The Propagation of Errors in the Numerical Solution of Markov Models

by

Brenan Joseph Mc Carragher

S.B., Massachusetts Institute of Technology (1988)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1989

Signature of the Author _____
Department of Aeronautics and Astronautics
May 1989

Certified by _____
Dr. Philip S. Babcock, IV
Thesis Supervisor, C.S. Draper Laboratory, Inc.

Certified by _____
Professor Wallace E. Vander Velde
Thesis Supervisor, Professor of Aeronautics and Astronautics

Accepted by _____
Professor Harold Y. Wachman
Chairman, Departmental Graduate Committee

The Propagation of Errors in the Numerical Solution of Markov Models

by

Brenan Joseph Mc Carragher

Submitted to the Department of Aeronautics and Astronautics
on May 12, 1989
in partial fulfillment of the requirements for
the degree of Master of Science

Abstract

Equations that bound the roundoff and the integration errors incurred in the numerical solution of the matrix exponential as applied to Markov models are developed. The power series solution techniques of Taylor and Padé are considered. Additionally, error bounding equations for several error reduction techniques are developed. In order to determine efficiency, the computer work for each of the solution methods is given. The error bounding equations are combined with the computer work equations to give a comprehensive comparison of the solution techniques on a basis of the amount of computer work to achieve a given accuracy. Finally, a methodology for the automatic selection of the integration technique and the integration parameters that produce a user-specified accuracy in the minimum amount of computer work is developed. A suggested method for the solution of this constrained minimization problem is presented.

Acknowledgements

I would like to express my heartfelt gratitude to the many people who made this process called thesis much more enjoyable:

Professor Vander Velde, for asking the right questions to get me focused on the whole problem.

Debbie Allinger, Monica Hutchins, Frank Leong, Gene Rosch, Andrei Schor, and Jeff Zinchuk, for the support and the harassment of being a student at Draper.

Phil Babcock, for having the confidence to give me the freedom.

My other friends, especially those at 1st RPC, for reminding me that there is more to life than thesis.

My family, for the love and encouragement you have always given me.

My parents, for their innumerable sacrifices, unceasing love, and unwavering faith. I could never have done it without you.

This report was prepared at The Charles Stark Draper Laboratory, Incorporated under an internal research and development contract.

Publication of this report does not constitute approval by the Draper Laboratory of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

I hereby assign my copyright of this thesis to The Charles Stark Draper Laboratory, Incorporated, Cambridge, Massachusetts.

Brenan J. McCarragher

Permission is hereby granted by The Charles Stark Draper Laboratory, Incorporated to the Massachusetts Institute of Technology to reproduce any or all of this thesis.

In Loving Memory of My Grandfather

Dr. Alfred J. Horschak

Table of Contents

List of Figures

List of Tables

Nomenclature

A	continuous transition matrix
a, b	transition rates
AIE	absolute integration error
ARE	absolute roundoff error
CW	computer work
$D_q(At)$	denominator polynomial of degree q for the Padé series
E	relative error matrix or vector
ϵ	machine precision
E_i	error due to matrix inversion
EP_j	extrapolation parameter for the j^{th} level of extrapolation
E^R	renormalized error matrix
f	number of failure levels; number of states in a chain model
F, G, H, J	dummy vectors for the accumulator algorithms
factor	chain model approximation to a certain state for bounding purposes
flop	basic unit of computer work
IE	integration error
IE^0	integration error of first extrapolation
IE^∞	integration error of second extrapolation
IE(n)	integration error for n^{th} integration pass using time step of $\frac{\Delta t}{2^{n-1}}$

j	level of extrapolation
K	discrete probability transition matrix
λ_{ij}	continuous transition rate from S_i to S_j
λ_{max}	maximum transition rate in a given Markov model
λ_{min}	minimum transition rate in a given Markov model
M	base matrix
m	number of states in the Markov model; dimension of the base (or transition) matrix
M_k	Taylor base matrix of degree k
mop	computer work necessary to multiply two matrices
M_{pq}	Padé base matrix of numerator degree p and denominator degree q
M^R	renormalized base matrix
n	number of time step to reach the final time
n'	switching point for variable renormalization
n^*	number of time steps used to determine factor
$N_p(At)$	numerator polynomial of degree p for the Padé series
P^*	exact probability
P'	probability after first integration pass with full time step
$P^{(')}$	probability after (') th integration pass with fractional time step
$P^{(\circ)}$	probability after ($\circ$) th extrapolation
$P(n)$	state probability vector after the n^{th} trial
$P(t)$	state probability vector at time t
$P_i(n)$	probability of S_i after the n^{th} trial

$P_i(t)$	probability of S_i at time t
p_{ij}	transition probability from S_i to S_j
P^R	renormalized state vector
R(At)	infinite Padé series for the exponential of At
RE	roundoff error
RIE	relative roundoff error
$R_{pq}(At)$	truncated Padé series with a numerator of power p and a denominator of power q for the exponential of At
RRE	relative roundoff error
s	switching point for the combined algorithm
S_i	i^{th} state in a Markov model
SRE	specified relative error
T(At)	infinite Taylor series for the exponential of At
t_c	characteristic time of the system
$T_k(At)$	truncated Taylor series for the exponential of At
t_{max}	final time
V, W, X, Y, Z	accumulator vectors for the accumulator algorithms
vop	computer work necessary to multiply a vector by a matrix
Δt	time step
Δt^*	time step used to determine factor

Chapter 1

Introduction

1.1 Background

Markov models are stochastic tools that are used extensively in queueing theory and reliability analysis. They are described by a set of either difference or differential equations. The solution of these equations is often difficult to obtain in closed form because of the highly coupled nature of the equations. Thus, some form of numerical integration is necessary.

One of the most common classes of Markov models encountered is the discrete-state, continuous-transition Markov model. The solution to the set of differential equations describing this class of models is the matrix exponential. Specializing further, we require that the matrix of transitions be time invariant. In this case, the solution to the set of differential equations is the time-invariant matrix exponential, commonly referred to as simply the matrix exponential.

There are many ways to calculate the matrix exponential. Moler and Van Loan [Moler and Van Loan, 1978] discuss computational stability and efficiency of nineteen different methods to compute the matrix exponential. They state that a form of a power series method, called scaling and squaring, is one of the most effective they know when properly implemented. The power series that they suggest are Taylor series and Padé series. This paper will concentrate on these power series solutions and some of their variations

The thrusts of the Moler and Van Loan paper are computational stability and efficiency. Only brief attention is given to the areas of roundoff and integration error, despite roundoff errors being one of the power series method's weakest points. The focus of this thesis, therefore, is the quantification and bounding of the roundoff and integration errors associated with the power series methods of solution for the matrix exponential applied to Markov models.

1.2 Problem Description

The fact that error is incurred when integrating numerically is generally accepted. The amount of error incurred, on the other hand, is often debated. The user is required to determine a solution algorithm and the integration parameters. Each of the user specified parameters has direct impact on the amount of numerical error incurred. Unfortunately, the impact of these parameters on the accuracy of the integration is not readily obvious. In order to appropriately select the solution algorithm and the integration parameters, knowledge and experience with numerical analysis is necessary.

Solutions to the problem of parameter selection, however, are subtle. A simple approach to numerical error is to ignore the effects and hope that the results generated are sufficient. Obviously, this is not a desired solution. A more sophisticated attempt is to track the error during the integration process, resulting in both an approximate result and an error estimate [Ward, 1977; Butler, 1988]. Unfortunately, the user may have to integrate the problem several times until a solution with an acceptable accuracy is obtained which could be very costly. Alternatively, the accuracy obtained could be much greater than desired or warranted meaning that extra computer time was wasted in the solution process. These two events occur due to a lack of *a priori* knowledge about the propagation of the numerical error based on the system and integration parameters.

The problem, then, is to develop a methodology that has *a priori* knowledge about the error propagation patterns. Because this insight is based on the system and integration parameters, it can be exploited to determine a set of integration parameters that guarantees the solution is within the user-specified accuracy in the minimum amount of computer work. The methodology must be robust enough to be applicable to a general Markov model with constant transition rates.

1.3 Objectives and Overview

The objective of this research is to develop a robust methodology for error bounding that will automatically determine the integration method and the integration parameters while insuring that the result will meet a user-specified accuracy in the minimum amount of computer work. The methodology will need to be applicable to a general Markov model. Additionally, it should be easily implemented on a digital computer so the process of determining the integration technique and parameters can be automated, requiring no input from the user other than the desired accuracy.

