out in Section 3.2, there is a slight difference in the way the single substitution phase in Scheme 1 and Scheme 2 are implemented in the GTH algorithm when a row-wise sparse storage format is used. In the substitution phase, Scheme 2 calls for the solution of a homogeneous linear system in which the stationary vector is postmultiplied by the lower-triangular matrix obtained from the **LU** decomposition. The substitution is accomplished by accessing a different column of the lower-triangular matrix at each step. This situation makes it necessary to have a doubly-nested loop in the code for accessing the elements of the columns. On the other hand, the substitution phase of Scheme 1 requires the solution of a homogeneous linear system in which the stationary vector is premultiplied by an upper-triangular matrix. This being so, the substitution phase of Scheme 1 conforms nicely to a row-wise sparse storage implementation, and therefore, it may be handled

in a single loop which accesses a row of the upper-triangular matrix at each step. In the actual implementation, the normalization equation (i.e., $\|\boldsymbol{\pi}\|_1 = 1$) is used to discard the last equation and form a nonzero right-hand side in both schemes.

An alternative storage scheme is one that keeps the unused array elements in the form of a linked list (basically, a *freepool*). In this case, an extra integer array which, for each nonzero element, stores the location of the next nonzero element in the same row, is required. If a given element is the last in its row, then the corresponding pointer may be set to 0. Though this scheme circumvents the problem of moving array elements back and forth in memory, it introduces overhead due to the manipulation of the freepool. Note that just as in the ordinary row-wise sparse storage format, all elements in a row may be kept ordered according to their column indices in this last implementation. Let us finally point out that the effect of using a column-wise sparse storage format with scheme 2 is the same as using a row-wise sparse storage format with scheme 1 and vice versa.

Although, not as costly as the extra reads and writes due to the discussed implementation problems, GTH has $N - k$ extra additions at step k of the elimination procedure, amounting to $\sum_{k=1}^{N-1} (N - k) = N(N - 1)/2$ extra operations in the solution of the coupling matrix. Similarly, there are $n_i(n_i - 1)/2$ extra additions in the solution of the i^{th} diagonal block. Clearly, this is an upper bound. Since these extra addition operations are performed only on nonzero elements in a sparse storage implementation, the actual overhead may be smaller. Moreover, because of the need to introduce one more row to the linear system, extra $n_i + 1$ locations are used during the solution of each $\mathbf{P}_{i,i}$ in the disaggregation step. Lastly, in order to form the $(n_i + 1)^{st}$ equation during the disaggregation step, a number of additions equal to the number of nonzero elements in $\mathbf{P}_{i,*}$ are performed. If extra memory locations are available, these N matrix-vector multiplications may be performed at the outset

and the resulting vectors stored for future use.

Let us point out that both programming difficulties and overhead storage (arrays to store the indices of nonzero array elements and pointers between elements) increase with the sophistication of the storage scheme. Together with the row-wise sparse storage format, we also experimented with the sparse storage format proposed by Knuth (see [36] for a brief explanation) which provides equally fast access to columns and rows of a matrix. For each nonzero element, in addition to having a pointer to the next nonzero element in the same row, there is an additional pointer to the next nonzero element in the same column. Obviously, there are two arrays which keep the locations of the first nonzero element in each row and each column, respectively. However, as remarked earlier, extra time will be spent for obtaining memory elements from and returning memory elements to the freepool with this kind of implementation. With Knuth's sparse storage format, there is no need to transpose the matrix because rows may be treated as columns and columns as rows.

Knuth's sparse storage format : Eight arrays are needed.

a : stores real value of each nonzero element (real array)
ja : stores column index of each nonzero element in a (integer array)
ia : stores row index of each nonzero element in a (integer array)
jan : for each nonzero element stores location of next nonzero element in the same column in a (integer array)
ian : for each nonzero element stores location of next nonzero element in the same row in a (integer array)
jaf : stores locations of first nonzero elements in each column in a (integer array)
iaf : stores locations of first nonzero elements in each row in a (integer array)
if : stores locations of free elements in a as a linked list (integer array)

The contents of arrays for the 4×4 example used to explain the row-wise sparse

storage format are shown for Knuth's sparse storage format below.

Figure 3.4: An example using Knuth's sparse storage format

	1	2	3	4	5	6	7	8	9	10	11	12	13	...	la
a:	-2.1	1.7	0.4	0.8	-0.8	0.2	1.5	-1.7	0.3	0.2	-0.5	*	*	*	*
ja:	1	3	4	1	2	1	2	3	2	3	4	*	*	*	*
ia:	1	1	1	2	2	3	3	3	4	4	4	*	*	*	*
jan:	4	8	11	6	7	0	9	10	0	0	0	*	*	*	*
ian:	2	3	0	5	0	7	8	0	10	11	0	*	*	*	*
jaf:	1	5	2	3											
iaf:	1	4	6	9											
if:	*	*	*	*	*	*	*	*	*	*	*	13	14	...	0

There is an integer variable named `is` that points to the beginning of the freepool. In our case, `is = 12`. The size of the array `a` is given by the integer variable `la`.

Although one may be inclined to think that Knuth's sparse storage format is more advantageous than the row-wise sparse storage format, due to the overhead of handling the freepool, numerical experiments suggest that Knuth's format is likely be useful only if implemented in a programming language that provides primitives for pointers to memory.

If some information about the structure of the coefficient matrix is available, then another possibility may be to employ a two-dimensional banded storage structure. As for GE, we recommend the row-wise sparse storage format with scheme 1.

Keeping in mind that memory is much slower than CPU, one must evaluate the significance of extra operations performed and excess time spent in a sparse storage implementation as opposed to the increased stability and accuracy that accrue from GTH, in the context of real-life applications. In Chapter 4, we provide results of IAD with GTH for three different problems, one being a real-life application.

Chapter 4

Applications with GTH in IAD

In the previous chapter, we investigated the effects of using the Grassmann-Taksar-Heyman (GTH) method to solve the coupling matrix formed in the aggregation step of the IAD algorithm. Then, we extended the idea in such a way that the GTH method can be incorporated into the disaggregation step. In this chapter, we demonstrate the effect of using the GTH method in the IAD algorithm on various examples and elaborate on the conditions under which it should be applied.

Although, the GTH algorithm is presented in Section 3.3 in Matlab for clarity, we have experimented with the complete IAD algorithm actually in sparse storage. All routines used are part of the software package MARCA (Markov Chain Analyzer) (see [47]). The routines are written in FORTRAN and compiled in both double-precision and quadruple-precision floating-point arithmetic. These experiments were performed on a SUN SPARC station 2. For each problem solved, the residual error and the relative error in the solution are computed. The relative error is computed as $\|\pi - \tilde{\pi}\|_2 / \|\pi\|_2$, where π is the quadruple-precision solution and $\tilde{\pi}$ is the double-precision solution obtained by the IAD algorithm. Both π and $\tilde{\pi}$ are normalized so that their 1-norms are unity. The residual error is computed as $\|(\mathbf{I} - \mathbf{P}^T)\tilde{\pi}^T\|_2$, which theoretically is 0 if $\pi = \tilde{\pi}$.

We have already shown the problem associated with GE when NCD chains having

a degree of coupling less than machine epsilon are to be solved. Therefore, we are eager to see how GE and GTH behave comparatively for chains having a degree of coupling larger than machine epsilon. We have experimented with implementation schemes 1 and 2 on the row-wise sparse storage format used in MARCA. The GTH implementation of scheme 1 is accomplished by shifting contents of arrays rather than using the alternative with the free pool.

The first two problems appear quite frequently in the literature. Although the order of the matrices considered in these two problems are quite small, it is still instructive to examine the effects of using the IAD algorithm on such problems. The third problem investigated is a real-life example and had been studied with a variety of parameters in the past.

Table 4.1: Notation for parameters of numerical methods

n	Order of the stochastic transition matrix, $\mathbf{P}$
n_z	Number of nonzero elements in the matrix
N	Number of strongly connected components
nzb	Number of nonzero blocks
$Iter$	Number of iterations required to get a residual norm of less than 10^{-15}
T_{total}	Total time spent in the IAD algorithm, (in CPU seconds)
Err_{res}	$\ (\mathbf{I} - \mathbf{P}^T)\tilde{\boldsymbol{\pi}}^T\ _2$
Err_{rel}	$\ \boldsymbol{\pi} - \tilde{\boldsymbol{\pi}}\ _2 / \ \boldsymbol{\pi}\ _2$
γ	Decomposability parameter
M_{agg}	Method used in aggregation phase
M_{disagg}	Method used in disaggregation phase

Table 4.2: Solvers used

GE_1	Sparse Gaussian elimination using Scheme 1
GE_2	Sparse Gaussian elimination using Scheme 2
GTH_2	Sparse Grassmann-Taksar-Heyman algorithm using Scheme 2
GTH_1	Sparse Grassmann-Taksar-Heyman algorithm using Scheme 1

The *decomposability parameter*, γ , is input by the user; it is used to determine the strongly connected components in the transition matrix by simply ignoring the

elements that are less than the suggested value. Therefore, γ may be taken as an approximation to the degree of coupling. If the matrix is not already in the form (1.31), then symmetric permutations are used to put it into the form in which the diagonal blocks form the strongly connected aggregates.

4.1 Test Problem 1

The first problem that we consider is the 8×8 Courtois' matrix ([5]) with all row sums equal to 1. The degree of coupling for this matrix is 0.001.

$$\mathbf{P} = \begin{pmatrix} 0.85 & 0 & 0.149 & 0.0009 & 0 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.65 & 0.249 & 0 & 0.0009 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.8 & 0.0996 & 0.0003 & 0 & 0 & 0.0001 & 0 \\ 0 & 0.0004 & 0 & 0.7 & 0.2995 & 0 & 0.0001 & 0 \\ 0.0005 & 0 & 0.0004 & 0.399 & 0.6 & 0.0001 & 0 & 0 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.6 & 0.2499 & 0.15 \\ 0.00003 & 0 & 0.00003 & 0.00004 & 0 & 0.1 & 0.8 & 0.0999 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.1999 & 0.25 & 0.55 \end{pmatrix}.$$

$$\boldsymbol{\pi}^T = \begin{pmatrix} 0.8928265275450187E - 01 \\ 0.9275763750513320E - 01 \\ 0.4048831201636394E - 01 \\ 0.1585331908198259E + 00 \\ 0.1189382069041751E + 00 \\ 0.1203854811060527E + 00 \\ 0.2777952524492734E + 00 \\ 0.1018192664446740E + 00 \end{pmatrix}.$$

Table 4.3: Results for Problem 1: $n = 8$, $n_z = 41$, $\gamma = 0.001$, $N = 3$, $nzb = 9$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	4	0.04	$0.286E - 16$	$0.824E - 13$
GE_2	GE_2	4	0.04	$0.347E - 16$	$0.904E - 13$
GTH_2	GTH_2	4	0.04	$0.250E - 16$	$0.420E - 15$
GTH_1	GTH_1	4	0.04	$0.208E - 16$	$0.282E - 15$

4.2 Test Problem 2

The second problem that we consider is quite similar to the class of 10×10 matrices used in [34].

$$\mathbf{S} = \begin{pmatrix} 0.1 & 0.3 & 0.1 & 0.2 & 0.3 & \beta & 0 & 0 & 0 & 0 \\ 0.2 & 0.1 & 0.1 & 0.2 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.2 & 0.2 & 0.4 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.1 & 0.2 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0.3 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0.1 & 0.2 & 0.2 & 0.4 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0.3 & 0.2 & 0.2 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0.7 & 0 & 0 & 0.2 \end{pmatrix}$$

If $\mathbf{D} = \text{diag}(\frac{1}{1+\beta}, 1, 1, 1, 1, \frac{1}{1+\beta}, 1, 1, 1, 1)$, then $\mathbf{P} = \mathbf{D}\mathbf{S}$ is a NCD stochastic matrix of order 10, and the degree of coupling is $\frac{\beta}{1+\beta}$.

(a) $\beta = 10^{-7}$

$$\boldsymbol{\pi}^T = \begin{pmatrix} 0.1008045195787271E+00 \\ 0.8012666139606563E-01 \\ 0.3015519514905696E-01 \\ 0.6031039029811392E-01 \\ 0.7926508439180686E-01 \\ 0.1008045195787271E+00 \\ 0.1967651659427390E+00 \\ 0.7003949685919185E-01 \\ 0.1619417899342436E+00 \\ 0.1197871768713281E+00 \end{pmatrix}$$

Table 4.4: Results for Problem 2(a): $n = 10$, $n_z = 48$, $\gamma = 10^{-7}$, $N = 2$, $nzb = 4$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	1	0.03	$0.416E-16$	$0.126E-08$
GE ₂	GE ₂	1	0.03	$0.486E-16$	$0.126E-08$
GTH ₂	GTH ₂	1	0.03	$0.340E-16$	$0.398E-16$
GTH ₁	GTH ₁	1	0.03	$0.349E-16$	$0.239E-15$

(b) $\beta = 10^{-14}$

$$\pi^T = \begin{pmatrix} 0.1008045115305868E + 00 \\ 0.8012666301149126E - 01 \\ 0.3015519575701284E - 01 \\ 0.6031039151402568E - 01 \\ 0.7926508598986232E - 01 \\ 0.1008045115305868E + 00 \\ 0.1967651699097019E + 00 \\ 0.7003949827125116E - 01 \\ 0.1619417931991359E + 00 \\ 0.1197871792863454E + 00 \end{pmatrix}$$

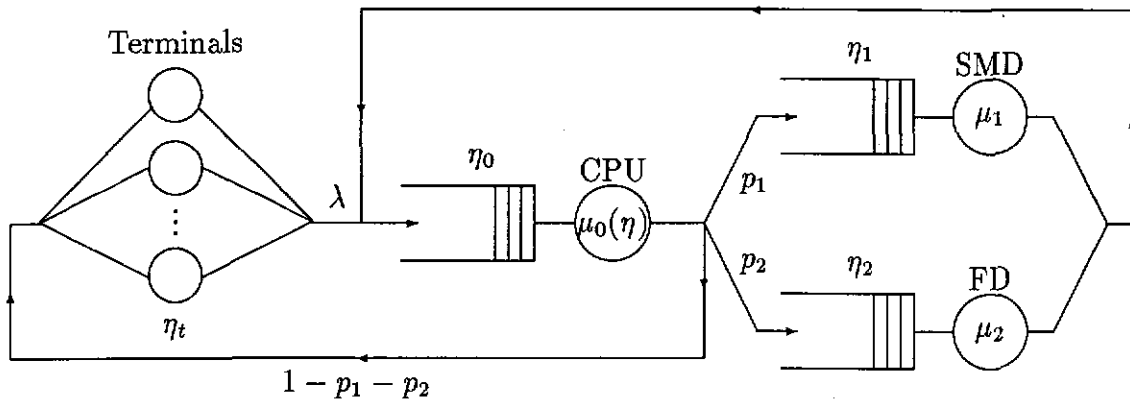
Table 4.5: Results for Problem 2(b): $n = 10$, $n_z = 48$, $\gamma = 10^{-14}$, $N = 2$, $nzb = 4$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	1	0.03	0.423E - 16	0.468E - 02
GE ₂	GE ₂	1	0.03	0.462E - 16	0.448E - 02
GTH ₂	GTH ₂	1	0.03	0.340E - 16	0.328E - 15
GTH ₁	GTH ₁	1	0.03	0.735E - 16	0.167E - 15

4.3 Test Problem 3

The third problem that we consider appears in [46]. It represents a time-shared, multiprogrammed, paged, virtual memory computer system modeled as a closed queueing network (see Figure 4.1).