Chapter 2 is a discussion of the Markov process and Markov models. It is supplied for general background. Both discrete-transition and continuous-transition models are discussed. A simplification, referred to a chain model, is introduced. The chain model is often used for error bounding purposes. The characteristics of a Markov model that are important in the solution of the matrix exponential are defined. Lastly, an example from reliability analysis of the use of Markov models is given. This example is carried throughout the report as a basis of numerical results to demonstrate and clarify some of the issues and techniques.

A discussion of the computation of the matrix exponential is given in Chapter 3. This discussion includes how the set of differential equations that describe the Markov model are solved by computing the matrix exponential. The power series solution techniques of Taylor and Padé are shown to be insufficient by themselves. However, a method know as scaling and squaring, when used in conjunction with the Taylor and Padé series, results in an effective and accurate solution. Since efficiency is also of concern, the amount of computer work required by each of the methods is also given.

In order to develop an *a priori* error bounding methodology, the error propagation patterns must first be understood. Chapter 4 discusses the propagation patterns for roundoff error. Once the patterns have been identified, equations are developed that can be used to bound the roundoff error. The roundoff error patterns suggest means by which the roundoff error could be reduced. Two such roundoff error reduction techniques are developed and shown to be beneficial for certain classes of Markov models. Lastly, numerical results of the roundoff error are given for the various solution techniques described.

Integration error is heavily dependent on the series solution method employed. For this reason the integration error of the Taylor series techniques is discussed in Chapter 5, while the integration error of the Padé series techniques is explained in Chapter 6. The approach in both chapters is similar to that taken for the roundoff error. First, the integration error propagation patterns are given, and then equations to bound the error are developed. A series of bounds are given, with each tighter bound requiring more computer work. The bounding equations suggest several methods to reduce the integration error, but only one of these leads to a new algorithm. This new technique, known as Richardson Extrapolation, reduces the integration error for the general Markov model. The chain model that was discussed in Chapter 2 is now used to bound the integration error. Finally, numerical results are given for the integration error incurred by the various techniques.

Chapters 5 and 6 are very similar in their format and approach. As such, Chapter 6 borrows heavily from Chapter 5.

Equivalent work comparisons are conducted in Chapter 7. The integration methods are compared on the basis of how much computer work is necessary to obtain a given accuracy. Essentially, this chapter collects the information given previously and presents it based on the critical parameters of accuracy and computer work. The comparison studies are given by power series methods first, and then the final section compares all the methods. Many methods are shown to be effective for a limited class of models, but not necessarily appropriate for the general case.

Finally, the methodology is completed in Chapter 8, when the automatic selection of the integration technique and the integration parameters is discussed. The necessary inputs from the user are specified. Essentially, the inputs required are the Markov model system parameters and the desired accuracy. The work-error equations that were presented earlier are converted into a useful form. Using these work-error equations, the minimization of the computer work subject to the error constraint is discussed. A solution technique and algorithm for this minimization are suggested.

Chapter 2

The Markov Process and Markov Models

2.1 The Discrete-State Markov Process

Many stochastic processes are defined by probabilistic descriptions of a series of independent trials or experiments. These processes are considered to have no memory. That is, the probability of a certain event occurring is independent of any previous trials and independent of the current state of the system. The Bernoulli and the Poisson are two of the better known independent stochastic processes.

In contrast to the no-memory stochastic processes, the Markov process is characterized by a series of dependent trials. As such, Markov processes do have memory, although the dependence of future events on past events is of a very simple nature. The probability distributions of the future behavior of a Markov process are dependent only on the present state and not on how the system arrived in that state. Or alternatively, the 'future' of a Markov process depends only on the 'present' and not on the 'past'.

The above statement--the future of a Markov process depends only on the present and not on the past--can be taken to be an informal definition of the Markov condition. That is, this statement can be used define or test a Markov process. The Markov condition requires that the conditional probabilities that describe the future of the system not depend on the past history of the process. The present state of the system must specify all the necessary historical information relevant to the future of the process.

Markov processes may be used in many situations. They are often used in operations research and queuing theory to describe the dynamics of queues. For example, the number of people waiting in line at a checkout counter is dependent on the number of people that were waiting in the previous instant but not on the number of people that were already served. Markov models are also used in system reliability analysis to determine the probability that a certain system is working at a given instant in time. Again, the future of the system is dependent on the the present operational state and not on how long the system has been in that operational state.

2.2 Discrete-State Discrete-Transition Markov Models

Consider a system that can be described at any time by a set of mutually exclusive and collectively exhaustive states, $S_1, S_2, \ldots S_m$. We assume the set of states to be finite or countably infinite. Additionally, we allow the system to change state, or make a transition, only at discrete instances of time. The particular instances of time at which a transition may occur are called trials; the first instance is called the first trial, the second instance is called the second trial, and so forth.

We now denote $S_i(n)$ to be the event that the system is in the i^{th} state immediately following the n^{th} trial. The probability of such an event is written as $P[S_i(n)]$, often simply written as $P_i(n)$. Given the complete history of the system and that the system is in state i immediately following the n^{th} trial, $S_i(n)$, there is a conditional probability that the system will make a change of state to state j in the next instance of time, $S_j(n+1)$. This conditional probability is called the transition probability and is denoted as $p_{ij}(n)$. This can be written succinctly as

$$P[S_j(n+1) \mid S_i(n)\, S_{i-1}(n-1)\, S_{i-2}(n-2) \ldots S_o(0)] = p_{ij}(n)$$

Note that p_{ii} is the conditional probability that the system enters S_i given that it is already in S_i. This is equivalent to saying that the system never left S_i.

The general process referred to above is know as a discrete-state, discrete-transition Markov process because both the states and the transitions are discrete. A system is said to be discrete-state, discrete-transition if the transition probabilities for a series of trials satisfy the Markov condition as given here

$$P[S_j(n) \mid S_i(n-1)\, S_{i-1}(n-2)\, S_{i-2}(n-3) \ldots S_o(0)] = P[S_j(n) \mid S_i(n-1)]$$

If the state of the system immediately before the n^{th} trial is known, the Markov condition requires that the transition probabilities for the n^{th} trial do not depend on any of the past history of the system. Therefore the present state of the system contains all the necessary information to accurately specify the future of the system.

In many cases the conditional transition probabilities do not depend on the trial number and may be written as

$$P[S_j(n+1) \mid S_i(n)] = p_{ij}$$

The quantity p_{ij} is now the conditional probability that the system will transition on the next trial to S_j given that the system is presently in S_i. Since probabilities are non-negative and the states are mutually exclusive and collectively exhaustive, we always have $0 \leq p_{ij} \leq 1$. Also, since the state must make a transition into some state (even if the system returns to its original state), we have

$$\sum_{j=1}^{m} p_{ij} = 1 \qquad i = 0, 1, \ldots, m$$

Additionally, since the states are mutually exclusive and collectively exhaustive, the system must be in one of those states at all instances in time. We then have what is referred to as the conservation of system probability. Namely, the probability of all the possible states must sum to unity.

$$\sum P_i(n) = 1 \qquad \text{for all } n \geq 0$$

It is often convenient to represent the transition probabilities in a (mxm) transition matrix, $K = [p_{ij}]$, where p_{ij} is the element in the i^{th} column and the j^{th} row.

$$K = \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{m1} \\ p_{12} & p_{22} & \cdots & p_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ p_{1m} & p_{2m} & \cdots & p_{mm} \end{bmatrix}$$

We also define the state vector P(n) to be the vector of state probabilities immediately following the n^{th} trial.

$$P(n)^T = [P_1(n), P_2(n), \ldots, P_m(n)]$$

The solution to the probabilities of the states is then given by

$$P(n+1) = K\,P(n)$$

where P(n) is the state vector immediately after the n^{th} trial. The above equation can be used in a iterative fashion to give the probabilities immediately following the desired n^{th} trial. More about solution techniques is given in Chapter 3.

As a short example, consider a system of two biased coins. The first coin has a probability of heads of 0.7 and a probability of tails of 0.3. The second coin has a probability of heads of 0.6 and a probability of tails of 0.4. In order to make this a Markov

process we need to impose some sort of memory, or dependence on the previous state. Therefore, if the last toss was a head, coin one will be flipped; if the last toss was a tail, coin two will be flipped. Now we have a discrete-state, discrete-transition Markov process where the outcome of the future event is dependent only on the present state of the system and not on how the system arrived in that state.