Figure 4.1: An interactive computer system



The system consists of a number of users using the terminals, a central processing unit (CPU), a secondary memory device (SMD), and a filing device (FD). The degree of multiprogramming in the system at any given time is $\eta = \eta_0 + \eta_1 + \eta_2$, where η_0 , η_1 , η_2 are respectively the number of processes in the CPU, SMD, and FD queues at that moment. Furthermore, η_t represents the number of processes currently being generated at the terminals but not yet transmitted to the CPU for execution. λ is the mean generation time of processes by users working on the terminals. The mean service rate of the CPU depends on η , and is given by $\mu_0(\eta)$. The processes that leave the CPU proceed to the SMD, to the FD, and back to the terminals with probabilities p_1 , p_2 , $1 - p_1 - p_2$ respectively. The mean service rate at the SMD is given by μ_1 and at the FD by μ_2 . All arrival and service rates are exponential. Note that, it is not possible to apply analytical techniques to solve this model since the CPU execution time depends on the degree of multiprogramming.

Given the necessary parameters, MARCA generates the transition probability matrix corresponding to the specific queueing system under consideration. We experiment with different combinations of parameters, because we want to observe the effect of making the transition probability matrix larger, and closer to the identity matrix, so that it is almost decomposable. The generated matrix has a block tridiagonal structure with a small percentage of nonzero elements. We must remark that this type of matrix is frequently encountered in queueing network analysis.

(a) For the first example, the following parameters are chosen.

$$\eta_i + \eta = 3, \quad \lambda = (10^{-4})\eta_i, \quad p_1\mu_0(\eta) = 100(\eta/128)^{1.5}, \quad p_2\mu_0(\eta) = 0.05,$$

$$1 - p_1 - p_2 = 0.002, \quad \mu_1 = 0.2, \quad \mu_2 = 1/30$$

Table 4.6: Sizes of aggregates for Problem 3(a): $\gamma = 10^{-3}$

10	6	3	1
----	---	---	---

Table 4.7: Results for Problem 3(a): $n = 20$, $n_z = 80$, $\gamma = 10^{-3}$, $N = 4$, $nzb = 10$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	5	0.10	$0.468E - 15$	$0.381E - 12$
GE ₂	GE ₂	5	0.10	$0.483E - 15$	$0.391E - 12$
GTH ₂	GTH ₂	5	0.10	$0.447E - 15$	$0.404E - 12$
GTH ₁	GTH ₁	5	0.10	$0.448E - 15$	$0.404E - 12$

(b) The parameters in this example are the same as those in 3(a) except

$$\mu_2 = (10^{-5})/30.$$

Sizes of aggregates for Problem 3(b) for $\gamma = 10^{-6}$ are the same as for Problem 3(a).

Table 4.8: Results for Problem 3(b): $n = 20$, $n_z = 80$, $\gamma = 10^{-6}$, $N = 4$, $nzb = 10$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	4	0.08	$0.429E - 16$	$0.392E - 13$
GE ₂	GE ₂	4	0.08	$0.429E - 16$	$0.392E - 13$
GTH ₂	GTH ₂	4	0.08	$0.363E - 17$	$0.224E - 13$
GTH ₁	GTH ₁	4	0.08	$0.363E - 17$	$0.226E - 13$

(c) The parameters in this example are the same as those in 3(a) except

$$\mu_2 = (10^{-10})/30.$$

Sizes of aggregates for Problem 3(c) for $\gamma = 10^{-11}$ are the same as for Problem 3(a).

Table 4.9: Results for Problem 3(c): $n = 20$, $n_z = 80$, $\gamma = 10^{-11}$, $N = 4$, $nzb = 10$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	1	0.07	$0.974E - 17$	$0.345E - 14$
GE ₂	GE ₂	1	0.07	$0.974E - 17$	$0.345E - 14$
GTH ₂	GTH ₂	1	0.07	$0.999E - 20$	$0.421E - 16$
GTH ₁	GTH ₁	1	0.07	$0.999E - 20$	$0.421E - 16$

(d) The parameters in this example are the same as those in 3(a) except

$$\mu_2 = (10^{-14})/30.$$

Sizes of aggregates for Problem 3(d) for $\gamma = 10^{-15}$ are the same as for Problem 3(a).

Table 4.10: Results for Problem 3(d): $n = 20$, $n_z = 80$, $\gamma = 10^{-15}$, $N = 4$, $nzb = 10$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	1	0.02	$0.551E - 16$	$0.204E - 13$
GE ₂	GE ₂	1	0.02	$0.551E - 16$	$0.204E - 13$
GTH ₂	GTH ₂	1	0.02	$0.846E - 28$	$0.354E - 24$
GTH ₁	GTH ₁	1	0.02	$0.841E - 28$	$0.354E - 24$

(e) The parameters in this example are the same as those in 3(a) except

$$\eta_t + \eta = 10.$$

Table 4.11: Sizes of aggregates for Problem 3(e): $\gamma = 10^{-3}$

1	3	6	10	15	21	28
36	45	55	66			

Table 4.12: Results for Problem 3(e): $n = 286$, $n_z = 1,606$, $\gamma = 10^{-3}$, $N = 11$, $nzb = 31$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	8	0.47	$0.519E-16$	$0.142E-11$
GE ₂	GE ₂	8	0.64	$0.969E-16$	$0.469E-11$
GTH ₂	GTH ₂	8	1.18	$0.412E-16$	$0.233E-12$
GTH ₁	GTH ₁	8	2.72	$0.449E-16$	$0.233E-12$

(f) The parameters in this example are the same as those in 3(e).

Table 4.13: Sizes of aggregates for Problem 3(f): $\gamma = 10^{-4}$

65	55	165
----	----	-----

Table 4.14: Results for Problem 3(f): $n = 286$, $n_z = 1,606$, $\gamma = 10^{-4}$, $N = 3$, $nzb = 7$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	5	0.88	$0.583E-16$	$0.178E-11$
GE ₂	GE ₂	5	1.09	$0.531E-16$	$0.132E-11$
GTH ₂	GTH ₂	5	2.89	$0.398E-16$	$0.383E-12$
GTH ₁	GTH ₁	5	13.84	$0.350E-16$	$0.383E-12$

(g) The parameters in this example are the same as those in 3(e) except

$$\mu_2 = (10^{-5})/30.$$

Sizes of aggregates for Problem 3(g) for $\gamma = 10^{-6}$ are the same as for Problem 3(e).

Table 4.15: Results for Problem 3(g): $n = 286$, $n_z = 1,606$, $\gamma = 10^{-6}$, $N = 11$, $nzb = 31$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	3	0.20	$0.254E-15$	$0.675E-11$
GE ₂	GE ₂	3	0.27	$0.254E-15$	$0.675E-11$
GTH ₂	GTH ₂	3	0.44	$0.293E-15$	$0.671E-11$
GTH ₁	GTH ₁	3	1.01	$0.293E-15$	$0.671E-11$

(h) The parameters in this example are the same as those in 3(e) except

$$\mu_2 = (10^{-5})/30.$$

Table 4.16: Sizes of aggregates for Problem 3(h): $\gamma = 10^{-4}$

11	10	45	10	9	36	9	8	28	8	7	21
7	6	15	6	5	10	5	4	6	4	3	3
3	2	1	2	1	1						

Table 4.17: Results for Problem 3(h): $n = 286$, $n_z = 1,606$, $\gamma = 10^{-4}$, $N = 30$, $nzb = 88$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	2	0.11	$0.232E-15$	$0.193E-12$
GE ₂	GE ₂	2	0.16	$0.232E-15$	$0.193E-12$
GTH ₂	GTH ₂	2	0.23	$0.273E-15$	$0.155E-12$
GTH ₁	GTH ₁	2	0.34	$0.273E-15$	$0.155E-12$

(i) The parameters in this example are the same as those in 3(e) except

$$\mu_2 = (10^{-5})/30.$$

Table 4.18: Sizes of aggregates for Problem 3(i): $\gamma = 10^{-3}$

1	2	1	3	2	4	1	3	5	2	4	6	1	3	5	7
2	4	6	8	1	3	5	7	9	2	4	6	8	10		
1	3	5	7	9	11	2	4	6	8	10	1	3	5	7	9
2	4	6	8	1	3	5	7	2	4	6	1	3	5	2	4
1	3	2	1												

Table 4.19: Results for Problem 3(i): $n = 286$, $n_z = 1,606$, $\gamma = 10^{-3}$, $N = 66$, $nzb = 196$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	3	0.16	$0.247E-16$	$0.469E-13$
GE ₂	GE ₂	3	0.24	$0.247E-16$	$0.469E-13$
GTH ₂	GTH ₂	3	0.39	$0.111E-15$	$0.107E-14$
GTH ₁	GTH ₁	3	0.49	$0.111E-15$	$0.107E-14$

(j) The parameters in this example are the same as those in 3(a) except

$$\eta_t + \eta = 20 \text{ and } \lambda = (10^{-7})\eta_t.$$

Table 4.20: Sizes of aggregates for Problem 3(j): $\gamma = 10^{-6}$

1	3	6	10	15	21	28
36	45	55	66	78	91	105
120	136	153	171	190	210	231

Table 4.21: Results for Problem 3(j): $n = 1,771$, $n_z = 11,011$, $\gamma = 10^{-6}$, $N = 21$, $nzb = 61$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	3	3.79	$0.234E-16$	$0.263E-12$
GE ₂	GE ₂	3	4.11	$0.234E-16$	$0.263E-12$
GTH ₂	GTH ₂	3	15.12	$0.514E-18$	$0.605E-14$
GTH ₁	GTH ₁	3	82.47	$0.514E-18$	$0.605E-14$

(k) The parameters in this example are the same as those in 3(a) except

$$\eta_t + \eta = 20, \mu_1 = 0.2(10^{-10}), \text{ and } \mu_2 = (10^{-10})/30.$$

Sizes of aggregates for Problem 3(k) for $\gamma = 10^{-12}$ are the same as for Problem 3(j).

Table 4.22: Results for Problem 3(k): $n = 1,771$, $n_z = 11,011$, $\gamma = 10^{-12}$, $N = 21$, $nzb = 61$

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE ₁	GE ₁	3	2.23	$0.390E-16$	$0.466E-03$
GE ₂	GE ₂	3	2.88	$0.390E-16$	$0.466E-03$
GTH ₂	GTH ₂	3	9.12	$0.399E-16$	$0.583E-15$
GTH ₁	GTH ₁	3	26.89	$0.399E-16$	$0.584E-15$

In all of the problems considered, both IAD with GE and IAD with GTH converged (see *Iter* in Table 4.1) with a residual of order machine epsilon irregardless of the implementation scheme chosen. For each problem, the number of iterations taken by both methods is the same. However, the relative error in IAD with GE is much larger than that of IAD with GTH in the first two small problems where IAD with GTH has a relative error almost of order machine epsilon. We presume this is due to the smaller size of the systems solved in these two problems. Both of these problems have small coefficient matrices giving rise to coupling matrices of size 3×3 and 2×2 , respectively. Similarly, the largest systems solved in the disaggregation step of the IAD algorithm are of size 3×3 and 5×5 , respectively. Consequently, we expect to lose a relative accuracy of roughly 2 digits in both problems when IAD with GTH is used. The results confirm that we do even better than our expectations and lose only a single digit of accuracy in each problem. However, as in the third problem, there are cases where the GTH algorithm does not achieve its worst case bound due to statistical effects in rounding error accumulation although the matrices solved are considerably larger. Just as indicated in [32], it is possible to improve the entrywise relative error in the GTH algorithm by forming the pivot element in higher precision (quadruple-precision in our case).

As expected, the number of iterations taken to achieve the prespecified tolerance criterion decreases as the decomposability parameter becomes smaller for a given problem. Among the problems considered, when the decomposability parameter is greater than or equal to 10^{-6} , IAD with GE provided results in which 3 to 5 digits of accuracy is lost. An observation regarding the results of Problem 3 is the difference between T_{total} 's for implementation schemes 1 and 2. Firstly, although it is not the case for the third problem, the nonzero structure of a nonsymmetric matrix changes if the matrix is transposed, thus totally affecting the reduction process. Secondly, both

methods in scheme 2 store the multipliers, and thus they spend extra time. Finally, remember that scheme 1 implementation of GTH is accomplished by shifting data around in memory, a very time consuming process.

4.4 Conclusion

Our interest lies in the computation of the stationary probability vectors of ill-conditioned NCD Markov chains. The GTH method, which avoids subtractions, is a much more stable version of GE. For that reason, it is a good candidate to be used in the two-level IAD algorithm. We have experimented on several problems, applying this idea versus GE in the IAD technique. The GTH approach to calculating the pivot element by taking the negated sum of the off-diagonal elements in the unreduced part of the pivot column proves to be valuable for singular M-matrices with 0 column sums, and it is shown to be quite effective on the problems of interest.

We employed the GTH idea to solve the linear systems of equations formed in both the aggregation and disaggregation steps of the IAD algorithm. For the small problems, we observed that as the NCD chain becomes more ill-conditioned, the relative error in IAD with GE increases drastically, whereas it remains fixed in the order of machine epsilon when GTH is used. In all cases, the size of the systems solved provide an estimation for the worst case bound on the relative error for IAD with GTH. Additionally, just as explained in Section 3.4, it was of no surprise to see IAD with GTH take on the order of 10 times as much time as IAD with GE for the larger chains in Problem 3. We also verified for IAD with GE that a significant difference between the relative error and the residual error is a clear indication of an ill-conditioned problem.

In conclusion, ordinary GE should definitely be avoided in both steps of the iterative IAD algorithm when solving NCD chains with a degree of coupling less

than machine epsilon. However, if an approximation to the stationary vector of a large NCD Markov chain is sought in a short time, IAD with GE may be used. On the contrary, if relative error in the stationary vector of the NCD chain is deemed as of utmost importance, then IAD with GTH has to be recommended. However, we do recommend IAD with GE when the decomposability parameter for a given problem is greater than or equal to 10^{-6} . Only 3 to 5 digits of accuracy was lost in such problems considered. Examination of several sparse storage formats for both GTH and GE has indicated one disadvantage of the GTH method. The time to execute the IAD algorithm on large irreducible NCD Markov chains tends to be longer when the GTH method is used in the aggregation and disaggregation steps of the iterative solver. Memory requirement of the GTH algorithm is generally slightly higher than that of GE; nevertheless this mostly depends on the sparse storage format chosen. One last remark would be to direct the attention to the possibility of exploiting the inherent parallelism in the formation of the coupling matrix. Similarly, parallel implementation of the solution of N nonsingular systems in the disaggregation step needs to be investigated.