We will define S_1 as heads and S_2 as tails. The transition probability matrix and the discrete-transition discrete-state Markov model are as given below. In the graphical model, a circle is representative of a state of the system while a line with an arrow is representative of a transition probability in the direction of the arrow.

$$K = \begin{bmatrix} 0.7 & 0.6 \\ 0.3 & 0.4 \end{bmatrix}$$

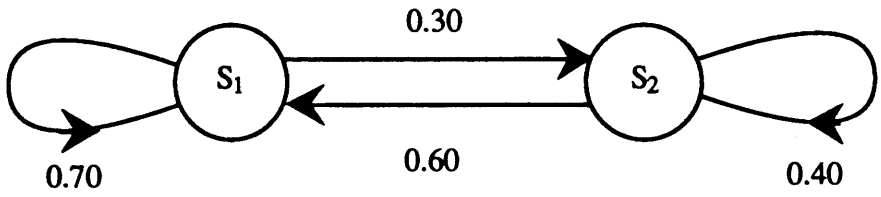


Figure 2-1: Two-State Discrete-Transition Cyclical Markov Model

If we know the initial state (the first coin to be flipped), we can discern any future probability of getting either a head or a tail on the next flip. Assuming the system initially begins in state 1 (coin 1 is tossed first) the example given above results in the following three state vectors for n = 0, n = 1 and n = 2. P(0) is obviously the initial state probability vector.

$$P(0)^T = [\ 1.00\ \ 0.00\ ]$$

$$P(1)^T = [\ 0.70\ \ 0.30\ ]$$

$$P(2)^T = [\ 0.67\ \ 0.33\ ]$$

Therefore, after two coins were tossed, the probability that the last toss was a head is $P_1(2) = 0.67$, and the probability that the last toss was a tail is $P_2(2) = 0.33$. Notice also that the sum of the components of each vector is unity. This is in accordance with the conservation of system probability as mentioned above.

Suppose we wanted to know how the process behaves after a long period of time. We can than calculate what are known as the limiting-state or steady-state probabilities. The steady-state of a system is achieved when the probabilities of all the states no longer change with increases in trial number. To calculate the limiting-state probabilities, we can step the process through many trials until no change in the state probabilities is noticed. However, for many systems this stepping process may be very time consuming. Alternatively, we can balance the changes in probability of the states. If the increase in probability is equal to the decrease in probability for all states, the system has reached steady-state. It will not change probabilities with additional trials. In our example, this balance of probability can be written as

$$P_1 \, p_{12} = P_2 \, p_{21}$$

Solving for the probability of state 1 gives

$$P_1 = P_2 \frac{p_{21}}{p_{12}}$$

Notice that the trial number is not important since we are trying to find limiting-state probabilities, and hence not included in the equation. From the conservation of system probability, we know that P_1 and P_2 must sum to unity. Using these two equations we can solve for the limiting-state probabilities.

$$P_1 + P_2 = 1$$

$$P_2 \frac{p_{21}}{p_{12}} + P_2 = 1$$

$$P_2 = \frac{1}{1 + \frac{p_{21}}{p_{12}}}$$

For the above example, this results in

$$P_2 = \frac{1}{1 + \frac{0.6}{0.3}} = \frac{1}{3} \approx 0.33$$

Using the conservation of system probability gives

$$P_1 = 1 - P_2 = \frac{2}{3} \approx 0.67$$

These results show that for the coin toss example, steady-state has been achieved after only two tosses of the coin.

2.3 Discrete-State Continuous-Transition Markov Models

Discrete-state continuous-transition Markov models are also concerned with a system that may be described at any time as being in one of a finite or countable infinite number of discrete states, S_i $i = 1,2, \ldots m$. The states must again be mutually exclusive and collective exhaustive. For a continuous-transition process, however, the system may make a state transition at any instant in continuous time. The Markov condition is still satisfied, that is, the future behavior of the process is dependent only on the present state of the system and not on how the system reached that present state. The 'future' is dependent only on the 'present' and not on the 'past'.

In the case of continuous-transition Markov processes, the discrete-transition probabilities are replaced by continuous-transition rates, usually denoted by λ. The transition rates could be functions of time. However, time-varying transition rates greatly complicate the analysis and will not be considered in this paper. In an analogous way to the discrete-transition process, λ_{ij} is the transition rate for going from S_i to S_j. Because they are rates, the units for λ are (events/unit time) or $(\text{time})^{-1}$. For convenience, the probability of being in a state at time t, $P[S_i(t)]$, will be written as $P_i(t)$.

A discrete-transition Markov model resulted in a set of difference equations of the form

$$P(n+1) = K\,P(n)$$

We construct a new state vector after each trial to give the probabilities for the Markov model. A continuous-transition Markov model, however, results in a set of differential equations. Instead of trials, we are now concerned with time; instead of a transition probability matrix K, we are now concerned with a continuous rate transition matrix A. Thus, the form of the differential equations for the solution to the continuous-transition Markov problem is given by

$$\frac{dP(t)}{dt} = A\,P(t)$$

where the continuous state vector is simply given by

$$P(t)^T = [P_1(t)\ P_2(t)\ \ldots\ P_m(t)]$$

The conditional probability that an S_i to S_j transition will occur in an infinitesimal time dt (given the system is in S_i) is given by (λ_{ij} dt). Therefore, the differential change of the probability of S_j in time dt is simply the conditional probability the the S_i to S_j transition occurs multiplied by the probability of being in S_i at time t. This results in the differential equation

$$dP_j(t) = (\lambda_{ij}\, dt)\, P_i(t)$$

This can be written to give the time rate of change in a more conventional form as

$$\frac{dP_j(t)}{dt} = \lambda_{ij}\, P_i(t)$$

In completely determining the continuous-transition matrix A, however, there is a breakdown in the discrete to continuous analogue. The discrete transition probability p_{ii} is the conditional probability of staying in S_i on the next trial given that the system is currently in S_i. This probability could be determined by the fact that the system must make a transition (even if it is to return to the original state) for every trial. But, in continuous time, the system may make a change of state at any interval of time, not only at discrete times. In fact, the system may never make a change of state. Because we are in continuous time, a change of state is considered to have taken place, S_i to S_j, only if $i \neq j$. Obviously, because a change of state does not have to occur, it is not required that the transition rates out of a state sum to unity. Thus, we cannot simply sum the transition rates and subtract this sum from one to get the transition rate for the system staying in a certain state.

Nevertheless, the conservation of system probability still holds. This means that the sum of the probability in all the states of the system must still sum to unity. Therefore, the probability increase in S_j in a given time period due to transition rate λ_{ij} must be accompanied by a corresponding decrease in probability in S_i for the given time period due to transition rate λ_{ij}. This decrease is caused by the transition rate λ_{ii}. Summing over all the transitions exiting S_i gives

$$\lambda_{ii} = -\sum_{\substack{j \\ j \neq i}}^{m} \lambda_{ij}$$

We can now construct the continuous-transition matrix, A.

$$A = \begin{bmatrix} -\sum_j \lambda_{1j} & \lambda_{21} & \cdots & \lambda_{m1} \\ \lambda_{12} & -\sum_j \lambda_{2j} & \cdots & \lambda_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{1m} & \lambda_{2m} & \cdots & -\sum_j \lambda_{mj} \end{bmatrix}$$

The set of differential equations for an entire Markov model can be easily written in matrix notation using the continuous-transition matrix, A. Using this notation, the set of differential equations becomes

$$\begin{bmatrix} \frac{dP_1(t)}{dt} \\ \frac{dP_2(t)}{dt} \\ \cdots \\ \frac{dP_m(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\sum_j \lambda_{1j} & \lambda_{12} & \cdots & \lambda_{1m} \\ \lambda_{21} & -\sum_j \lambda_{2j} & \cdots & \lambda_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{m1} & \lambda_{m2} & \cdots & -\sum_j \lambda_{mj} \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ \cdots \\ P_m(t) \end{bmatrix}$$

This can be written more succinctly as

$$\frac{dP(t)}{dt} = A\,P(t)$$

Knowing the initial conditions P(0), the solution to this set of differential equations is given by the matrix exponential

$$P(t) = e^{At}\,P(0)$$

or

$$P(t) = \exp(At)\,P(0)$$

Methods for the calculation of the matrix exponential, which gives the state probabilities, will be discussed more formally in Chapter 3.

To help understand the continuous-transition Markov model, it may be helpful at this time to introduce the concept of probability flow. Consider probability as a substance

that is fluid and can flow from one state to another state. The flow rate of the probability is equal to the product of the transition rate and the probability of being in the state where the transition originates. Thus, the flow rate from state 1 to state 2 is $\lambda_{12} P_1(t)$. The time rate of change of a state's probability is then equal to the flow entering minus the flow leaving that state. For example, the time rate of change of the probability of state 2 could be

$$\frac{dP_2(t)}{dt} = \lambda_{12} P_1(t) - \lambda_{23} P_3(t)$$

Thus a state with no exiting transitions (transition rates of zero) has no outward probability flow. Likewise, a state with zero probability has no outward probability flow because there is no probability to flow.

The concept of probability flow makes the derivation of the continuous transition matrix easier to understand. As was stated above, the time rate of change of a state's probability is equal to the flow entering minus the flow leaving that state. A differential equation can be therefore be derived for each state.

$$\frac{dP_i(t)}{dt} = \sum_m (\lambda_{mi} P_m) - \sum_n (\lambda_{in} P_n)$$

A set of these state equations, along with a specified initial condition, completely describe the system. Moreover, the above is a short-hand notation for the transition matrix A. Thus, probability flow gives a conceptual view of the meaning of the terms in the transition matrix A.

Obviously, the discrete-transition example of two biased coins cannot be used to demonstrate continuous-transition Markov models. Instead we think of an engineer, her office, and her laboratory. Define S_1 as the state that the engineer is in her office and S_2 as the state that the engineer is in her laboratory. Assume that the lab and the office are connected so that any intermediary places may be neglected. Suppose now that the engineer goes from the office to the lab once every four hours for a rate of $\lambda_{12} = 0.25\ hr^{-1}$. Also suppose that this engineer goes from the lab to the office once every two hours for a rate of $\lambda_{21} = 0.50\ hr^{-1}$. Assuming the engineer never needs to go outside of the office/lab area, this scenario would result in the following discrete-state, continuous-time Markov model.

$$\begin{bmatrix} \frac{dP_1(t)}{dt} \\ \frac{dP_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} -0.25 & 0.50 \\ 0.25 & -0.50 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \end{bmatrix}$$

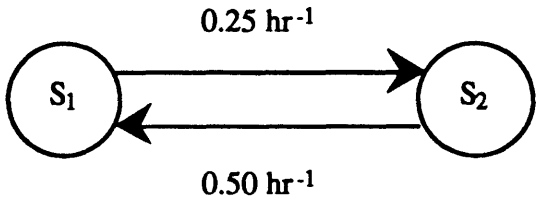


Figure 2-2: Two-State Continuous-Transition Cyclical Markov Model

Assuming that the engineer starts at t = 0 in her office, we have the following state vectors with time given in hours.

$$P^T(0.0) = [1.000 \;\; 0.000]$$

$$P^T(0.5) = [0.896 \;\; 0.104]$$

$$P^T(1.0) = [0.824 \;\; 0.176]$$

$$P^T(2.0) = [0.741 \;\; 0.259]$$

Again, the sum of the components of each vector is unity in accordance with the conservation of system probability. The limiting state probabilities, $P(\infty)$, can be determined in a similar manner to discrete-transition models. We know that the probability flow balances in both directions for all states, so we can set the time rate of change equal to zero to solve for the probabilities. We must again employ the conservation of system probability to solve for state probabilities.

$$\lambda_{12} P_1 = \lambda_{21} P_2$$

$$P_1 = \frac{\lambda_{21}}{\lambda_{12}} P_2$$

$$P_1 + P_2 = 1$$

Thus, the steady-state probabilities are given by

$$P_2(t) = \frac{1}{1 + \frac{\lambda_{21}}{\lambda_{12}}} = \frac{1}{1 + \frac{0.50}{0.25}} \approx 0.667$$

$$P_1(t) = 1 - P_2(t) \approx 1 - 0.667 \approx 0.333$$

This yields the steady state vector

$$P^T(\infty) = [0.667 \;\; 0.333]$$

2.4 Definitions of Terms for Markov Models

It is convenient at this time to define some terms concerning Markov models that are of importance for this paper. Additionally, some common terms that refer to Markov models are given to make this description more complete.

Transient State S_i: At least one state S_k can eventually be reached from S_i, yet the system can never return to S_i from S_k. This state must have a zero limiting-state probability.

Recurrent State S_i: S_i is recurrent provided that the system can return to S_i from all states that can eventually be reached from S_i. This state can have non-zero probability at time equals infinity.

Trapping State S_i: (Also Absorbing State) A state that has no exiting transitions, only entering transitions. Additionally, this state can have non-zero probability at time equals infinity.

Non-Cyclical Markov Model: A Markov model that is made up entirely of transient states and trapping states. The model does not allow the system to return to a state once the system has left that state. This is a pure decay Markov model.

Cyclical Markov Model: A Markov model that contains at least one recurrent state. Once the system has left a certain state S_i, this model allows for the system to return to that state S_i at a future time.

Transition Level: (Also Failure Level) The minimum number of transitions necessary to reach a certain state from the initial state.

Non-Cyclical Chain Model: A model that has only one state per transition level with at most one exiting transition per state and at most one entering transition per state.

Cyclical Chain Model: A cyclical Markov model that has only one state per transition level with at most one forward exiting transition and one reverse exiting transition, as well as at most one forward entering transition and one reverse entering transition.

Steady-State Probabilities: (Also Limiting-State Probabilities). The probabilities at time equals infinity. The probability of the system no longer changes with trial or time. A balance of probability flow is achieved.

As an example, we give a Markov model and label the parts of the drawing that correspond to some of these definitions.

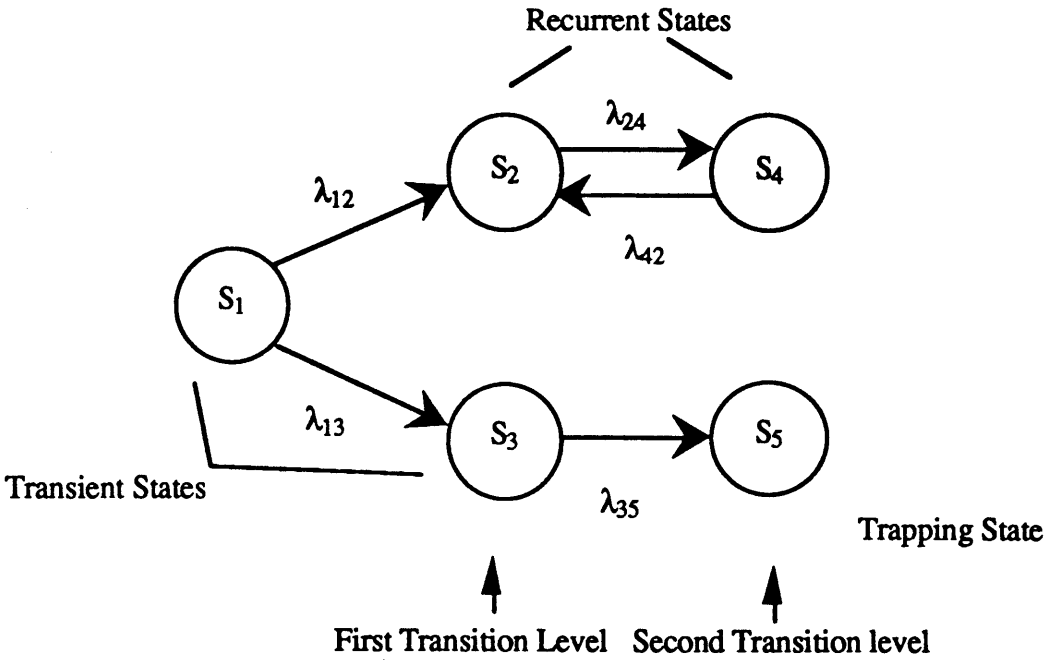


Figure 2-3: General Case Markov Model

2.5 Characteristics of Markov Models

There are two important properties of Markov models that are used frequently in this paper that warrant explanation. The first concept, which has already been introduced, is the conservation of system probability. This property is dependent on the number of states in the system. The second is the concept of the characteristic time of the system. This property is dependent on the particular system parameters, such as the transition rates and the final time of the system.