Chapter 5

AM Method for Markov Chains

We are interested in solving (2.1) by iterative methods that are suitable for implementation on a multiprocessor system. One such technique is the *arithmetic mean (AM) method*, which is actually a family of algorithms for the solution of linear systems. In a series of papers ([11, 12, 13, 37, 38]), several of these schemes were studied and implemented on multivector computers, such as the Cray Y-MP, and tested on various nonsingular linear systems arising from discrete approximations to parabolic and elliptic partial differential equations. The numerical experiments showed that these methods compare favorably with existing techniques such as the biconjugate gradient method, particularly when $\mathbf{A}$ is strongly asymmetric ([37, 38]). It should be mentioned, however, that in the symmetric positive definite case the arithmetic mean method can be used as an efficient additive preconditioner for the conjugate gradient algorithm ([13, 1]).

The aim of this chapter is to extend the arithmetic mean method for large, sparse systems of linear equations to the case where $\mathbf{A}$ is a singular, irreducible M-matrix, and to observe the convergence behavior of the method on a meaningful model problem. The necessary background on splitting methods is provided in Section 2.2. In Section 5.1, we introduce the method of arithmetic mean and prove its convergence. In Section 5.2, we describe a simple reliability model on which experiments were con-

ducted. In Section 5.3, we report on some numerical experiments with the reliability model, where two forms of the arithmetic mean method are compared. In Section 5.4, we draw conclusions.

5.1 Arithmetic Mean Method

Let us consider two distinct splittings of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{H}_1 - \mathbf{K}_1 = \mathbf{H}_2 - \mathbf{K}_2.$$

The arithmetic mean method for solving (2.1) can be formulated as follows:

$$\mathbf{H}_1 \mathbf{w}_1^{(k)} = \mathbf{K}_1 \mathbf{x}^{(k)} \quad (5.1.1)$$

$$\mathbf{H}_2 \mathbf{w}_2^{(k)} = \mathbf{K}_2 \mathbf{x}^{(k)} \quad (5.1.2)$$

$$\tilde{\mathbf{x}}^{(k+1)} = \frac{1}{2}(\mathbf{w}_1^{(k)} + \mathbf{w}_2^{(k)})$$

$$\mathbf{x}^{(k+1)} = (1 - \alpha)\mathbf{x}^{(k)} + \alpha\tilde{\mathbf{x}}^{(k+1)}$$

($k = 0, 1, \dots$); here $\mathbf{x}^{(0)}$ is an initial approximation to the solution, and $\alpha \in (0, 1)$ a parameter. Notice that the two linear systems (5.1.1)–(5.1.2) can be solved in parallel. It should be mentioned that in this general form, the arithmetic mean method is, at least formally (if not in spirit), a special case of a *multisplitting* method. Multisplittings were first introduced in [33], and their application to singular M-matrices was considered in [20], where graph theory was used to study consistency and convergence. To prove the convergence of the arithmetic mean method, however, graph theory is not needed. Indeed, we have the following result.

Theorem 5.1 *Let $\mathbf{A}$ be a singular, irreducible M-matrix, and assume that the following conditions are both satisfied:*

- (i) *Splittings $\mathbf{A} = \mathbf{H}_1 - \mathbf{K}_1 = \mathbf{H}_2 - \mathbf{K}_2$ are weak regular;*

(ii) Matrix $\mathbf{H}_1 + \mathbf{H}_2$ is nonsingular.

Then, for each initial vector $\mathbf{x}^{(0)}$ and $\alpha \in (0, 1)$, the sequence $\{\mathbf{x}^{(k)}\}$ generated by the arithmetic mean method is convergent to a solution of (2.1).

Proof: By hypothesis (i), the matrix $\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}$ is nonnegative; by (ii) it is also nonsingular, since $\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1} = \mathbf{H}_1^{-1}(\mathbf{H}_1 + \mathbf{H}_2)\mathbf{H}_2^{-1}$. Hence, we can consider a single splitting of the form $\mathbf{A} = \mathbf{M} - \mathbf{N}$, where $\mathbf{M} = 2(\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1})^{-1}$. But $\mathbf{T} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A} = \frac{1}{2}(\mathbf{H}_1^{-1}\mathbf{K}_1 + \mathbf{H}_2^{-1}\mathbf{K}_2)$ is nonnegative, hence $\mathbf{A} = \mathbf{M} - \mathbf{N}$ is a weak regular splitting and $\mathbf{T}$ is weakly semiconvergent. By Theorem 2.1, the matrix $\mathbf{T}_\alpha = \mathbf{I} - \alpha\mathbf{M}^{-1}\mathbf{A}$ is semiconvergent for all $\alpha \in (0, 1)$; but $\mathbf{T}_\alpha$ is precisely the iteration matrix in the method of arithmetic mean. $\square$

This theorem extends to the singular case some of the convergence results given in [33, 13] and [37].

There are several possible choices of the matrices $\mathbf{H}_1, \mathbf{H}_2$ such that the splittings $\mathbf{A} = \mathbf{H}_i - \mathbf{K}_i$ ($i = 1, 2$) satisfy the assumptions of Theorem 5.1. In addition, $\mathbf{H}_1$ and $\mathbf{H}_2$ must be chosen so that the two linear systems (5.1.1)–(5.1.2) are easily solved; as we shall see, in some important special cases it is possible to introduce additional parallelism in the algorithm by a judicious selection of $\mathbf{H}_1$ and $\mathbf{H}_2$.

The first two examples were given in [13] and [37]. Let $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$ ² with $\mathbf{D}$ diagonal, $\mathbf{L}$ strictly lower-triangular and $\mathbf{U}$ strictly upper-triangular. Also, let $\rho \geq 0$ and let $\mathbf{W}$ be a positive diagonal matrix. Consider the splittings

$$\begin{aligned}\mathbf{A} &= \mathbf{H}_1 - \mathbf{K}_1 = (\mathbf{D} + \rho\mathbf{W}) - \mathbf{L} - (\rho\mathbf{W} + \mathbf{U}) \\ &= \mathbf{H}_2 - \mathbf{K}_2 = (\mathbf{D} + \rho\mathbf{W}) - \mathbf{U} - (\rho\mathbf{W} + \mathbf{L}).\end{aligned}$$

These are regular splittings of $\mathbf{A}$, and furthermore $\mathbf{H}_1 + \mathbf{H}_2 = \mathbf{A} + \mathbf{D} + 2\rho\mathbf{W}$,

² $\mathbf{L}$ and $\mathbf{U}$ should not be confused with the lower and upper-triangular matrices in the LU decomposition.

a nonsingular matrix for all $\rho \geq 0$ (see [2], p.152). Therefore the assumptions in Theorem 5.1 are fulfilled, and the arithmetic mean method is convergent for all $\alpha \in (0, 1)$ and nonnegative ρ .

Another possible choice is the following:

$$\begin{aligned}\mathbf{A} &= \mathbf{H}_1 - \mathbf{K}_1 = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{L}\right) - \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{U}\right) \\ &= \mathbf{H}_2 - \mathbf{K}_2 = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{U}\right) - \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{L}\right).\end{aligned}$$

For $0 < \omega \leq 1$ the splittings are regular and $\mathbf{H}_1 + \mathbf{H}_2 = \mathbf{A} + \frac{2-\omega}{\omega}\mathbf{D}$ is nonsingular. Again, Theorem 5.1 can be applied and convergence is ensured for all $\alpha \in (0, 1)$ and $\omega \in (0, 1]$. In this particular example, steps (5.1.1) and (5.1.2) correspond, respectively, to a forward and a backward SOR sweep, to be carried out concurrently. The results of these two sweeps are averaged to obtain $\tilde{\mathbf{x}}^{(k+1)}$; finally, the new approximate value $\mathbf{x}^{(k+1)}$ is formed as a convex combination of $\mathbf{x}^{(k)}$ and $\tilde{\mathbf{x}}^{(k+1)}$.

The parameters ρ , ω and α can all be regarded as acceleration parameters. In general, it is difficult to have good estimates for the optimal value of these parameters. Concerning α , some results are available in case that $\mathbf{T} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$ has only real eigenvalues, but in order to be applied one needs information on the spectrum of $\mathbf{T}$ that may be impossible to obtain (see [31]).

Other possible choices for the splittings are suggested by the *alternating group explicit* (AGE) class of parallel iterative methods; see [9], for instance. These methods can be used when $\mathbf{A}$ is tridiagonal or block tridiagonal, a condition often fulfilled in practice, at least for applications to PDE's and Markov chains. For example, let

$$\mathbf{A} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & & & \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \\ & & & & \mathbf{A}_{p-1} & \mathbf{B}_{p-1} & \mathbf{C}_{p-1} \\ & & & & & \mathbf{A}_p & \mathbf{B}_p \end{pmatrix} \quad (5.2)$$

where all the matrices $\mathbf{A}_i$, $\mathbf{C}_i$ are nonpositive and the $\mathbf{B}_i$'s are nonsingular M-matrices. Now, let

$$\tilde{\mathbf{H}}_1 = \begin{pmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & & \\ \mathbf{A}_2 & \mathbf{B}_2 & & & \\ & & \mathbf{B}_3 & \mathbf{C}_3 & \\ & & \mathbf{A}_4 & \mathbf{B}_4 & \\ & & & \ddots & \\ & & & & \mathbf{B}_{p-1} & \mathbf{C}_{p-1} \\ & & & & \mathbf{A}_p & \mathbf{B}_p \end{pmatrix} \quad (5.3)$$

and

$$\tilde{\mathbf{H}}_2 = \begin{pmatrix} \mathbf{B}_1 & & & & \\ & \mathbf{B}_2 & \mathbf{C}_2 & & \\ & \mathbf{A}_3 & \mathbf{B}_3 & & \\ & & & \mathbf{B}_4 & \mathbf{C}_4 \\ & & & \mathbf{A}_5 & \mathbf{B}_5 \\ & & & & \ddots \\ & & & & & \mathbf{B}_p \end{pmatrix} \quad (5.4)$$

(we are assuming that p is even, without loss of generality). Also, let $\tilde{\mathbf{K}}_i = \mathbf{A} - \tilde{\mathbf{H}}_i$ for $i = 1, 2$. Now take ρ to be any positive real scalar, and let $\mathbf{H}_i = \tilde{\mathbf{H}}_i + \rho\mathbf{I}$ and $\mathbf{K}_i = \rho\mathbf{I} - \tilde{\mathbf{K}}_i$ ($i = 1, 2$). The splittings

$$\begin{aligned} \mathbf{A} &= \mathbf{H}_1 - \mathbf{K}_1 = (\tilde{\mathbf{H}}_1 + \rho\mathbf{I}) - (\rho\mathbf{I} - \tilde{\mathbf{K}}_1) \\ &= \mathbf{H}_2 - \mathbf{K}_2 = (\tilde{\mathbf{H}}_2 + \rho\mathbf{I}) - (\rho\mathbf{I} - \tilde{\mathbf{K}}_2) \end{aligned} \quad (5.5)$$

are regular and the matrix $\mathbf{H}_1 + \mathbf{H}_2$ is nonsingular for all $\rho > 0$, and Theorem 5.1 applies. This approach can be useful on a parallel computer with q processors, where q is a divisor of p . If p processors are available, each of the two independent tasks (5.1) breaks down into $p/2$ independent subtasks that can be executed concurrently. For the important special case where the diagonal blocks $\mathbf{B}_i$ are tridiagonal matrices and the off-diagonal blocks are diagonal matrices, the independent subtasks can be

efficiently handled on a multivector computer by cyclic reduction (this was shown in [38], where this form of the arithmetic mean method was first proposed). Variants on this choice of the splittings may also be considered. For instance, it is possible to replace each $\mathbf{B}_i$ in $\tilde{\mathbf{H}}_1$ and $\tilde{\mathbf{H}}_2$ by $\mathbf{D}_i$, where $\mathbf{B}_i = \mathbf{D}_i - \mathbf{L}_i - \mathbf{U}_i$ is the standard splitting into strictly lower-triangular, diagonal and strictly upper-triangular parts. This leads to slower convergence of the method, but now less operations per iteration are required. In the next section we describe a Markov chain problem for which these forms of the arithmetic mean method are well suited.

5.2 A Reliability Model

We consider a simple reliability model with 2 different classes of machines. In order to keep the model small for the time being, we assume there are 3 machines in each class. Each machine is subject to breakdown and a subsequent repair. A state is completely specified by the ordered pair (n_1, n_2) , where n_1 denotes the number of intact machines of the first class and n_2 denotes the number of intact machines of the second class. Since both n_1 and n_2 can assume values from 0 to 3, the total number of possible states is $(3 + 1)(3 + 1) = 16$. We order these 16 states as follows:

state :	(3, 3)	(3, 2)	(3, 1)	(3, 0)	(2, 3)	(2, 2)	(2, 1)	(2, 0)	(1, 3)	(1, 2)	(1, 1)	(1, 0)	(0, 3)	(0, 2)	(0, 1)	(0, 0)
index :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

The times between successive breakdowns and successive repairs are both exponentially distributed. The breakdown rates of class 1 machines and class 2 machines are respectively λ_1 and λ_2 . Similarly, the repair rates of the two classes of machines are μ_1 and μ_2 . The transition rate matrix for the described reliability model is then given by $\mathbf{Q}$ on the next page. The diagonal elements indicated by asterisks are the negated sums of the off-diagonal elements in their corresponding rows. It can be

shown that the matrix is block (4×4) tridiagonal with the additional property that the diagonal blocks themselves are tridiagonal matrices, and the subdiagonal and superdiagonal blocks are diagonal matrices. The specified ordering places all nonzero entries due to breakdowns above the main diagonal and all nonzero entries due to repairs below the main diagonal. The nonzero entries in the lower and upper-diagonals of the diagonal blocks represent breakdown or repair of machines of the second class, whereas the nonzero entries in the off-diagonal blocks represent breakdown or repair of machines of the first class. $\mathbf{Q}$ is asymmetric if the repair rates are not equal to the breakdown rates.

$$\mathbf{Q} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \end{matrix} & \left(\begin{array}{cccccccccccccccc} * & 3\lambda_2 & & & 3\lambda_1 & & & & & & & & & & & \\ \mu_2 & * & 2\lambda_2 & & & 3\lambda_1 & & & & & & & & & & \\ & 2\mu_2 & * & \lambda_2 & & & 3\lambda_1 & & & & & & & & & \\ & & 3\mu_2 & * & & & & 3\lambda_1 & & & & & & & & \\ \mu_1 & & & & * & 3\lambda_2 & & & 2\lambda_1 & & & & & & & \\ & \mu_1 & & & \mu_2 & * & 2\lambda_2 & & & 2\lambda_1 & & & & & & \\ & & \mu_1 & & & 2\mu_2 & * & \lambda_2 & & & 2\lambda_1 & & & & & \\ & & & \mu_1 & & & 3\mu_2 & * & & & & 2\lambda_1 & & & & \\ & & & & 2\mu_1 & & & & * & 3\lambda_2 & & & \lambda_1 & & & \\ & & & & & 2\mu_1 & & & \mu_2 & * & 2\lambda_2 & & & \lambda_1 & & \\ & & & & & & 2\mu_1 & & & 2\mu_2 & * & \lambda_2 & & & \lambda_1 & \\ & & & & & & & 2\mu_1 & & & 3\mu_2 & * & & & & \lambda_1 \\ & & & & & & & & 3\mu_1 & & & & * & 3\lambda_2 & & \\ & & & & & & & & & 3\mu_1 & & & \mu_2 & * & 2\lambda_2 & \\ & & & & & & & & & & 3\mu_1 & & & 2\mu_2 & * & \lambda_2 \\ & & & & & & & & & & & 3\mu_1 & & & & 3\mu_2 & * \end{array} \right) \end{matrix}.$$

Let π be the stationary distribution of the underlying Markov chain. Our goal is to solve the singular system $\pi\mathbf{Q} = \mathbf{0}$ subject to $\|\pi\|_1 = 1$ (the normalization condition). However, one can solve the equivalent system $-\mathbf{Q}^T\pi^T = \mathbf{0}$ instead. Here, the coefficient matrix is a singular irreducible M-matrix and the theory developed in the previous sections applies. From this point on, we let $\mathbf{A} = -\mathbf{Q}^T$ and $\mathbf{x} = \pi^T$, so that we can use the notation introduced in the earlier sections. Observe that, if the repair rates are much larger than the breakdown rates, the mass is accumulated in the upper-triangular part of $-\mathbf{Q}^T$. As mentioned earlier, the arithmetic mean method is most effective in the solution of nonsingular linear systems with such

highly asymmetric coefficient matrices. We expect to notice the same effect in the solution of singular linear systems.