It was stated earlier that since the states of a Markov model are mutually exclusive and collectively exhaustive, the probability of being in any one of the states must sum to unity given the system was initially in one of these states. For finite-state continuous-transition models, this is written more formally as

$$\sum_{i=1}^{m} P_i(t) = 1$$

Thus the conservation of system probability gives additional information about the system to be solved. It is known that at all instances in time the sum of the probabilities of the states must be unity. Therefore, if the probabilities of all the states but one are known, the probability of the unknown state S_j can be computed by summing the other states and subtracting this sum from unity.

$$P_j(t) = 1 - \sum_{\substack{i \\ i \neq j}} P_i(t)$$

Additionally, if some of the state probabilities are known while others are unknown, the collective probability of the unknown states can be determined by summing the known probabilities and subtracting this sum from unity. Since the state probabilities must be non-negative, a bound on any of the unknown state probabilities can be derived.

$$P_h(t) + P_j(t) + P_k(t) = 1 - \sum_{\substack{i \\ i \neq h,\, j,\, k}} P_i(t)$$

$$P_j(t) \leq 1 - \sum_{\substack{i \\ i \neq h,\, j,\, k}} P_i(t)$$

It will be shown later that the conservation of system probability is instrumental in calculating the numerical errors in the probabilities for the trapping states. Also, the conservation of probability will be used extensively to significantly reduce roundoff error in a technique called variable renormalization.

The other important characteristic of Markov models is the characteristic time. The characteristic time of the system is dependent on the parameters of the model, such as transition rates and times, rather than the model architecture. For a two-state, pure-decay system with transition rate λ_{12}, the characteristic time of the system t_c is defined as the point at which the product of the transition rate λ and the time is unity, or equivalently as the reciprocal of the transition rate.

$$t_c \lambda_{12} = 1$$

$$t_c = \frac{1}{\lambda_{12}}$$

An interesting thing occurs around the characteristic time of the system. For the two-state case given above, the probabilities as functions of time are

$$P_1(t) = e^{-\lambda t}$$

$$P_2(t) = 1 - e^{\lambda t}$$

Solving for $P_1(t_c)$ gives ($\lambda t_c = 1$)

$$P(t_c) = \begin{bmatrix} 0.3678 \\ 0.6321 \end{bmatrix}$$

Thus, the characteristic time is approximately the point at which the majority of the probability is no longer in the first state S_1. Conversely, let us determine the point at which the probability of the first state is exactly one-half. Setting the probability to one-half and solving for t gives

$$0.5 = e^{-\lambda t}$$

$$\ln 0.5 = -\lambda t$$

$$t = \frac{\ln 2}{\lambda} \approx \frac{0.6931}{\lambda} \approx \frac{1}{\lambda}$$

This relationship can often be used as an approximation to the characteristic time for non-cyclical systems. It will be shown later that the point where the first state no longer has the majority of the probability plays an important role in the error propagation.

For systems with more than one transition rate, the characteristic time becomes more complicated. For a general Markov model, the characteristic time is usually determined through the chain model approximations. We will therefore defer further discussion of the characteristic time to the next section.

2.6 Chain Model Approximations

2.6.1 The Non-Cyclical Chain Model

As defined above, a non-cyclical chain model is a model that has only one state per transition level with at most one exiting transition per state and at most one entering transition per state. The chain model is fundamentally easier to solve for the probabilities

because of the simplicity of the architecture. As such it will be used to approximate both the probabilities and the error associated with more complex Markov models.

The figure below shows a four-state non-cyclical Markov chain model with constant transition rates λ_{12}, λ_{23}, λ_{34}.

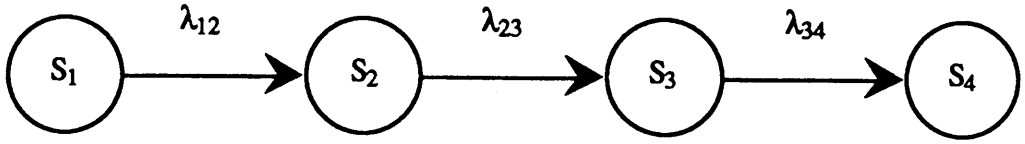


Figure 2-4: Four-State, Non-Cyclical Markov Chain Model

The state probabilities as a function of time, which can be solved for using Laplace transforms or another appropriate method, are given by the following equations.

$$P_1(t) = e^{-\lambda_{12}t}$$

$$P_2(t) = \frac{\lambda_1}{\lambda_2 - \lambda_1}\left[e^{-\lambda_1 t} - e^{-\lambda_2 t}\right]$$

$$P_3(t) = \frac{\lambda_1 \lambda_2}{(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)} e^{-\lambda_1 t} + \frac{\lambda_1 \lambda_2}{(\lambda_2 - \lambda_1)(\lambda_2 - \lambda_3)} e^{-\lambda_2 t} + \frac{\lambda_1 \lambda_2}{(\lambda_3 - \lambda_1)(\lambda_3 - \lambda_2)} e^{-\lambda_3 t}$$

$$P_4(t) = 1 - [P_1(t) + P_2(t) + P_3(t)]$$

$$= 1 - \frac{\lambda_2 \lambda_3}{(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)} e^{-\lambda_1 t} - \frac{\lambda_1 \lambda_3}{(\lambda_2 - \lambda_1)(\lambda_2 - \lambda_3)} e^{-\lambda_2 t} - \frac{\lambda_1 \lambda_2}{(\lambda_3 - \lambda_1)(\lambda_3 - \lambda_2)} e^{-\lambda_3 t}$$

It can be seen that the probabilities sum to unity in accordance with the conservation of system probability. This chain model approximation yields a relatively simple transition matrix A.

$$A = \begin{bmatrix} -\lambda_{12} & 0 & 0 & 0 \\ \lambda_{12} & -\lambda_{23} & 0 & 0 \\ 0 & \lambda_{23} & -\lambda_{34} & 0 \\ 0 & 0 & \lambda_{34} & 0 \end{bmatrix}$$

The chain model can be even further simplified by setting the transition rates equal to each other, $\lambda_{12} = \lambda_{23} = \lambda_{34} = \lambda$. Simplifying the equations for the state probabilities of this system results in the following

$$P_1(t) = e^{-\lambda t}$$

$$P_2(t) = (\lambda t)\, e^{-\lambda t}$$

$$P_3(t) = \frac{(\lambda t)^2}{2} e^{-\lambda t}$$

$$P_4(t) = 1 - [\, P_1(t) + P_2(t) + P_3(t)]$$

$$= 1 - \left[1 + (\lambda t) + \frac{(\lambda t)^2}{2}\right] e^{-\lambda t}$$

This model yields an even simpler transition matrix

$$A = \begin{bmatrix} -\lambda & 0 & 0 & 0 \\ \lambda & -\lambda & 0 & 0 \\ 0 & \lambda & -\lambda & 0 \\ 0 & 0 & \lambda & 0 \end{bmatrix}$$

Obviously, a pattern exists that makes the construction of the transition matrix trivial.

The most important result of this chain model is that a simple equation can be given for any state in the chain model if all the transition rates are equivalent. In general, the probability of any non-trapping state in a chain model can be given by

$$P_{f+1}(t) = \frac{(\lambda t)^f}{f!} e^{-\lambda t}$$

where f is the transition level of the state and not the number of the state. The final trapping state can be determined using the conservation of system probability.

$$P_f(t) = 1 - \sum_{i=0}^{f-1} \frac{(\lambda t)^i}{i!} e^{-\lambda t}$$

These easy formulas for the probabilities of the chain model is the primary advantage of the chain model.

There is another variation of the chain model that will be used to bound the error in a desired state at a certain instant in time. For example, suppose it is desired to obtain an

upper bound for the probability of S_3 in a four-state chain model. Designate λ_{max} as the fastest transition rate in the model and λ_{min} as the slowest transition rate in the model. If all the transitions in the model prior to S_3 (i.e. λ_{12} and λ_{23}) are λ_{max} and the exiting transition for S_3 is λ_{min}, the probability in S_3 would be the maximum possible for the S_3 given the system transition rates. This model gives an upper bound for the probability in S_3.