$\mathbf{A}$ may be written in block form as

$$\mathbf{A} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & \\ & \mathbf{A}_3 & \mathbf{B}_3 & \mathbf{C}_3 \\ & & \mathbf{A}_4 & \mathbf{B}_4 \end{pmatrix},$$

where $\mathbf{C}_i = -i\mu_1\mathbf{I}$, $i = 1, 2, 3$, $\mathbf{A}_i = -(4 - i + 1)\lambda_1\mathbf{I}$, $i = 2, 3, 4$, and

$$\mathbf{B}_i = \begin{pmatrix} -* & -\mu_2 & & \\ -3\lambda_2 & -* & -2\mu_2 & \\ & -2\lambda_2 & -* & -3\mu_2 \\ & & -\lambda_2 & -* \end{pmatrix}, \quad i = 1, 2, 3, 4.$$

In Section 5.3 we report on numerical experiments with the reliability model in which the number of machines in the two classes is much larger. If there are $p - 1$ machines in the first class and $m - 1$ in the second class, then $\mathbf{A}$ will be block $p \times p$, where each block is $m \times m$ (see (5.2)), and the number of states is $n = pm$. Similarly structured matrices arise in the modeling of queueing networks (see [19]). Note that this simple model problem is chosen for the sake of illustration and it admits a closed form (product form) solution which may be computed in the following way.

The two classes of machines are 'isomorphic' in the sense that they behave the same; both have exponentially distributed breakdowns and repairs (probably with different parameters). Moreover, there is no interaction between the two classes of machines. It is, therefore, possible to analyze each class of machine separately. Once the stationary probability of having k_1 (k_2) intact machines in class 1 (2) is found, the joint stationary probability of having k_1 intact machines of the first class and k_2 intact machines of the second class may be computed by multiplying the individual stationary probabilities. To put it in another way, $\pi(k_1, k_2) = \pi_1(k_1)\pi_2(k_2)$, where $\pi_1(k_1)$ ($\pi_2(k_2)$) is the stationary probability of having k_1 (k_2) intact machines

in class 1 (2), $\pi(k_1, k_2)$ is the joint stationary probability mentioned before, and $0 \leq k_1 \leq p-1$, $0 \leq k_2 \leq m-1$.

Without loss of generality, let us concentrate on class 1 machines. The reliability model for class 1 machines may be perceived as an M/M/ $p-1/p-1/p-1$ (i.e., $p-1$ server, finite storage, finite population) queue (see [22], pp.108–109). Hence, the stationary probability of having k_1 class 1 intact machines is

$$\pi_1(k_1) = \pi_1(p-1) \left(\frac{\lambda_1}{\mu_1} \right)^{p-1-k_1} \binom{p-1}{k_1}, \quad 0 \leq k_1 \leq p-1. \quad (5.6)$$

Similarly, the stationary probability of having k_2 class 2 intact machines is given by

$$\pi_2(k_2) = \pi_2(m-1) \left(\frac{\lambda_2}{\mu_2} \right)^{m-1-k_2} \binom{m-1}{k_2}, \quad 0 \leq k_2 \leq m-1. \quad (5.7)$$

One should first find $\pi_1(p-1)$ and $\pi_2(m-1)$ through the normalization equations $\|\pi_1\|_1 = 1$ and $\|\pi_2\|_1 = 1$, respectively. Thereafter, remaining elements of π_1 and π_2 may be obtained using (5.6) and (5.7). The explicit formulae for $\pi_1(p-1)$ and $\pi_2(m-1)$ may also be written down, but they are cumbersome, and we believe they would be of minor help.

However, most real-life Markov models cannot be solved explicitly and therefore call for iterative solution methods. One such Markov chain is the multi-media traffic queue on an ATM (Asynchronous Transfer Mode) link with overload control ([52]). Other chains with the same property may be found in systems where servers have service rates which depend on the class and/or number of customers (see [35]). These models lead to block tridiagonal matrices but cannot be solved explicitly. Therefore, the arithmetic mean method is well suited for their solution.

5.3 Numerical Experiments

In this section we report on the results of numerical experiments. A few of them are summarized in Tables 5.1 and 5.2. We compared two forms of the arithmetic

mean method, denoted by AM_1 and AM_2 . We use form (5.5) of the arithmetic mean method, where AM_1 corresponds to $\mathbf{H}_1$ and $\mathbf{H}_2$ in (5.3)–(5.4) and AM_2 corresponds to the same matrices except that the $\mathbf{B}_i$ blocks are now replaced by their diagonal parts $\mathbf{D}_i$, as explained at the end of Section 5.1.

The experiments were carried out for reliability models of different size (up to $n = 90,000$ states; for convenience we set $p = m$ so that $n = p^2$) and for several choices of the parameters λ_i, μ_i . All calculations were performed on the Cray Y-MP at North Carolina Supercomputing Center, using single precision. In our tables we report the number of iterations required by each method to satisfy the stopping criterion (convergence was considered achieved when $\|\mathbf{r}^{(k)}\|_2 < 10^{-12}$, where $\mathbf{r}^{(k)} = \mathbf{A}\mathbf{x}^{(k)}$ denotes the residual after k iterations). However, it is clear that the number of iterations alone is not a good basis for comparison, since the two iterative schemes have different arithmetic complexity per iteration. The more expensive single iteration corresponds to AM_1 , requiring the solution of p pentadiagonal systems in p unknowns at each step. The cheaper scheme is AM_2 , which requires only the solution of $2p$ diagonal systems of size p per iteration. The two schemes also differ in the number of operations required in forming the right-hand sides. In the actual implementation, all the necessary matrix factorizations are done at the outset, and only computation of the new right-hand sides and triangular solves are performed at each iteration. Notice that no solves are actually needed for AM_2 , which is in fact an explicit method.

The computational complexity for an n -state problem (with $n = p^2$) is easily established: AM_1 requires about $21n$ multiplicative operations at the outset and about $17n$ operations per iteration; for AM_2 the cost is about $7n$ operations at the outset plus $11n$ operations per iteration. Notice that storage requirements are lower for AM_2 than for AM_1 .

Methods AM_1 and AM_2 are well suited for implementation on a multiprocessor system with p processors, since each of the two independent tasks in (3.1) breaks down into $p/2$ subtasks which can be executed concurrently with perfect balancing of the workload among the processors. The averaging operation following (3.1) requires synchronization of the processors, and is bound to introduce some overhead cost. Nevertheless, very high efficiency can be obtained through a careful implementation, as reported in the literature on the arithmetic mean method.

In Table 5.1 we consider two reliability models of sizes $n = 2,500$ and $n = 10,000$. The breakdown rates are $\lambda_1 = 1.0$, $\lambda_2 = 2.0$ and the repair rates are given in the table under μ_1, μ_2 . For each method, we report the number of iterations corresponding to two choices of the relaxation parameters. These are $\alpha = 1.0$ and α_b . Here, α_b is the value of α corresponding to the least number of iterations. This 'optimal' value was found experimentally, with increments of 10^{-2} in the trial runs for $n = 2,500$ and of 10^{-1} for $n = 10,000$. The initial iterate $\mathbf{x}^{(0)}$ was taken to be the vector with all entries equal to $1/n$, though better choices are possible (notice that $\mathbf{x}^{(0)} \notin R(\mathbf{M}^{-1}\mathbf{A})$, as required). We found that normalizing the approximate solution after each iteration is superfluous, except that without normalization the norm of the residual tends to behave somewhat erratically in the first few steps. Since this had no consequence on the number of iterations required for convergence, we chose not to normalize our successive approximations until the end of the iterations.

Although Theorem 5.1 guarantees convergence of the arithmetic mean method only for $\alpha \in (0, 1)$, it turns out that for this particular class of problems the value of $\alpha = 1$ implies convergence. Indeed, the optimal value of α was always found to be greater than or equal to 1; see [2], p.234 for an explanation of this fact. Care should be taken in the selection of α since the arithmetic mean method appears to be quite sensitive with respect to small changes in this parameter. More precisely,

values of α slightly greater than the optimal α_b may result in a considerable increase in the number of iterations, or even in divergence. Usually, $\alpha = 1$ is quite close to α_b . On the other hand, the choice of ρ appears to be unimportant, and we found $\rho = 0$ to be satisfactory. In order to study the effect of asymmetry in $\mathbf{A}$ on

Table 5.1: Results for $n = 2,500$ and $n = 10,000$ ($\lambda_1 = 1.0$, $\lambda_2 = 2.0$)

n	μ_1	μ_2	AM ₁		AM ₂	
2,500	1.5	2.5	$\alpha = 1.0$	310	$\alpha = 1.0$	1,449
			$\alpha_b = 1.25$	246	$\alpha_b = 1.12$	1,293
2,500	5.0	10.0	$\alpha = 1.0$	213	$\alpha = 1.0$	1,013
			$\alpha_b = 1.23$	171	$\alpha_b = 1.10$	920
2,500	10.0	20.0	$\alpha = 1.0$	149	$\alpha = 1.0$	661
			$\alpha_b = 1.17$	122	$\alpha_b = 1.08$	612
2,500	50.0	100.0	$\alpha = 1.0$	73	$\alpha = 1.0$	237
			$\alpha_b = 1.08$	64	$\alpha_b = 1.01$	235
10,000	1.5	2.5	$\alpha = 1.0$	634	$\alpha = 1.0$	2,932
			$\alpha_b = 1.3$	506	$\alpha_b = 1.1$	2,665
10,000	5.0	10.0	$\alpha = 1.0$	431	$\alpha = 1.0$	2,038
			$\alpha_b = 1.2$	357	$\alpha_b = 1.1$	1,853
10,000	10.0	20.0	$\alpha = 1.0$	296	$\alpha = 1.0$	1,323
			$\alpha_b = 1.2$	244	$\alpha_b = 1.0$	1,323
10,000	50.0	100.0	$\alpha = 1.0$	152	$\alpha = 1.0$	461
			$\alpha_b = 1.1$	128	$\alpha_b = 1.0$	461

Table 5.2: Results for $n = 2,500$ ($\alpha = 1.0$)

λ_1	λ_2	μ_1	μ_2	AM ₁	AM ₂
0.001	0.1	1.0	0.1	53	476
0.001	0.1	0.01	1.0	119	★
0.001	0.001	1.0	0.001	96	436
0.01	0.001	0.1	1.0	132	194
0.01	0.001	1.0	1.0	60	106
0.01	0.001	1.0	0.01	772	1,031

(★ : failed to converge in less than 2,000 iterations)

the number of iterations, we have tried different values of the repair rates μ_1, μ_2 . As these parameters increase, the entries above the main diagonal of $\mathbf{A}$ become larger (in magnitude) with respect to those located below the main diagonal, which are held constant. As a consequence, both schemes require less iterations for convergence. The scheme that benefits more from this kind of asymmetry is AM_2 .

As expected, if the roles of λ_i, μ_i are interchanged, the behavior of AM_1 and AM_2 remains unaffected. The arithmetic mean method is sensitive to asymmetry in $\mathbf{A}$, but the number of iterations is the same whether the mass of the matrix is located above or below the main diagonal, as one would expect.

The results in Table 5.1 correspond to only one possible kind of asymmetry, where the parameters λ_i, μ_i were chosen in a rather special way. There are, however, many other possible situations. We ran a few experiments with $n = 2,500$ (and $\alpha = 1.0$) and various values of the parameters λ_i, μ_i ; some of the results are reported in Table 5.2.

We see that depending on the problem, the two schemes exhibit widely different behavior. In the first three cases AM_1 is substantially cheaper, both in terms of iterations and operation count, than AM_2 ; in the last three cases, however, AM_2 appears to be competitive with AM_1 in terms of number of arithmetic operations, although the number of iterations required for convergence is still higher.

Thus, while form AM_1 of the arithmetic mean method is to be preferred in general, there are problems where form AM_2 might be useful, especially if storage limitations arise.

To conclude this section, we remark that there are choices of λ_i, μ_i which lead to nearly completely decomposable chains, for which splitting methods by themselves are known to perform rather poorly. This is the case, for example, if λ_1 and μ_1 are very small compared to λ_2 and μ_2 ; IAD methods should be used in this case.

5.4 Conclusion

In this chapter we have extended the arithmetic mean method to the problem of computing the stationary probability distribution vector of an irreducible Markov chain. Two forms of this method are rich in parallel subtasks and are well-suited to solve problems with block tridiagonal structure, which arise frequently in the analysis of reliability models and queueing networks. We have compared these two forms of the arithmetic mean method and we have considered the effect of asymmetry in the coefficient matrix on the number of iterations. While the behavior of the iterative schemes can be quite different for different problems, in nearly all cases we found that the arithmetic mean method performs satisfactorily, and we concluded that parallel implementation of this scheme would result in a competitive Markov chain solver.

Future work should focus on extending the applicability of this method to matrices with a more complicated block structure, and to nearly completely decomposable problems. For instance, the arithmetic mean method might prove useful in combination with aggregation/disaggregation techniques, or as a preconditioner for Krylov subspace iterations (see [35]).

Chapter 6

State Space Exploration

Dynamic state exploration is an effective technique to explore states of sparse, large Markov chains having highly unbalanced [29] stationary probabilities and to compute related performance measures for them. In this technique, states of interest are explored starting from a particular state until a predetermined stopping criterion is met. In this chapter the dynamic state exploration idea is extended to the case where more than one starting state is considered simultaneously. This approach easily lends itself to parallel implementation and reduces the number of operations to be carried out for each starting state considerably. Furthermore, it is shown that this approach gives quite an accurate feeling for the nature of the states without having to solve the complete Markov chain. Experiments are carried out on a Markov chain arising from an Asynchronous Transfer Mode (ATM) traffic queue [52].

In Section 6.1, we describe how information from multiple starting states may be utilized to locate states of interest rapidly. In Section 6.2, we point out two different approaches that may be used to bound performance measures of interest. In Section 6.3, we describe the problem experimented with and discuss the results of these experiments. Finally, we draw some conclusions in Section 6.4.

6.1 State Exploration in Markov Chains

Most real-life systems modeled in the form of Markov chains yield stochastic matrices that have few outgoing transitions from each state. However, it is not correct to assume beforehand that only a few states will accumulate most of the probability mass. In fact, a system such as the multi-media traffic queue discussed in Section 6.3 may need many states to concentrate 90% of the probability mass. Therefore, exploration of states, most probable ones for instance, starting from a single state as suggested in [8], [41], and [42], may be time consuming, sometimes even not viable.