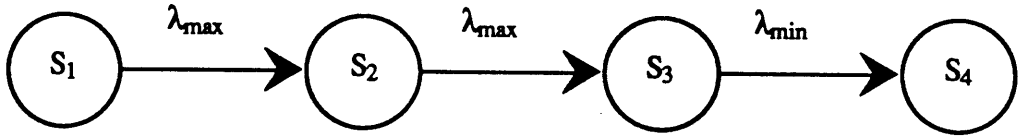


Figure 2-5: Non-Cyclical Chain Model for Upper Limit of State Probability

On the other hand, if a lower bound at a certain point in time is desired, the transition leading into S_3 would be λ_{min} and the transition leading out of S_3 would be λ_{max}. This model gives a lower bound for the probability in S_3.

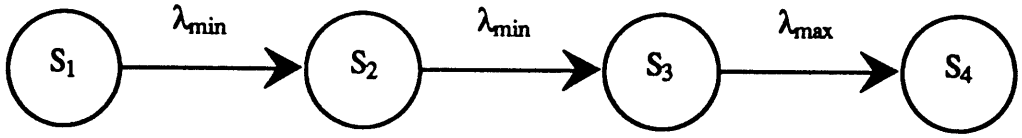


Figure 2-6: Non-Cyclical Chain Model for Lower Limit of State Probability

The closed-form solutions for the probability of S_3 in these models are given below (determined using Laplace transforms). The other states are usually not of interest in this type of analysis. However, they are easily calculated. The probabilities for S_1 and S_2 are unchanged from above with λ_{min} as the transition rate. The probability for S_4 can be calculated using the conservation of system probability.

Upper Bound Probability for S_3

$$P_3(t) = \left(\frac{\lambda_{max}}{\lambda_{min} - \lambda_{max}}\right)\left[\lambda_{max} t\, e^{-\lambda_{max} t} + \frac{\lambda_{max}}{\lambda_{min} - \lambda_{max}}\left(e^{-\lambda_{min} t} - e^{-\lambda_{max} t}\right)\right]$$

Lower Bound Probability for S_3

$$P_3(t) = \left(\frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)\left[\lambda_{min} t\, e^{-\lambda_{min} t} + \frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}\left(e^{-\lambda_{max} t} - e^{-\lambda_{min} t}\right)\right]$$

Obviously, a chain Markov model greatly simplifies the calculations for the state probabilities. Because of this simplicity, the chain model will often be used to approximate

more complex Markov models. The primary reason for the chain model approximation is to simplify the equations for error propagation. In this manner, the error propagation patterns of complex Markov models can be better understood and approximated by the error propagation patterns of the simple chain model.

To assure that the chain model is a good approximation to a complex Markov model, certain guidelines are necessary. The probabilities in a Markov model are very heavily dependent on the transition level as can be seen by the state probabilities given above. An increase in transition level causes an additional (λt) term in the probability while maintaining the same exponential term. Thus, it is important that the chain model approximation should have the same number of transition levels as the Markov model to be approximated. The states of a Markov model that are at the same transition level are all approximated by the state of the chain model that is at that same transition level.

Once the transition levels are set, the transition rates must be determined. Depending on the purpose of the chain model, different sets of transition rates may be used. Two examples to provide bounds on the probability on a certain state have already been discussed. Another possibility is to use the fastest transition rate in the original model. This is often a good first approximation. Additionally, it has the advantages that the maximum transition rate is easily determined and that the solution of such a chain model is easily computed. However, it may not be conservative. For example, suppose there are two transition rates leaving the first state. Also suppose one of the transition rates (λ) is two orders of magnitude greater than the other and that the faster transition rate is the maximum rate for the model. In this case, the probability of S_1 is

$$P_1(t) = e^{-1.01(\lambda t)}$$

The chain model would give

$$P_1(t) = e^{-(\lambda t)}$$

Clearly, the chain model does not give a conservative estimate on the probability of S_1. However, it does give a reasonably good estimate.

On the other hand, if the two transitions leaving S_1 are similar in magnitude, we get the following probabilities for the actual model and the chain model, respectively

$$P_1(t) = e^{-2(\lambda t)}$$

$$P_1(t) = e^{-(\lambda t)}$$

In this case, the chain model may not be a very good approximation to that system as the argument of the exponential in the chain model may be several times the argument of the exponential in the original model. This approach to the selection of the transition rates may be used if the transition rates vary significantly in magnitude. Otherwise, an alternative method is suggested.

This latter case suggests an alternative method. The second choice of the transitions is to sum all the transitions leaving any given transition level. For the transitions exiting the zero failure level f=0 (assuming only one state at this level), this summation could be

$$\lambda_{f=0} = \sum_i \lambda_{1i}$$

The maximum transition rate for the chain model would then be the maximum of the sum of these exiting transitions.

$$\lambda = \max_j (\lambda_{f=j})$$

In this manner, the chain model can account for transition rates that are similar in magnitude by adding them together. Like the first method, the transition rate for this method can be easily determined. Also, the solution of such a chain model is again easily computed. This technique, however, can give misleading values in lower transition level states. For example, if the transitions for the third failure level yield the fastest transition sum, the chain model may give approximations for the probability of the zero, first and second transition levels that are too low. Nevertheless, this method does give good estimates in general.

A third technique uses a progressive selection of transition rates. By progressive is meant that the fastest transition rate at each transition level is used unless a faster transition rate was used to arrive at the current transition level. In that case, the faster transition level would still be used. For example, if the fastest transition rate at the first level is 10^{-5} hr^{-1} and the fastest rate at the second level is 10^{-4} hr^{-1}, the 10^{-5} hr^{-1} would be used as the transition rate for the first level, and the 10^{-4} hr^{-1} would be used as the transition rate for the second level. However, if the fastest transition for the third level is 10^{-6} hr^{-1}, the 10^{-4}

hr^{-1} would continue to be used for the third transition level. It is also possible to combine the progressive method and the second technique of summing exiting transitions at failure levels to determine the transitions for the chain model. Because the later transition levels are dependent on the probabilities in the lower transition levels, a progressive technique should give better approximations.

The third technique has the advantage of being the most accurate of the methods discussed. However, as is often the case, the most accurate method is also the most cumbersome. The solution to the chain model is no longer as simple because there may be different transitions at each transition level. In many cases, the additional accuracy gained by using the third technique is not worth the extra computation necessary. As such, most of this paper will use the second technique--a single transition rate determined as the sum of the transitions--to determine the transition rates for the chain model.

2.6.2 The Cyclical Chain Model

The cyclical chain model is similar to the non-cyclical chain model in its base construction. It also has an equivalent number of transition levels as the Markov model it is approximating. However, the selection of the transition rates is more complicated. Because of the return transition rates, we can no longer be sure of the direction of the change of probability. A non-cyclical model is always pure-decay. A cyclical model, on the other hand, has probability returning to states from where it came. This causes problems in the selection of the transition rates.

If we have a certain state in mind, we can again obtain upper and lower bounds similar to the non-cyclical case. For a lower bound on S_i, all transitions that may bring probability into S_i will be λ_{min}, while all transitions that may bring probability away from S_i will be λ_{max}. This results in the following four-state chain model with the lower bound being imposed on S_3.

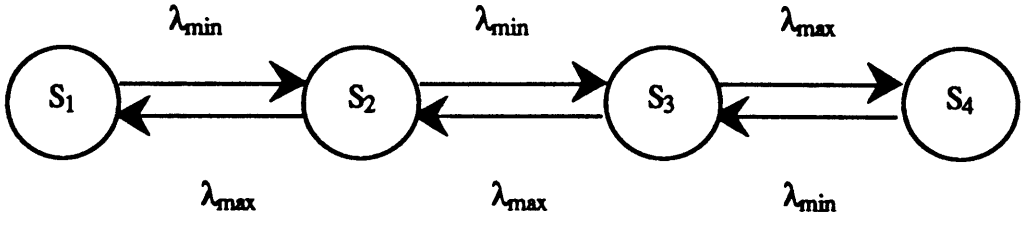


Figure 2-7: Cyclical Chain Model for Lower Limit of State Probability

For an upper bound on S_i, the opposite scenario is necessary. All transitions that may bring probability into S_i will be λ_{max}, while all transitions that may bring probability away from S_i will be λ_{min}. An upper bound on S_3 is then given by the following model

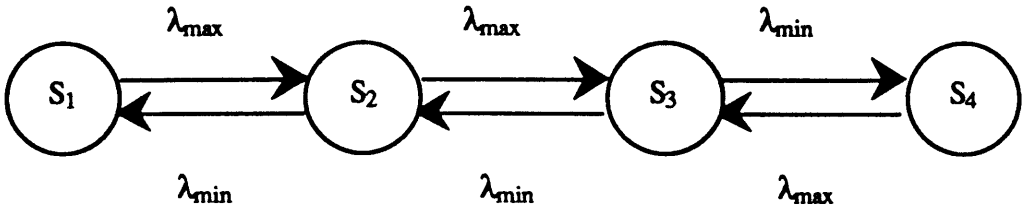


Figure 2-8: Cyclical Chain Model for Upper Limit of State Probability

Unfortunately, for a general case, the methods used to determine the transitions for a non-cyclical model are inapplicable. If we assign the forward transition rates as λ_{min} and the return transitions as λ_{max}, the probabilities are unequally distributed, being lopsided to the lower-transition states. If we assign the forward transition rates as λ_{max} and the return transitions as λ_{min}, the probabilities are unfairly lopsided to the higher transition states. There is simply too much flexibility in a cyclical Markov model to effectively choose the proper forward and return transitions for the cyclical chain model for the general case.