Since the aim of the exploration process is to locate states of interest as quickly as possible, one can start from different states and explore groups of states in the desired direction simultaneously. In this technique, it is quite probable (almost certain) that some explored states will occur in more than one set of explored states. Normally, one should explore a larger number of states from each starting state for a small number of starting states; whereas, if many starting states are considered, a sufficient number of steps from each one needs to be taken to locate those states of interest accurately. The exploration algorithm inspired by the work in [41] is given on the next page. Note that the same algorithm is executed for each starting state.

Define:

- s_s : starting state at the current iteration
- s_c : state chosen at the current iteration
- $\mathcal{E}^{(k)}$: set of explored states at iteration k
- $\mathcal{U}^{(k)}$: set of unexplored states at iteration k
- $\mathcal{T}$: neighboring states of s_c
- k : *cardinality* of $\mathcal{E}^{(k)}$ (i.e., number of elements in set $\mathcal{E}^{(k)}$)
- $u^{(k)}$: *cardinality* of $\mathcal{U}^{(k)}$
- $m^{(k)}$: sum of k and $u^{(k)}$ (introduced for notational convenience)

Figure 6.1: Dynamic state exploration algorithm

1. Pick up a starting state, s_s

$$k = 0; \quad s_c = s_s; \quad \mathcal{E}^{(0)} = \emptyset; \quad \mathcal{U}^{(0)} = \emptyset$$

2. $k = k + 1$

$$\mathcal{T} = \{s_j \mid \Pr(s_c \rightarrow s_j) \neq 0, \quad s_j \neq s_c\}$$

$$\mathcal{E}^{(k)} = \mathcal{E}^{(k-1)} \cup \{s_c\}$$

$$\mathcal{U}^{(k)} = (\mathcal{U}^{(k-1)} \cup \mathcal{T}) \setminus \mathcal{E}^{(k)}$$

$$u^{(k)} = \#(\mathcal{U}^{(k)})$$

$$m^{(k)} = k + u^{(k)}$$

3. For convenience, map $\mathcal{E}^{(k)}$ to $\{1, 2, \dots, k\}$ (make sure s_s is mapped to 1)

Also map $\mathcal{U}^{(k)}$ to $\{k+1, k+2, \dots, m^{(k)}\}$

Form the $(m^{(k)} + 1) \times (m^{(k)} + 1)$ stochastic matrix

$$\mathbf{P}^{(k)} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,k} & p_{1,k+1} & \cdots & p_{1,m^{(k)}} & p_{1,m^{(k)}+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{k,1} & \cdots & p_{k,k} & p_{k,k+1} & \cdots & p_{k,m^{(k)}} & p_{k,m^{(k)}+1} \\ p_{k+1,1} & \cdots & p_{k+1,k} & p_{k+1,k+1} & \cdots & p_{k+1,m^{(k)}} & p_{k+1,m^{(k)}+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{m^{(k)},1} & \cdots & p_{m^{(k)},k} & p_{m^{(k)},k+1} & \cdots & p_{m^{(k)},m^{(k)}} & p_{m^{(k)},m^{(k)}+1} \\ 1 & \cdots & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \quad (6.1)$$

where

$$p_{i,m^{(k)}+1} = 1 - \sum_{j=1}^{m^{(k)}} p_{i,j}, \quad 1 \leq i \leq m^{(k)} \quad (6.2)$$

4. Solve $\boldsymbol{\pi}^{(k)} \mathbf{P}^{(k)} = \boldsymbol{\pi}^{(k)}$ for $\boldsymbol{\pi}^{(k)}$ subject to $\|\boldsymbol{\pi}^{(k)}\|_1 = 1$

5. Stop if the stopping criterion is met

6. Choose a new state, s_c , according to some rule from $\mathcal{U}^{(k)}$

Update the starting state, s_s and reorder $\mathcal{E}^{(k)}, \mathcal{U}^{(k)}$ (optional)

Goto 2

Here is how the algorithm works for a single starting state. First a starting state, s_s , is picked up either randomly by the computer or by the user who may have knowledge about the nature of the states. Initially, the starting state is the only explored state and its neighbors (states within a single transition excluding itself) form the set of unexplored states. Then we form a matrix, $\mathbf{P}^{(k)}$, which has entries for the one-step transition probabilities among all the explored and unexplored states. Since this matrix is not stochastic, we add one more column that sums up the rows of the matrix to 1. Finally, we include one last row with all 0's except for a 1 at the column position corresponding to the starting state (see step 3 of the algorithm). Note that this matrix is different than the matrix considered in [41], firstly because, it is a stochastic matrix. Secondly, all transitions among the explored and unexplored states have to be known. On the other hand, there is no need to know the transitions from the unexplored states to states other than the explored and unexplored states. Therefore, all neighbors of the unexplored states need not be explored in this algorithm. The reason behind introducing a 1 to the position of the starting state in the last row is to restart the original process from the starting state whenever the process makes a transition to one of the unknown states (states yet have to be discovered and about which nothing is known). In essence, we aggregate all unknown states to a single state and place this unknown state at the end of the matrix in each dimension. The stopping criterion needed in step 5 may either be a fixed number of states to be explored or more naturally, a threshold imposed on the mean time between two visits to unknown states (MTTU). Clearly, MTTU can be computed as the sum of the mean number of visits to explored states between two visits to the aggregated unknown state. As shown in [8], [41], and [42], an approximation to the amount of time spent in the set of explored states at iteration k , in our case, may be given by

$$\text{MTTU}^{(k)} = \frac{\sum_{i=1}^k \pi_i^{(k)}}{\pi_{m^{(k)}+1}^{(k)}}, \quad (6.3)$$

where the stationary vector elements are available at the end of step 4. However, rather than imposing a threshold on $\text{MTTU}^{(k)}$, we choose to use $\sum_{i=1}^k \pi_i^{(k)}$ as the stopping criterion. The reason for this choice is that, there is a probability, namely $\sum_{i=k+1}^m \pi_i^{(k)}$, which is not accounted for in $\text{MTTU}^{(k)}$. Secondly, we do not know how large a value is to be picked up as a threshold for MTTU at the outset. Clearly, a threshold value greater than or equal to $\sum_{i=1}^k \pi_i^{(k)}$ implies a value that is greater than or equal to $(\sum_{i=1}^k \pi_i^{(k)}) / (1 - \sum_{i=1}^k \pi_i^{(k)})$ for $\text{MTTU}^{(k)}$. However, the converse statement is not true. A state, s_c , is chosen from the set of unexplored states if the stopping criterion has not been met. If the most probable states are of interest, at step 6, we choose $s_c = s_j$ so that

$$\pi_j^{(k)} = \max_{k+1 \leq i \leq m^{(k)}} \pi_i^{(k)}. \quad (6.4)$$

If least probable states are to be explored, we replace $\max$ by $\min$. Thereafter, the chosen state becomes one of the explored states. It is removed from the set of unexplored states and we continue with the next iteration. At each iteration, neighbors of the recently chosen state that do not exist in the sets of explored and unexplored states also join the set of unexplored states. Clearly, if all states of the Markov model were to be explored, we would have the exact solution vector at step 4 and $\sum_{i=1}^k \pi_i^{(k)} = 1$ (i.e., MTTU would be infinitely large). Hence, the algorithm always terminates with a properly chosen threshold.

The mapping of states to integers as described in step 3 is introduced to enhance clarity and the understanding of the algorithm. From an implementation point of view, this mapping is unnecessary. With proper indexing, there is essentially no need to reconstruct the arrays storing the sets of explored and unexplored states, and for

that matter, the stochastic matrix right from the beginning at each iteration of the algorithm. One can continue from where it was left off at the last iteration and simply build on to the existing data structures. Moreover, the algorithm can also be applied to the continuous-time Markov chain if the matrix in step 3 is replaced accordingly. The fact that the transition rate matrix has negative diagonal entries is irrelevant, since transitions from a state onto itself are not considered in the search process. In this case, the second statement in step 2 should be rewritten as

$$\mathcal{T} = \{s_j \mid \text{Rate}(s_c \rightarrow s_j) \neq 0, s_j \neq s_c\} \quad (6.5)$$

One improvement we find useful is to sort the solution vector obtained in step 4 of the algorithm and to reform the sets of explored and unexplored states accordingly. If most probable states are sought, then set $s_s = s_l$ so that

$$\pi_l^{(k)} = \max_{1 \leq i \leq m^{(k)}} \pi_i^{(k)} \quad (6.6)$$

and

$$\begin{aligned} \mathcal{E}^{(k)} &= \{\text{the } k \text{ states with the highest probabilities in } \boldsymbol{\pi}^{(k)}\} \\ \mathcal{U}^{(k)} &= \{\text{the neighbors of the states in } \mathcal{E}^{(k)}\} \end{aligned}$$

at step 6 of the algorithm. If least probable states are to be explored, pick up the lowest probabilities when forming $\mathcal{E}^{(k)}$. Note that this is a costly operation, and it basically requires the matrix at step 3 to be reorganized. The reason for this reorganization is that, now an unexplored state may have become an explored state, and we need to include its neighbors in the set of unexplored states as well. Similarly, an explored state may become an unexplored state, and all or some of its neighbors may need to be removed from the set of unexplored states. Even itself may have to be removed if its not a neighbor of any explored state. However, this operation need not be performed at each iteration, but just frequently enough, so that misjudgements

made earlier may be corrected. Otherwise, once a state becomes an explored state, it will remain as one until the algorithm terminates. An alternative is to change only the starting state to the state with the highest (lowest) probability in the set of explored states. In this case, a reorganization as the one discussed will not be necessary.

If we forget about this last improvement that requires reorganization of the explored and unexplored states, then, with a careful implementation, each iteration may build on to the preceeding one. Since we do not expect to carry out very many iterations, we suggest Gaussian elimination as the solution method applied to the system

$$(\mathbf{I} - \mathbf{P}^{(k)})^T (\boldsymbol{\pi}^{(k)})^T = \mathbf{o} \quad (6.7)$$

subject to the same constraint in step 4. Back substitution is the only solution phase required since the coefficient matrix is singular and the right-hand side is 0. Even forward elimination may resume from where it left off at the last iteration. The justification for this choice of the solution method is that, we plan to consider only sufficiently many starting states so that each matrix generated from a starting state is not larger than the size for which the computer produces solutions quickly. Additionally, a bound on the maximum number of operations to be performed may be estimated beforehand as opposed to the case with matrix iterative methods. One important observation regarding the solution procedure is that the last column in $\mathbf{P}^{(k)}$, which sums up the rows to 1, has no significance. Once the transposed system is formed, this last column becomes the last row of the transposed system, and it will be all 0's after the system is reduced to upper-triangular form (since we have a singular system of linear equations).

Just as in [41], let us assume that on the average each state has r output transitions (excluding itself) and a fraction p of those end in unexplored states. With

the given assumption, $\mathcal{U}^{(k-1)} = pr(k-1)$ and $\mathcal{U}^{(k)} = prk$. Therefore, the number of extra rows in $\mathbf{P}^{(k)}$ is $pr + 1$. An upper bound on the number of multiplications required to carry out the reductions in between columns $((k-1) + pr(k-1) + 1)$ and $(k + prk + 1)$ for rows 1 through $((k-1) + pr(k-1))$ and to reduce the extra $pr + 1$ rows to upper-triangular form is given by

$$\sum_{j=1}^{(k-1)+pr(k-1)} \sum_{i=1}^{j-1} (pr+1) + \sum_{j=(k-1)+pr(k-1)+1}^{k+prk+1} \sum_{i=1}^{j-1} (k+prk+1-i+1) = O(k^2(pr+1)^3).$$

For the single back substitution phase, we have $O((k+prk+1)^2) = O(k^2(pr+1)^2)$ multiplications. Hence, at iteration k the total number of multiplications performed to solve the system in step 4 is $O(k^2(pr+1)^3)$. The total number of operations over N iterations of the algorithm is $O(N^2(pr+1)^3)$. Since we carry out the solution procedure in temporary storage and choose not to overwrite $\mathbf{P}^{(k)}$, the storage requirements for the algorithm over N iterations is $(N + prN + 1)^2 + (N + prN)r$. The total storage needed is then $O(N^2(pr+1)^2)$.

Let us return to the idea of starting the exploration process from multiple states. There are two alternatives that may be followed. The first approach is to start from a few states (we choose a single starting state) and to explore for a considerable amount of time (i.e., a threshold probability of 0.9999 in the set of explored states, for instance) from each one. The other approach that may be taken is to start from a larger number of states and to explore fewer transitions out of each starting state (i.e., a threshold probability of 0.99 in the set of explored states, for instance). Remember that, our ultimate goal is to explore sufficiently many states so that we can obtain good bounds on performance measures dependent upon these states. In the next section, we discuss how multiple sets of explored states may be combined to achieve bounds on performance measures.

6.2 Bounding Performance Measures

Let us first briefly mention a result for NCD Markov chains. It is well-known that (see [6], [40]) the conditional stationary distribution of a NCD block may be tightly bounded. Furthermore, the marginal distribution, which removes this conditioning, may be also bounded by an aggregation step following the computation of the conditional stationary vector of each NCD block. Consequently, it is possible to compute the best known bounds for the stationary distribution of a NCD block, and if the states of interest are the states of a NCD block, this technique gives the best achievable bounds for the stationary probabilities of the corresponding block of states. Interestingly, the same technique gives good bounds for the conditional stationary distribution of the most probable states, when the states explored as such through dynamic state exploration concentrate more than, say, 99% of the probability mass. As the percentage increases, the bounds get tighter.

Another technique is to replicate the set of explored states other than the starting state and to classify these replicated states according to the minimum number of steps taken from the starting state to reach each one. After the classification of replicated states into disjoint subsets is performed, each subset is aggregated (the way in which aggregation is done differs depending on whether a lower or an upper bound for the conditional stationary vector is sought) and an equivalent system derived. By this technique, looser but yet computationally efficient lower and upper bounds on the stationary distribution of most probable states may be obtained. See [25], [41], and [42] for details.

We remark that either of the two techniques may be employed when the sought performance measure is dependent on the most probable states. So far, we have not come up with a scheme which would help bound measures depending on the least probable states (such as the blocking probability in a finite buffer) tightly. After

careful consideration of the two techniques described, we think the first scheme may be implemented in such a way that, it has, if not less, comparable computational efficiency to the second technique for a reasonable number of explored states (i.e., in the order of hundreds). Moreover, the bounds computed in this way will be definitely tighter than the bounds obtained through the approximation in the second technique since both schemes consider the same number of explored states. Now, we are in a position to show how bounds on the conditional stationary distribution of a set of states $\mathcal{S}' \subset \mathcal{S}$, where $\#(\mathcal{S}') = n' > 0$, may be computed (see [6], p.810). Please refer to the algorithm in Figure 6.2 on the next page.

We would like to emphasize that the lower and upper bounds obtained by the described technique are the tightest bounds computable when only partial information (i.e., $\mathbf{P}'$) is available and nothing is known about transitions from states in $\mathcal{S} \setminus \mathcal{S}'$ to states in $\mathcal{S}'$ (i.e., $\mathbf{x}$). If one is to claim that there may be bounds tighter than the ones supplied in (6.10) and (6.11), it is always possible to play the role of an adversary and to choose a distribution for $\mathbf{x}$ that makes the opponent's guess fail.