2.6.3 The Characteristic Time for the Chain Model

As was stated earlier, the characteristic time of a Markov model is important to the analysis of Markov models. For a two-state model, the characteristic time t_c of the system was defined to be the point in time equal to the reciprocal of the transition rate.

$$t_c = \frac{1}{\lambda}$$

This is also the point in time where approximately one-half of the probability is no longer in state 1. It was also stated that the characteristic time is difficult to define and describe for a general case. Once again, we turn to the chain model to approximately define the characteristic time for a general model.

For a non-cyclical chain model that has more than two states, we define a characteristic time for each of the states in the model that has an exiting transition rate. Thus for an f-state chain model, we have (f-1) characteristic times, one for every state except the trapping state. The characteristic time for the first state of a n-state chain model is defined as the reciprocal of the transition rate leaving that state. For the second state, we

cannot define the characteristic time to simply be the reciprocal of the transition rate leaving that state. This definition fails to account for the dependence of the second state probability on the first state. This dependence is accounted for by summing the reciprocals of the two transition rates leaving the first and second states. Thus the characteristic time for the second state is given by

$$t_{c2} = \frac{1}{\lambda_{12}} + \frac{1}{\lambda_{23}}$$

We can extend the characteristic time for state 2 to develop the characteristic time for all the states in a general f-state chain model. For each state in the model, the characteristic time is the sum of the reciprocal transition rates necessary to arrive and exit that state.

$$t_{cf} = \sum_{i=1}^{f-1} \frac{1}{\lambda_{i\,i+1}}$$

Thus, we have a characteristic time for each state except the trapping state in a chain model.

Most often, the interesting situations occur when any characteristic time is first reached. This will be demonstrated in Chapter 4 when discussing the bounding of the roundoff error. The first characteristic time is reached at the reciprocal of the transition rate exiting the first state. Thus, the characteristic time of the first state will be used as the characteristic time of the system in all further analysis.

$$t_c = \frac{1}{\lambda_{12}}$$

The characteristic time will become very important for defining certain properties of both the roundoff error and the integration error propagation patterns.

2.7 An Example of the Use of Markov Models in Reliability Analysis

The Markov model is a means to calculate the probability of a system being in a certain state as a function of time. As such, Markov models are ideally suited for determining the reliability of a system where the probability of being in certain 'states of failure', or failure states, as a function of time is the desired quantity.

Most types of system components can have their failure patterns described by what is commonly called the bathtub curve. The first portion of the curve is known as the infant mortality section. In this section the rate of failure is high but steadily decreasing. These

failures are usually due to defects in the components. The second portion of the curve is a long, steady period considered the normal working point of most systems. It is usually characterized by a low and constant failure rate. This means that failures occur at random times and at a uniform rate without any obvious pattern. The last area of the curve is due to the aging effects of the components. In this region, the failure rates steadily increase because the components begin to wear out.

The normal working point of the system may be simulated using Markov models. The transitions are constant failure rates and the states of the system are states of failure. The constant failure rate implies that the probability of failure of a component is independent of age (time), so continuous-transition Markov models described above are applicable. The discrete-transition Markov models may be used if the component failures are given as probabilities for a certain period of time. As the discrete method of reliability analysis is less common, a continuous-transition Markov model will be used for this illustration.

As the example, consider a two-component system connected in parallel as shown in the figure below.

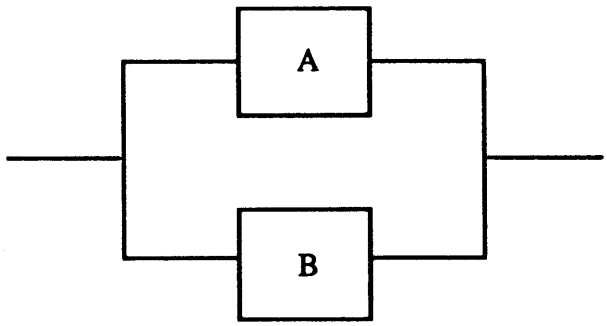


Figure 2-9: Block Diagram for Two-Component Parallel System

Component A has an associated constant failure rate of a, and Component B has an associated constant failure rate b. In order for the system to be considered operational there must exist at least one operational path through the components. We can therefore see that there are five states of failure:

State 1: Both A and B are operational; system operational
State 2: A has failed; only B is operational; system operational
State 3: B has failed; only A is operational; system operational
State 4: A has failed and then B has failed; no operational path; system loss
State 5: B has failed and then A has failed; no operational path; system loss

If one is not concerned about the order in which failures occur, State 4 and State 5 can be combined into one new state.

State 4: Both A and B have failed; no operational path; system loss

The above two-component system with states 4 and 5 combined is described by the following Markov model.

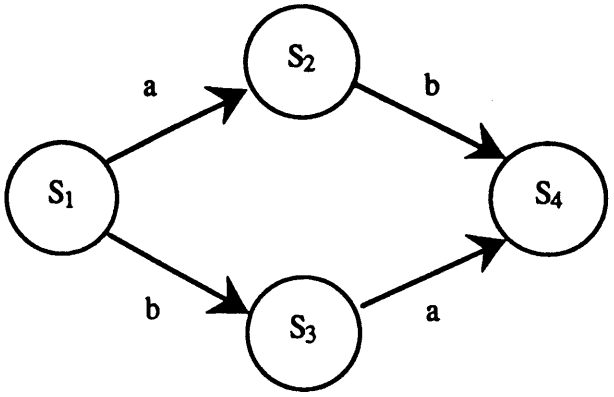


Figure 2-10: Markov Model for Two-Component Parallel System

The continuous-transition matrix A for the above Markov model is given below. Notice that the columns of the matrix sum to zero in accordance with the conservation of system probability.

$$\frac{dP(t)}{dt} = \begin{bmatrix} -(a+b) & 0 & 0 & 0 \\ a & -b & 0 & 0 \\ b & 0 & -a & 0 \\ 0 & b & a & 0 \end{bmatrix} P(t)$$

One can clearly see the composition of the transition matrix through the concept of probability flow. The total transition rate leaving a state is placed on that state's associated diagonal. For example, the total flow leaving S_1 is (a+b). This flow rate is placed on the first diagonal with the associated negative sign indicating that the direction of the probability flow is outward. Notice that the fourth diagonal is zero indicating that the fourth state is a trapping state with no exiting transitions. For a non-cyclical model, only states with zero on the associated diagonal will have non-zero steady-state probabilities.

The solution of the probabilities of the above system is given by the matrix exponential.

$$P(t) = e^{At}$$

Using Laplace transforms or another similar method for the solution of the differential equations, the probabilities can be show to be

$$\begin{aligned}
P_1(t) &= e^{-(a+b)t} \\
P_2(t) &= e^{-bt} - e^{-(a+b)t} \\
P_3(t) &= e^{-at} - e^{-(a+b)t} \\
P_4(t) &= 1-e^{-bt} - \frac{b}{a+b}(1-e^{-(a+b)t}) \\
&\quad + 1-e^{-at} - \frac{a}{a+b}(1-e^{-(a+b)t})
\end{aligned}$$

For the above system, the solution via Laplace transforms was relatively straight forward. However, when systems become more complex, the transform solution method becomes unmanageable and other techniques are employed. These techniques will be discussed further in Chapter 3.

The reliability of the system is the probability of being in an operational state at the given final time. In the above example, the reliability of the system is given by the sum of the probability of states 1, 2 and 3. Using the conservation of system probability, the reliability of the system is also the probability of state 4 subtracted from unity. The unreliability of the system is given by one minus the reliability of the system, or in this example the probability of state 4.