For our purposes, bounds on the conditional stationary vector of the set of explored states may be obtained by extracting the one-step transition probability matrix of the final set of explored states from $\mathbf{P}^{(k)}$ (step 1). If multiple starting states are considered, then the final set of explored states is the union of the sets of explored states that come out of different starting states. In this case, the one-step transition probability matrix which needs to be formed from scratch has as many states as there are in the union. Since the one-step transition probability matrix is substochastic in both cases (given that all states of the Markov chain have not been explored), we augment it with a column and add one row (step 2). The extra column sums up the rows to 1. In order to compute the desired lower and upper bounds, we have to solve as many systems of linear equations as the number of states in the

Figure 6.2: Bounding the conditional stationary distribution of a subset of states

1. As always, let $\mathbf{P}$ be the stochastic transition probability matrix of the Markov chain defined on the state space $\mathcal{S}$. From $\mathbf{P}$, extract the one-step transition probability matrix $\mathbf{P}'$ (which is not stochastic) confined to the states in $\mathcal{S}'$.
2. Augment $\mathbf{P}'$ with one column and add one row to give the $(n' + 1) \times (n' + 1)$ stochastic matrix

$$\mathbf{W}' = \begin{pmatrix} \mathbf{P}' & \mathbf{e} - \mathbf{P}'\mathbf{e} \\ \mathbf{x}^T & 0 \end{pmatrix}, \quad (6.8)$$

where $\|\mathbf{x}\|_1 = 1$.

3. For $i = 1, 2, \dots, n'$:
 - (a) Substitute 1 to column position i in the last row of $\mathbf{W}'$; that is, $x_i = 1$.
 - (b) Solve $\mathbf{z}_i \mathbf{W}' = \mathbf{z}_i$, $\|\mathbf{z}_i\|_1 = 1$.
 - (c) Normalize the first n' components of $\mathbf{z}_i$ such that $\sum_{j=1}^{n'} z_{i,j} = 1$.
 - (d) Now let $\mathbf{z}_i$ be the normalized row vector of length n' .
4. Form the stochastic matrix

$$\mathbf{Z} = \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_{n'} \end{pmatrix}. \quad (6.9)$$

5. If $\mathbf{v}$ is the conditional stationary distribution vector of $\mathcal{S}'$, then

$$v_j^{inf} = \max\{\min_i(z_{i,j}); 1 - \sum_{k \neq j} \max_i(z_{i,k})\}, \quad (6.10)$$

$$v_j^{sup} = \min\{\max_i(z_{i,j}); 1 - \sum_{k \neq j} \min_i(z_{i,k})\}, \quad (6.11)$$

where $j = 1, 2, \dots, n'$.

final set of explored states (step 3–4). Each system is formed by placing a 1 to a different column position in the extra row except for the extra state introduced. The equalities (6.10) and (6.11) are direct consequences of the restricted convex combination and of the normalization of the solution vector (step 5). Note that, a single LU decomposition, and as many backsubstitutions as there are states in the final set of explored states, needs to be performed. In Section 6.3, we provide results of some experiments conducted on a large and sparse Markov chain.

6.3 Numerical Experiments

The real-life application selected is from the communications field. A multi-media traffic queue on an ATM link is modelled in the form of a multi-dimensional Markov queueing process, where each individual arrival stream is treated as an independent Markov-modulated process. For details, see [52]. In the same reference it is indicated that for such a system the overall arrival process becomes a Markov-modulated Poisson process and the queueing model may be described in the form of a continuous Markov chain with states

$$\{(q(t), x(t)) \mid 0 \leq q(t) \leq K - 1; x(t) \in X(t)\}. \quad (6.12)$$

Here $q(t)$ represents the queue length at time t ($K - 1$ is the maximum queue size including the one in the server) and $x(t)$ is the arrival phase under the given buffer occupancy (it is in fact the state of the superimposed Markov-modulated traffic stream at time t). Hence, we have a Quasi-Birth-Death (QBD) process where each level corresponds to a different buffer occupancy and each state on each level corresponds to an arrival phase. Without overload control, $q(t)$ and $x(t)$ are independent. Therefore, the phase transitions at each level are identical except for the boundary levels, 0 and $K - 1$. The transition rate matrix is then given by

$$\mathbf{Q} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{U} & & & \\ \mathbf{D} & \mathbf{A} & \mathbf{U} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{D} & \mathbf{A} & \mathbf{U} \\ & & & \mathbf{D} & \mathbf{A}_{K-1} \end{pmatrix} \quad (6.13)$$

$\mathbf{Q}$ is block tridiagonal. The matrices $\mathbf{A}_0$, $\mathbf{A}$, and $\mathbf{A}_{K-1}$ specify changes in the arrival phase whereas $\mathbf{U}$ and $\mathbf{D}$, which are diagonal matrices, describe changes in buffer occupancy. The number of states at each level is given by $N = \prod_{i=1}^M N_i$, where M is the number of independent traffic streams and N_i is the number of states in the Markov chain that corresponds to the i^{th} traffic stream. Consequently, $\mathbf{Q}$ is of size $NK \times NK$.

Before we specify the system parameters chosen, we will pause and discuss the applicability of dynamic state exploration in real-life problems. We especially would like to stress that the following is *not* an exercise in computing performance measures for a given problem by the best means available; but, sometimes due to the size of the problem under consideration, dynamic state exploration may be the only viable technique for obtaining a solution. For the ATM traffic queue, dynamic state exploration gives satisfying results only for low to very low loads, because it is those cases in which the probability mass is accumulated in a reasonably small number of states. Therefore, the technique is *not* recommended for solving such a problem in the most general case. Our aim is to direct the attention to the feasibility of dynamic state exploration as a tool that may be useful when a specific criteria is met.

For the statistical multiplexer that concentrates traffic from voice and video sources (i.e., $M = 2$), $K = 512$, $N_1 = 30$, and $N_2 = 20$ ([52]) giving rise to a $307,200 \times 307,200$ system. We consider a smaller, yet instructive, example for which K , N_1 , N_2 have the values 120, 30, 20 respectively. The linear system to be solved is $72,000 \times 72,000$ in this case, and the percentage of nonzero elements

is $490,681/72,000^2 = 0.94652 \times 10^{-4}$. The number of outgoing transitions from a state (excluding itself) is approximately 6 (i.e., $r \approx 6$). For the system under consideration, the probability mass is distributed across many states when the load is larger than 10%, and a single search from a given initial state does not provide much information unless the search extends over a considerable number of states. However, when the load is less than 0.1, most of the probability mass is concentrated in a smaller number of states. We also noticed that together with the load on the system, the number of states that must be explored for a given probability mass changes for different values of K , $N1$, and $N2$. We experimented with four different problems. These problems are derived from the ATM model using loads of 0.1, 0.01, 0.001, 0.0001 respectively. It was quite interesting to find out that the stochastic transition probability matrix in each case was nearly completely decomposable. In Tables 6.1–6.4, the decomposability parameter, γ , is on the order of the degree of coupling that corresponds to the one-step transition probability matrix for the given size and number of aggregates, ngr .

Table 6.1: Sizes and number of aggregates: load 0.1

γ	ngr	size of aggregates
10^{-5}	4	(120 – 66,120)
10^{-4}	539	(1 – 1,200)
10^{-3}	838	(1 – 120)
10^{-2}	2,623	(1 – 120)

Table 6.2: Sizes and number of aggregates: load 0.01

γ	ngr	size of aggregates
10^{-6}	4	(120 – 66,120)
10^{-5}	442	(1 – 3,360)
10^{-4}	838	(1 – 120)
10^{-3}	2,266	(1 – 120)
10^{-2}	26,780	(1 – 120)

Table 6.3: Sizes and number of aggregates: load 0.001

γ	ngr	<i>size of aggregates</i>
10^{-7}	2	(3,600 – 68,400)
10^{-6}	442	(1 – 3,600)
10^{-5}	838	(1 – 120)
10^{-4}	2,266	(1 – 120)
10^{-3}	25,947	(1 – 120)
10^{-2}	72,000	(1 – 1)

Table 6.4: Sizes and number of aggregates: load 0.0001

γ	ngr	<i>size of aggregates</i>
10^{-8}	2	(3,600 – 68,400)
10^{-7}	423	(1 – 3,600)
10^{-6}	838	(1 – 120)
10^{-5}	2,266	(1 – 120)
10^{-4}	25,828	(1 – 120)
10^{-3}	72,000	(1 – 1)
10^{-2}	72,000	(1 – 1)

We filled out Table 6.5 on the next page by computing the stationary probability vector of each one-step transition probability matrix using the iterative aggregation-disaggregation (IAD) algorithm with Gaussian elimination for the solution of the aggregates and the coupling matrix (see [48]). The marginal probabilities are given by

$$\Pi(i) = \sum_{s_j \in \text{block } i} \pi_j \quad \text{for } i = 0, 1, \dots, K-1, \quad (6.14)$$

where π is the stationary vector of the corresponding matrix.

As it can be seen in Table 6.6 on the next page, the number of states required to accumulate a given probability mass decreases with decreasing load for the system.

Table 6.5: Marginal probability distribution: loads 0.1, 0.01, 0.001, 0.0001

#cells	0.1	0.01	0.001	0.0001
0	0.90485×10^0	0.99048×10^0	0.99905×10^0	0.99990×10^0
1	0.85317×10^{-1}	0.94169×10^{-2}	0.95054×10^{-3}	0.95142×10^{-4}
2	0.87473×10^{-2}	0.97265×10^{-4}	0.98245×10^{-6}	0.98342×10^{-8}
3	0.96049×10^{-3}	0.10753×10^{-5}	0.10868×10^{-8}	0.10879×10^{-11}
4	0.11180×10^{-3}	0.12595×10^{-7}	0.12737×10^{-11}	0.2×10^{-15}
5	0.13693×10^{-4}	0.15518×10^{-9}	0.16×10^{-14}	
6	0.17545×10^{-5}	0.19994×10^{-11}		
7	0.23410×10^{-6}	0.268×10^{-13}		
8	0.32405×10^{-7}	0.4×10^{-15}		
9	0.46387×10^{-8}			
10	0.68488×10^{-9}			
11	0.10405×10^{-9}			
12	0.16234×10^{-10}			
13	0.25965×10^{-11}			
14	0.4251×10^{-12}			
15	0.712×10^{-13}			
16	0.122×10^{-13}			
17	0.22×10^{-14}			
18	0.4×10^{-15}			
19	0.1×10^{-15}			

Table 6.6: Number of states that accumulate a given probability mass: loads 0.1, 0.01, 0.001, 0.0001

$Pr \backslash Load$	0.1	0.01	0.001	0.0001
0.9	122	82	79	79
0.95	176	107	102	101
0.99	331	195	155	152
0.999	615	342	262	224
0.9999	964	524	395	319
0.99999	1,380	720	521	445

The average number of cells in the buffer at steady state is given by

$$\bar{w} = \sum_{i=0}^{K-1} i\Pi(i). \quad (6.15)$$

Table 6.7: Average number of cells in the buffer at steady state: loads 0.1, 0.01, 0.001, 0.0001

<i>Load</i>	<i>avg. # cells</i>
0.1	0.10622×10^0
0.01	0.96147×10^{-2}
0.001	0.95251×10^{-3}
0.0001	0.95162×10^{-4}

Note that the marginal probability distribution of having 0 cells in the buffer, which has a large portion of the probability mass for the chosen loads as one would expect, does not contribute to the average number of customers in the buffer. Hence, the values in Table 6.7 are due to marginal probabilities of having 1 or more cells in the buffer. In all problems, the probability of having 20 or more cells in the buffer turned out to be 0.

We carried out two types of experiments. In the first group of experiments, we considered a single starting state. Since it is not quite clear which state is the most probable one, we picked up state 1 as the starting state. This state lies in the first block, where there are no cells in the buffer. In the second group of experiments, we considered 10 starting states. We choose states 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400; they are uniformly distributed over the first four blocks in the matrix. We choose not to sort the solution vector obtained in step 4 of the exploration algorithm in Section 6.1. We present the results in five decimal digits precision. Let us represent by Π_e and $\bar{w}_e$, the marginal probability and the average number of cells in the buffer at steady state conditioned on the system being in the set of explored states.

1. Load = 0.1

Single starting state : 1

Threshold probability mass in explored states : 0.999

$k = 126$, $u^{(k)} = 55$

$$\begin{aligned}
0.44356 \times 10^{-6} &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\
0.10739 \times 10^{-6} &\leq \Pi_e(1) \leq 0.10000 \times 10^1 \\
0.24862 \times 10^{-7} &\leq \Pi_e(2) \leq 0.87076 \times 10^0 \\
0.47159 \times 10^{-8} &\leq \Pi_e(3) \leq 0.16769 \times 10^0 \\
0.71047 \times 10^{-9} &\leq \Pi_e(4) \leq 0.29289 \times 10^{-1} \\
0.10185 \times 10^{-11} &\leq \Pi_e(5) \leq 0.53274 \times 10^{-2} \\
0.00000 \times 10^0 &\leq \Pi_e(6) \leq 0.45214 \times 10^{-3} \\
0.17411 \times 10^{-6} &\leq \bar{w}_e \leq 0.33911 \times 10^1
\end{aligned}$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400

Threshold probability mass imposed on each starting state : 0.999

$k = 303$

$$\begin{aligned}
0.19859 \times 10^{-6} &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\
0.79026 \times 10^{-8} &\leq \Pi_e(1) \leq 0.10000 \times 10^1 \\
0.21722 \times 10^{-9} &\leq \Pi_e(2) \leq 0.29416 \times 10^{-1} \\
0.80788 \times 10^{-15} &\leq \Pi_e(3) \leq 0.42917 \times 10^{-3} \\
0.83370 \times 10^{-8} &\leq \bar{w}_e \leq 0.10601 \times 10^1
\end{aligned}$$

2. Load = 0.01

Single starting state : 1

Threshold probability mass in explored states : 0.9999

$k = 550, u^{(k)} = 299$

$$\begin{aligned} 0.86507 \times 10^0 &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\ 0.11184 \times 10^{-6} &\leq \Pi_e(1) \leq 0.13493 \times 10^0 \\ 0.11655 \times 10^{-8} &\leq \Pi_e(2) \leq 0.85478 \times 10^{-2} \\ 0.11417 \times 10^{-6} &\leq \bar{w}_e \leq 0.15203 \times 10^0 \end{aligned}$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400

Threshold probability mass imposed on each starting state : 0.9995

$k = 267$

$$\begin{aligned} 0.80007 \times 10^0 &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\ 0.21875 \times 10^{-6} &\leq \Pi_e(1) \leq 0.19983 \times 10^0 \\ 0.14153 \times 10^{-13} &\leq \Pi_e(2) \leq 0.63946 \times 10^{-4} \\ 0.12835 \times 10^{-15} &\leq \Pi_e(3) \leq 0.34401 \times 10^{-4} \\ 0.21875 \times 10^{-6} &\leq \bar{w}_e \leq 0.20006 \times 10^0 \end{aligned}$$

3. Load = 0.001

Single starting state : 1

Threshold probability mass in explored states : 0.9999

$k = 341, u^{(k)} = 244$

$$0.90345 \times 10^0 \leq \Pi_e(0) \leq 0.10000 \times 10^1$$

$$0.22791 \times 10^{-9} \leq \Pi_e(1) \leq 0.96552 \times 10^{-1}$$

$$0.22791 \times 10^{-9} \leq \bar{w}_e \leq 0.96552 \times 10^{-1}$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400

Threshold probability mass imposed on each starting state : 0.9995

$k = 170$

$$0.99567 \times 10^0 \leq \Pi_e(0) \leq 0.10000 \times 10^1$$

$$0.00000 \times 10^0 \leq \Pi_e(1) \leq 0.42981 \times 10^{-2}$$

$$0.00000 \times 10^0 \leq \Pi_e(2) \leq 0.23211 \times 10^{-4}$$

$$0.00000 \times 10^0 \leq \Pi_e(3) \leq 0.13660 \times 10^{-4}$$

$$0.00000 \times 10^0 \leq \bar{w}_e \leq 0.43855 \times 10^{-2}$$

4. Load = 0.0001

Single starting state : 1

Threshold probability mass in explored states : 0.9999

$k = 294$, $u^{(k)} = 283$

$$\begin{aligned} 0.99665 \times 10^0 &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\ 0.00000 \times 10^0 &\leq \Pi_e(1) \leq 0.33497 \times 10^{-2} \\ 0.00000 \times 10^0 &\leq \bar{w}_e \leq 0.33497 \times 10^{-2} \end{aligned}$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400

Threshold probability mass imposed on each starting state : 0.9997

$k = 155$

$$\begin{aligned} 0.99995 \times 10^0 &\leq \Pi_e(0) \leq 0.10000 \times 10^1 \\ 0.00000 \times 10^0 &\leq \Pi_e(1) \leq 0.46305 \times 10^{-4} \\ 0.00000 \times 10^0 &\leq \Pi_e(2) \leq 0.23167 \times 10^{-5} \\ 0.00000 \times 10^0 &\leq \Pi_e(3) \leq 0.13672 \times 10^{-5} \\ 0.00000 \times 10^0 &\leq \bar{w}_e \leq 0.55040 \times 10^{-4} \end{aligned}$$

The probability mass accumulated in the set of explored states (conditioned on the event that the process restarts from the starting state whenever it reaches the unknown states) increases over iterations, although at times it may show an erratic behavior and decrease from one iteration to another. An intuitive explanation is that the decision (i.e., choice of a new explored state) made at a specific iteration may very well be suboptimal, since the decisions are based on partial information about the system. With the inclusion of new states into the sets of explored and

unexplored states, new transitions are uncovered and a better decision is likely to be made at the next iteration. In other words, the value of the probability mass in the explored states for the current iteration may be less than the mass in an earlier iteration; however, if sufficiently many iterations are performed, the probability mass in the set of explored states will exceed the prespecified threshold. Although the probability mass is conditional, the probability of being in the set of explored states increases with the number of states in the set. If all states in the Markov chain were explored, then the probability mass would be unconditional. In order to find out a lower bound on the exact percentage of the conditional probability mass accumulated in the set of explored states when the search terminates, one has to consider each state in the set of explored states as a starting state and take the minimum among the probabilities computed. Hence, there are two reasons behind the termination of the search process with a smaller number of explored states than it is required to accumulate the unconditional probability mass. The first one is the fact that the probability mass accumulated in the set of explored states is conditional, and the second one is the fact that even this conditional probability mass may not be optimal due to the starting state chosen. For instance, even though 615 states are required to accumulate a probability mass of 0.999 for the ATM traffic queue with load 0.1 (see Table 6.6), only 126 states were explored when the search started from state 1. In other words, the unconditional probability mass accumulated in these 126 explored states is much less than 0.999. So the key point in the algorithm is to accumulate as much probability mass as possible in the states to be explored.

When multiple starting states are considered, it is possible to enforce the exploration of states in other parts of the Markov chain by choosing at least some of the starting states in different parts of the chain. In this way, the likelihood of skipping a state that should be included in the set of explored states is reduced. Additionally,

the number of states in the union of the sets of explored states is at least as many as the number of states in the smallest of these sets. Hence, multiple starting states most likely give better bounds than a single starting state for the same threshold probability mass.

For smaller load values, we employed a larger threshold probability mass to be accumulated in the set of explored states. In each problem, a smaller threshold probability mass for multiple starting states gave almost as good bounds as a larger threshold did for a single starting state. We observed that the dynamic state exploration technique is most effective when a large portion of the probability mass is concentrated in a small number of states as in the ATM model with the the lightest load.

6.4 Conclusion

In this chapter, we have shown how dynamic state exploration may be used to bound certain performance measures in systems with highly unbalanced stationary probabilities. The performance measures that may be bounded in this way are those that depend on the most probable states. However, the idea may also be used to locate states with smaller probabilities, and therefore help one acquire an understanding of the nature of the states in the chain.

The technique amounts to exploring the most probable states from a starting state until a stopping criterion is met. Once the total probability of being in the explored states exceeds a preset threshold, the search ends. An improvement over the original algorithm is to sort the solution vector obtained at a given step of the exploration process, and reform the set of explored states before starting with the next step. However, this is a costly operation and should not be performed frequently. We also stress that the exploration process can be further speeded up by starting the

search from multiple states and exploring for a smaller threshold probability mass from each one.

We have reported results on an ATM traffic queue having 72,000 states. The results indicate that a direct solution method such as Gaussian elimination may be employed effectively at each step of the exploration process for a small number of explored states. However, if the probability mass to be accumulated, extends over, say, more than 500 states, direct methods tends to be slow, and one should use matrix iterative methods in that case. We think dynamic state exploration is most effective when the number of states accumulating a large portion of the probability mass, 0.9999 for instance, is in the order of 100–200's. Since the ATM model considered is NCD, future work may focus on quasi-lumpability as an alternate technique for obtaining bounds on performance measures of sparse, large NCD Markov chains.

Chapter 7

Quasi-lumpability in NCD Chains

Let us start this chapter by asking ourselves the following questions. “Is it possible to obtain bounds on the stationary probability of being in each NCD block of a NCD Markov chain in an inexpensive way?” “Furthermore, if the answer to the preceeding question is yes, can one improve these bounds by exploiting the structure and symmetries of the chain?” The motivation behind seeking answers to such questions is that, in many cases performance measures of interest depend on the probability of being in certain groups of states. That is, probabilities need to be computed at a coarser level; each and every stationary probability is not needed. If the problem at hand is one in which the stationary probabilities of interest are those of the coupling matrix corresponding to the underlying NCD Markov chain, then the technique that we discuss in this chapter may be used to obtain answers to the above questions. Whereas, if all stationary probabilities of the NCD Markov chain are to be computed, IAD should be the solution method of choice.

In this chapter, we show that NCD Markov chains are quasi-lumpable. The state space partition is the natural one, and the technique may be used to compute bounds on the probability of being in each NCD block. The procedure amounts to solving linear systems of order equal to the number of NCD blocks in the chain. In Section 7.1, we provide the definition of quasi-lumpability coined in [10], and in Section 7.2,

we show how quasi-lumpability applies to NCD Markov chains (Subsection 1.2.2). The definition of lumpability is given in Subsection 1.2.3. The effects of quasi-lumpability on the 8×8 Courtois' matrix are illustrated in Section 7.3.

7.1 Definition

A DTMC is said to be ϵ -quasi-lumpable³ with respect to a given state space partition $\mathcal{S} = \bigcup_i \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $\forall i, j$ if its transition probability matrix $\mathbf{P}$ can be written as $\mathbf{P} = \mathbf{P}^- + \mathbf{P}^\epsilon$, where $\mathbf{P}^-$ is the largest lower bound (componentwise) for $\mathbf{P}$ that satisfies the lumpability condition

$$\forall \mathcal{S}_i, \mathcal{S}_j \subset \mathcal{S}, \forall s \in \mathcal{S}_i : \sum_{s' \in \mathcal{S}_j} p_{s,s'}^- = k_{i,j} \quad \text{for } i \neq j, \quad (7.1)$$

where $k_{i,j}$ is a constant value that depends only on i and j , $p_{s,s'}^-$ is the one-step transition probability of going from state s to state s' in the matrix $\mathbf{P}^-$, and no element in $\mathbf{P}^\epsilon$ is greater than ϵ in value. A similar definition applies to a CTMC, where the probability matrix $\mathbf{P}$ is substituted with the infinitesimal generator $\mathbf{Q}$ (see [10], p.2). In the next section we give a construction proof for the ϵ -quasi-lumpability of NCD Markov chains.

7.2 Construction

1. For a NCD Markov chain,

$$\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_N\},$$

where $\mathcal{S}_i$ is the set of states forming the i^{th} aggregate and $\#(\mathcal{S}_i) = n_i$.

³ ϵ should not be confused with machine epsilon defined in Section 3.2.

Form the matrix given by

$$\mathbf{P}^- = \begin{pmatrix} \mathbf{P}_{1,1} & \mathbf{P}_{1,2}^- & \cdots & \mathbf{P}_{1,N}^- \\ \mathbf{P}_{2,1}^- & \mathbf{P}_{2,2}^- & \cdots & \mathbf{P}_{2,N}^- \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N,1}^- & \mathbf{P}_{N,2}^- & \cdots & \mathbf{P}_{N,N}^- \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{matrix} \quad (7.2)$$

where

$$\mathbf{P}_{i,j}^- = \begin{cases} \mathbf{P}_{i,j} & \text{if } \mathbf{P}_{i,j}\mathbf{e} = k_{i,j}\mathbf{e} \\ \mathbf{P}_{i,j} - \mathbf{P}_{i,j}^\epsilon & \text{otherwise} \end{cases} \quad \text{for } i \neq j \quad (7.3)$$

Diagonal blocks of $\mathbf{P}^-$ are the same as those of $\mathbf{P}$. When $\mathbf{P}_{i,j}\mathbf{e} \neq k_{i,j}\mathbf{e}$, $\mathbf{P}_{i,j}^\epsilon$ is chosen so that $(\mathbf{P}_{i,j} - \mathbf{P}_{i,j}^\epsilon)\mathbf{e} = k_{i,j}\mathbf{e}$. Here, $k_{i,j} = \min(\mathbf{P}_{i,j}\mathbf{e})$. At this point, we stress that $\mathbf{P}^\epsilon$ may not be unique, and leave the discussion on how to make a better choice among the alternatives available to after the construction. Furthermore, $\mathbf{P}^\epsilon$ has nonzero blocks (in which there is at least one nonzero element) in locations corresponding to the nonzero blocks of $\mathbf{P}$ which do not have equal row sums. On the other hand, the number of zero blocks in $\mathbf{P}^-$ may be more than the number of zero blocks in $\mathbf{P}$. In other words, there may be nonzero blocks in $\mathbf{P}$ for which $k_{i,j} = 0$, implying $\mathbf{P}_{i,j}^- = 0$. Note that, if $\mathbf{P}^\epsilon$ is the null matrix, then $\mathbf{P}$ will be exactly lumpable, and the remaining steps in the construction should be skipped.

2. Once $\mathbf{P}$ is written as the sum of $\mathbf{P}^-$ and $\mathbf{P}^\epsilon$, we form another matrix,

$$\mathbf{P}^s = \begin{pmatrix} \mathbf{P}^- & \mathbf{y} \\ \mathbf{x}^T & 0 \end{pmatrix}, \quad (7.4)$$

where

$$\mathbf{y} = (\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, \dots, \bar{\mathbf{y}}_N)^T = \mathbf{P}^\epsilon \mathbf{e}, \quad (7.5)$$

and $\bar{\mathbf{y}}_i$ has n_i elements. The vector $\mathbf{x}$ is partitioned in the same way. We remark that $\|\mathbf{y}\|_\infty \leq \text{degree of coupling}(\mathbf{P})$, and thus we have ϵ -quasi-lumpability.

Essentially, the mass in the off-diagonal blocks which was preventing us from lumping $\mathbf{P}$ has been accumulated in an extra state.

3. Given that $\mathbf{P}$ is not exactly lumpable (i.e., $\mathbf{y} \neq \mathbf{o}$), $\mathbf{P}^s$ will not be lumpable. However, the lumpability condition may be enforced by increasing some elements in $\bar{\mathbf{y}}_i$ so as to make each element equal to $\|\bar{\mathbf{y}}_i\|_\infty$ and decreasing the corresponding diagonal elements. If it is possible for any diagonal element to become negative, the diagonal of $\mathbf{P}^s$ may be scaled by performing the transformation

$$\alpha \mathbf{P}^s + (1 - \alpha) \mathbf{I}, \quad (7.6)$$

where $0 < \alpha \leq 1$, on $\mathbf{P}^s$ as suggested in [10]. Let us call the matrix obtained in the end, $\tilde{\mathbf{P}}^s$.

4. $\tilde{\mathbf{P}}^s$ is lumpable, and it may be lumped to form the following matrix:

$$\mathbf{K}^s = \begin{pmatrix} \|\tilde{\mathbf{P}}_{1,1}^s\|_\infty & \|\tilde{\mathbf{P}}_{1,2}^s\|_\infty & \cdots & \|\tilde{\mathbf{P}}_{1,N}^s\|_\infty & \|\bar{\mathbf{y}}_1\|_\infty \\ \|\tilde{\mathbf{P}}_{2,1}^s\|_\infty & \|\tilde{\mathbf{P}}_{2,2}^s\|_\infty & \cdots & \|\tilde{\mathbf{P}}_{2,N}^s\|_\infty & \|\bar{\mathbf{y}}_2\|_\infty \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \|\tilde{\mathbf{P}}_{N,1}^s\|_\infty & \|\tilde{\mathbf{P}}_{N,2}^s\|_\infty & \cdots & \|\tilde{\mathbf{P}}_{N,N}^s\|_\infty & \|\bar{\mathbf{y}}_N\|_\infty \\ \|\bar{\mathbf{x}}_1\|_1 & \|\bar{\mathbf{x}}_2\|_1 & \cdots & \|\bar{\mathbf{x}}_N\|_1 & 0 \end{pmatrix}. \quad (7.7)$$

5. Bounds on the probabilities of the aggregates may be obtained using Courtois and Semal's method given in Section 6.2 (see also [6], [40]).

When constructing $\mathbf{P}^e$ the nonzero elements in blocks should be arranged, if at all possible, so that there is a minimum number of nonzero columns in $\mathbf{P}^e$. If all columns corresponding to states in $\mathcal{S}_i$ are zero in $\mathbf{P}^e$, then $\mathbf{x}_i = \mathbf{o}$, and the stationary probability of the i^{th} aggregate may be determined exactly in floating-point form. An intuitive explanation for this fact is the following. The transitions in $\mathbf{P}^e$ are essentially the transitions to and out of the extra state (in $\mathbf{P}^s$). Therefore, if it is not possible to make a transition to state, say, s in the matrix $\mathbf{P}^e$ (i.e., the column

of $\mathbf{P}^e$ that corresponds to state s is $\mathbf{o}$), then it will not be possible to return to state s from the extra state. This being so, the corresponding element in $\mathbf{x}^T$ needs to be 0. If all states in a NCD block possess this property, then the element in the last row of the lumped chain $\mathbf{K}^s$ corresponding to that NCD block has to be 0.

7.3 A Simple Example

Consider the 8×8 Courtois' matrix presented in Section 4.1. The degree of coupling for this matrix is 0.001. Note that, $\mathbf{P} = \mathbf{P}^- + \mathbf{P}^e$ as required.

$$\mathbf{P} = \begin{pmatrix} .85 & & .149 & .0009 & & .00005 & & .00005 \\ .1 & .65 & .249 & & .0009 & .00005 & & .00005 \\ .1 & .8 & .0996 & .0003 & & & .0001 & \\ & .0004 & & .7 & .2995 & & .0001 & \\ .0005 & & .0004 & .399 & .6 & .0001 & & \\ & .00005 & & & .00005 & .6 & .2499 & .15 \\ .00003 & & .00003 & .00004 & & .1 & .8 & .0999 \\ & .00005 & & & .00005 & .1999 & .25 & .55 \end{pmatrix}$$

$$\mathbf{P}^- = \begin{pmatrix} .85 & & .149 & .0003 & & .00005 & & .00005 \\ .1 & .65 & .249 & & .0003 & .00005 & & .00005 \\ .1 & .8 & .0996 & .0003 & & & .0001 & \\ & .0004 & & .7 & .2995 & & .0001 & \\ & & .0004 & .399 & .6 & .0001 & & \\ & .00005 & & & .00004 & .6 & .2499 & .15 \\ .00002 & & .00003 & .00004 & & .1 & .8 & .0999 \\ & .00005 & & & .00004 & .1999 & .25 & .55 \end{pmatrix}$$

$$\mathbf{P}^e = \begin{pmatrix} & & .0006 & & & & & \\ & & & .0006 & & & & \\ .0005 & & & & & & & \\ & & & .00001 & & & & \\ .00001 & & & & .00001 & & & \end{pmatrix}$$

$$\mathbf{P}^s = \begin{pmatrix} .85 & & .149 & .0003 & & .00005 & & .00005 & .0006 \\ .1 & .65 & .249 & & .0003 & .00005 & & .00005 & .0006 \\ .1 & .8 & .0996 & .0003 & & & .0001 & & \\ & .0004 & & .7 & .2995 & & .0001 & & \\ & & .0004 & .399 & .6 & .0001 & & & .0005 \\ & .00005 & & & .00004 & .6 & .2499 & .15 & .00001 \\ .00002 & & .00003 & .00004 & & .1 & .8 & .0999 & .00001 \\ & .00005 & & & .00004 & .1999 & .25 & .55 & .00001 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \end{pmatrix}$$

Note that there are no transitions to states 2, 3, 6, 7, and 8 in $\mathbf{P}^c$. Hence, x_2, x_3, x_6, x_7 , and x_8 in $\mathbf{P}^s$ have to be zero. Observe that $\mathbf{P}^s$ is still not lumpable. For it to be lumpable, the last column has to be modified.

$$\tilde{\mathbf{P}}^s = \begin{pmatrix} .85 & & .149 & .0003 & & .00005 & & .00005 & .0006 \\ .1 & .65 & .249 & & .0003 & .00005 & & .00005 & .0006 \\ .1 & .8 & .099 & .0003 & & & .0001 & & .0006 \\ & .0004 & & .6995 & .2995 & & .0001 & & .0005 \\ & & .0004 & .399 & .6 & .0001 & & & .0005 \\ & .00005 & & & .00004 & .6 & .2499 & .15 & .00001 \\ .00002 & & .00003 & .00004 & & .1 & .8 & .0999 & .00001 \\ & .00005 & & & .00004 & .1999 & .25 & .55 & .00001 \\ x_1 & & & x_4 & x_5 & & & & \end{pmatrix}$$

The lumped chain is given by

$$\mathbf{K}^s = \begin{pmatrix} .999 & .0003 & .0001 & .0006 \\ .0004 & .999 & .0001 & .0005 \\ .00005 & .00004 & .9999 & .00001 \\ \|\bar{x}_1\|_1 & \|\bar{x}_2\|_1 & 0 & 0 \end{pmatrix}.$$

$$\|\bar{x}_1\|_\infty = 1, \|\bar{x}_2\|_\infty = 0 \Rightarrow z_1 = [0.36923, 0.13077, 0.50000]$$

$$\|\bar{x}_1\|_\infty = 0, \|\bar{x}_2\|_\infty = 1 \Rightarrow z_2 = [0.16071, 0.33929, 0.50000]$$

$$0.16071 \leq \xi_1 \leq 0.36923$$

$$0.13077 \leq \xi_2 \leq 0.33929$$

$$0.50000 \leq \xi_3 \leq 0.50000 \Rightarrow \xi_3 = 0.50000$$

and $\xi_1 + \xi_2 + \xi_3 = 1$ in 5 decimal digits of accuracy.

We have obtained the stationary probabilities of aggregates by solving for the stationary vector of the original 8×8 chain. The probabilities accurate to 5 decimal digits (from Section 4.1) are:

$$\xi_1 = 0.22253, \quad \xi_2 = 0.27747, \quad \xi_3 = 0.50000$$

Hence, bounding the stationary probabilities of the aggregates of a NCD Markov chain amounts to solving at most N , $(N + 1) \times (N + 1)$, systems. These linear systems differ only in the last row. Therefore, only one LU decomposition needs to be performed as we mentioned in Section 6.2. Assuming that the transposed systems of equations are solved, the upper-triangular matrices will be different in the last columns only. Hence, the last column in each one of these systems needs to be treated separately during the triangularization phase. Thereafter, all back substitutions may be performed in parallel. Consequently, a solution method such as Gaussian elimination has a time complexity of $O(N^3)$ in the computation of the bounds.

If the chain is sparse and there are symmetries in the chain, it is quite likely that some elements of the vector $\mathbf{x}$ will turn out to be zero, thus tightening the bounds further as in the example considered. However, although ϵ is always less than or equal to the degree of coupling of the NCD Markov chain, $\mathbf{K}^s$ will have diagonal elements close to 1, and it seems that the bounds obtained by the procedure generally will not be tight. The ill-conditioned nature of NCD Markov chains is once again noticed, but this time from a different perspective. Observe that, the more information one has regarding the distribution of the probability mass in $\mathbf{x}^T$, the tighter the bounds become. In fact, there is a distribution $\mathbf{x}^T$ which gives the stationary probabilities of the coupling matrix.

Chapter 8

Concluding Remarks

In this thesis, stability and accuracy issues pertaining to iterative aggregation-disaggregation (IAD), which is very effective in the solution of ill-conditioned nearly completely decomposable (NCD) Markov chains, are addressed. It is shown that a modified version of Gaussian elimination (called the Grassman-Taksar-Heyman method), which executes in positive arithmetic, may be employed in both steps of IAD to enhance stability and relative error of the algorithm. The outcome of our numerical experiments suggest a low relative error in IAD with GTH even for large problems considered. However, although not obvious at first, a row-wise sparse storage implementation of GTH has an inherent complication which is difficult to overcome unless extra time and/or storage is spent.

Secondly, the arithmetic mean (AM) method, which is geared towards multiprocessor computers, is shown to be applicable to the computation of the stationary vector of Markov chains. The method is most effective on matrices which are strongly asymmetric. Two versions of the AM method considered are suitable for the solution of systems having a block-tridiagonal structure, and both methods are highly parallelizable.

It is also concluded that dynamic state exploration is a technique suitable for coping with vast state spaces when the performance measures of interest are of those

which require knowledge of the stationary probabilities of most probable states. The technique is well-suited to bounding performance measures of systems in which each state has only a few states within one-step transition and a probability mass such as 0.99999 is accumulated in 100-200 states. Dynamic state exploration is likely to be instrumental in highly available and reliable systems, and for that matter, it should be used with care.

Finally, it is shown that NCD Markov chains are quasi-lumpable. Following this idea, bounds (though not necessarily tight) on the stationary probability of each aggregate may be obtained in an inexpensive way.

Bibliography

- [1] L. M. Adams and E. G. Ong, Additive polynomial preconditioners for parallel computers, *Parallel Computing* **9**, (1988/9), 333–345.
- [2] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York (1979).
- [3] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, Pitman, London (1979) [reprinted by Dover, New York (1991)].
- [4] W. L. Cao and W. J. Stewart, Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains, *J. ACM* **32**, (1985), 702–719.
- [5] P.-J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York (1977).
- [6] P.-J. Courtois and P. Semal, Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition, *J. ACM* **31**, (1984), 804–825.
- [7] D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*, John Wiley & Sons, New York (1965).
- [8] D. D. Dimitrijevic and M.-S. Chen, Dynamic state exploration in quantitative protocol analysis, *Protocol Specification, Testing and Verification, IX '90*, (1990), North-Holland, 327–338.
- [9] D. J. Evans and W. Yousif, The solution of two-point boundary value problems by the Alternating Group Explicit (AGE) method, *SIAM J. Sci. Statist. Comput.* **9**, (1988), 474–484.
- [10] G. Franceschinis and R. R. Muntz, Bounds for quasi-lumpable Markov chains, *Performance*, (1993), 1–29.
- [11] I. Galligani and V. Ruggiero, Solving large systems of linear ordinary differential equations on a vector computer, *Parallel Computing* **9**, (1988/9), 359–365.
- [12] I. Galligani and V. Ruggiero, The arithmetic mean method for solving essentially positive systems on a vector computer, *Int. J. Computer Math.* **32**, (1989), 113–121.

- [13] I. Galligani, A splitting method for large sparse linear systems on a multivector computer, in *Computer Algorithms for Solving Linear Algebraic Equations—The State of the Art*, ed. by E. Spedicato, NATO ASI Series, Springer-Verlag, Berlin, (1991), 207–215.
- [14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore (1989).
- [15] W. K. Grassmann, M. I. Taksar and D. P. Heyman, Regenerative analysis and steady state distributions for Markov chains, *Oper. Res.* **33**, (1985), 1107–1116.
- [16] W. J. Harrod and R. J. Plemmons, Comparison of some direct methods for computing the stationary distributions of Markov chains, *SIAM J. Sci. Statist. Comput.* **5**, (1984), 453–469.
- [17] D. P. Heyman, Further comparisons of direct methods for computing stationary distributions of Markov chains, *SIAM J. Sci. Statist. Comput.* **8**, (1987), 226–232.
- [18] P. G. Hoel, S. C. Port and C. J. Stone, *Introduction to Stochastic Processes*, Houghton Mifflin, Boston (1972).
- [19] L. C. Kaufman, Matrix methods for queuing problems, *SIAM J. Sci. Statist. Comput.* **4**, (1983), 525–552.
- [20] J. P. Kavanagh and M. Neumann, Consistency and convergence of the parallel multisplitting method for singular M-matrices, *SIAM J. Matrix Anal. Applic.* **10**, (1989), 210–218.
- [21] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, D. Van Nostrand, New York (1960).
- [22] L. Kleinrock, *Queueing Systems 1*, John Wiley & Sons, New York (1975).
- [23] J. R. Koury, D. F. McAllister and W. J. Stewart, Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains, *SIAM J. Alg. Disc. Meth.* **5**, (1984), 164–186.
- [24] P. Lancaster and M. Tismenetsky, *The Theory of Matrices: With Applications*, Academic Press, Orlando (1985).
- [25] J. C. S. Lui and R. R. Muntz, Algorithmic approach to bounding the mean response time of a minimum expected delay routing system, *Performance Evaluation Review* **20**, (1992), 140–151.
- [26] C. D. Meyer and R. J. Plemmons, Convergent powers of a matrix with applications to iterative methods for singular linear systems, *SIAM J. Numer. Anal.* **14**, (1977), 699–705.
- [27] C. D. Meyer, Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems, *SIAM Rev.* **31**, (1989), 240–272.

- [28] C. D. Meyer, Sensitivity of the stationary distribution of a Markov chain, *SIAM J. Matrix Anal. Applic.* **15**, (1994), 715–728.
- [29] C. D. Meyer, personal communication.
- [30] M. K. Molloy, *Fundamentals of Performance Modeling*, Macmillan, New York (1989).
- [31] M. Neumann and R. J. Plemmons, Convergent nonnegative matrices and iterative methods for consistent linear systems, *Numer. Math.* **31**, (1978), 265–279.
- [32] C. A. O’Cinneide, Entrywise perturbation theory and error analysis for Markov chains, *Numer. Math.* **65**, (1993), 109–120.
- [33] D. P. O’Leary and R. E. White, Multi-splittings of matrices and parallel solution of linear systems, *SIAM J. Alg. Disc. Meth.* **6**, (1985), 630–640.
- [34] C. C. Paige, P. H. Styan and P. G. Wachter, Computation of the stationary distribution of a Markov chain, *J. Statist. Comput. Simulation* **4**, (1975), 173–186.
- [35] B. Philippe, Y. Saad and W. J. Stewart, Numerical methods in Markov chain modelling, *Oper. Res.* **40**, (1992), 1156–1179.
- [36] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, London (1984).
- [37] V. Ruggiero and E. Galligani, An iterative method for large sparse linear systems on a vector computer, *Computers Math. Applic.* **20**, (1990), 25–28.
- [38] V. Ruggiero and E. Galligani, A parallel algorithm for solving block tridiagonal linear systems, *Computers Math. Applic.* **24**, (1992), 15–21.
- [39] P. J. Schweitzer, A survey of aggregation-disaggregation in large Markov chains, in *Numerical Solution of Markov Chains*, ed. by W. J. Stewart, Marcel Dekker, Inc., New York, (1991), 63–88.
- [40] P. Semal, Analysis of Large Markov Models, Bounding Techniques and Applications, Doctoral thesis, Université Catholique de Louvain, Louvain, Belgium (1992).
- [41] E. de Souza e Silva and P. Mejía, State space exploration in Markov models, *Performance Evaluation Review* **20**, (1992), 152–166.
- [42] E. de Souza e Silva and H. R. Gail, Performability analysis of computer systems: from model specification to solution, *Performance Evaluation* **14**, (1992), 157–196.
- [43] H. Simon and A. Ando, Aggregation of variables in dynamic systems, *Econometrica* **29**, (1961), 111–138.
- [44] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York (1973).

- [45] G. W. Stewart and G. Zhang, On a direct method for the solution of nearly uncoupled Markov chains, *Numer. Math.* **59**, (1991), 1–11.
- [46] W. J. Stewart, A comparison of numerical techniques in Markov modelling, *Comm. ACM* **21**, (1978), 144–152.
- [47] W. J. Stewart, MARCA: Markov Chain Analyzer, A software package for Markov modeling, in *Numerical Solution of Markov Chains*, ed. by W. J. Stewart, Marcel Dekker, Inc., New York, (1991), 37–61.
- [48] W. J. Stewart and W. Wu, Numerical experiments with iteration and aggregation for Markov chains, *ORSA J. Comput.* **4**, (1992), 336–350.
- [49] W. J. Stewart, *An Introduction to the Numerical Solution of Markov Chains*, to be published by Princeton University Press, New Jersey (1994).
- [50] Y. Takahashi, A lumping method for the numerical calculation of stationary distributions of Markov chains, Research Report B-18, Dept. of Information Sciences, Tokyo Institute of Technology (1975).
- [51] H. M. Taylor and S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, Florida (1984).
- [52] J. Ye and S.-Q. Li, Analysis of multi-media traffic queues with finite buffer and overload control – Part I: Algorithm, *Proc. IEEE INFOCOM '91*, (1991), 1464–1474.