This system suggests two possible chain models. First, we may want to do a general chain model to provide estimates to the state probabilities. Secondly, we may want to provide an upper bound on the probability of state 4 since this state gives the unreliability of the system. In the first case, we will use the sum of the transition rates for the transition rate of the chain model. This yields the following model.

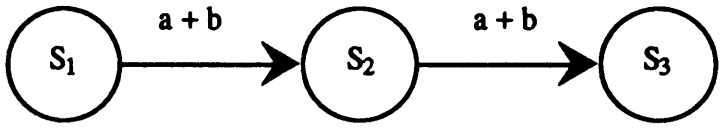


Figure 2-11: Chain Model Approximation for Two-Component Parallel System

Using the formulas for the state probabilities of a chain model, the estimated probabilities of the original model are

$$P_1(t) = e^{-(a+b)t}$$

$$P_2(t) = P_3(t) = (a+b)t\, e^{-(a+b)t}$$

$$P_4(t) = 1 - [P_1(t) + P_2(t)]$$

$$= 1 - [1 + (a+b)t]\, e^{-(a+b)t}$$

If we wish to bound the probability of state 4 we would use the following chain model. Assuming a > b, the chain model would be

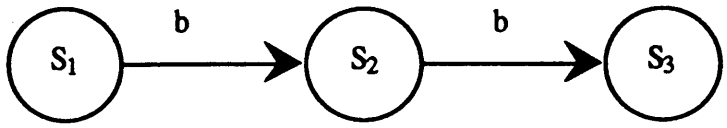


Figure 2-12: Chain Model Approximation for Bounding of State 4 for Two-Component Parallel System

This model gives the following bound on the probability of state 4

$$P_4(t) \leq 1 - [P_1(t) + P_2(t)]$$

$$\leq 1 - [1 + bt]\, e^{-bt}$$

Many physical systems have an added complexity that they are capable of being repaired. This implies that once a component is failed, it can be brought back to working order. Thus, if a failure of a component corresponds to a forward transition in a Markov model, the repair of a component corresponds to a reverse transition in a Markov model.

Suppose now that both Component A and Component B can be repaired with repair rates A and B, respectively. If we assume that the system can be repaired from the system loss state (S_4), the configuration is represented by the following cyclical Markov model.

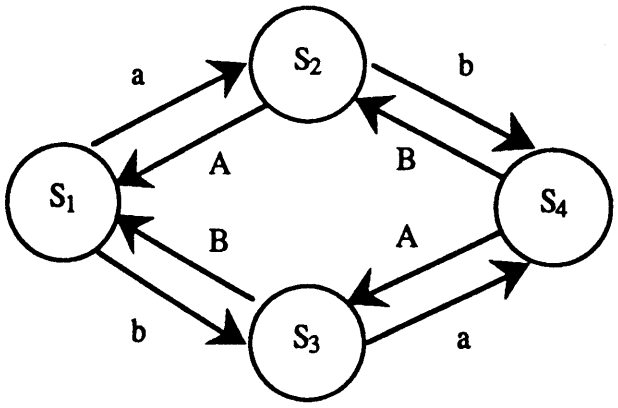


Figure 2-13: Cyclical Markov Model for Two-Component Parallel System with Repairs

The continuous-transition matrix A for the cyclical model is given below. Notice again that the columns of the transition matrix A sum to zero in accordance with the conservation of system probability.

$$\frac{dP(t)}{dt} = \begin{bmatrix} -(a+b) & A & B & 0 \\ a & -(A+b) & 0 & B \\ b & 0 & -(a+B) & A \\ 0 & b & a & -(A+B) \end{bmatrix} P(t)$$

Again, one can clearly see the composition of the transition matrix through the concept of probability flow. The total transition rate leaving a state is placed on that state's associated diagonal. Now each state has two exiting transitions as evidenced by the two terms on each of the diagonals. Notice that the fourth column is no longer a column of zeros, therefore it is no longer a trapping state. Also notice that the transition matrix is no longer lower triangular, a sign of a cyclical Markov model.

The solution of the probabilities of the above system is again given by the matrix exponential

$$P(t) = e^{At}$$

However, due to the repair transitions, the closed-form solution of the above system is quite complex and not very enlightening.

Again we can use a chain model to approximate the actual Markov model. As was stated earlier, it is most effective to use the chain model to bound a state probability. We

will use the chain model to bound the probability of state 4. Assuming a > b, we get the following model.

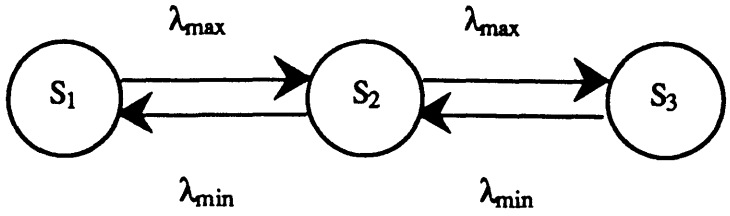


Figure 2-14: Cyclical Chain Model to Bound State 4 of the Two-Component Parallel System with Repairs

This model would then need to be solved using a method to be discussed in Chapter 3 to give the actual bound for the probability of state 4. Obviously not much is gained by approximating this small system with a chain model. However, it may turn out to be very advantageous for larger models.

Because many systems can get much more complicated than the simple two-component system here, alternative solution techniques for the matrix exponential are warranted. Chapter 3 is dedicated to discussing various techniques for the solution of the matrix exponential via power series methods.

2.8 Recapitulation

In conclusion, the Markov process is a stochastic process that is characterized by dependent trials. Markov processes possess a limited memory in that the present state of the system is dependent on only the previous state of the system and not on how the system arrived in that previous states. This can be thought of as the 'future' of the Markov process is dependent only on the 'present' state of the system and not on its 'past' history.

The Markov process, in turn, can be represented by two types of Markov models--discrete-state, discrete-transition models and discrete-state, continuous-transition models. A discrete-transition model is representative of a set of difference equations that describe the probabilities of the states of the system. Because we have discrete transitions, the system is only allowed to make a change of state of the system at discrete intervals of time called trials. The probability of being in a certain state of the system is then given after a certain trial, $P_i(n)$, where n is an integer corresponding to the trial number. The probability of changing state after the n^{th} trial is given by the discrete-transition probability p_{ij}. The

Markov model may be characterized by a discrete-transition matrix K made up of the transition probabilities. This matrix may then be propagated (by stepping) to the desired trial number to obtain the state probabilities.

On the other hand, the continuous-transition Markov model produces a set of differential equations that describe the state probabilities as functions of time. Unlike the discrete-transition model, the continuous-transition model may change state at any instance in time. The probability of being in a certain state is now determined by time instead of trial number, $P_i(t)$. The transitions are given as rates, rather than as probabilities, where λ_{ij} is the rate of going from S_i to S_j. The continuous-transition Markov model may be characterized by the continuous-transition matrix A consisting of the transition rates. The solution to the probabilities of the continuous-transition matrix is the matrix exponential e^{At}.

Markov models may be classified into two types, cyclical and non-cyclical. Cyclical models allow the system to return to a state once the system has left it. Non-cyclical models, however, do not allow the system to return to a state. Non-cyclical models are also known as pure-decay models.

There are also some properties that are inherent to Markov models. One of these, the conservation of system probability, states that the probability of all the states in the model must sum to unity at all times and trials. Another, the characteristic time, determines the point at which the product of the transition rate and the time is unity. Both of these characteristics will be instrumental in predicting and controlling the error propagation patterns.

Markov chain models are models with only one state per transition level. This restriction simplifies the understanding and the solution of the Markov model. For these reasons, chain models will be used to approximate, and at times bound, the error propagation patterns for more complicated Markov models.

The state probabilities of the Markov model are given by the matrix exponential e^{At}, where A is the transition matrix. In some simple cases, the direct, closed-form solution of the matrix exponential may be possible. For cases with a large state vector or for cases with return transitions, the closed-form solution of the matrix exponential is cumbersome. Thus, Chapter 3 gives some techniques for solving the matrix exponential. These techniques are based on power series solutions and are readily implemented on a computer.

Chapter 3

The Computation of the Matrix Exponential

Chapter 2 discussed the Markov model and the Markov process. It showed that a continuous-transition Markov model lead to a set of differential equations, the solution of which is the matrix exponential. The purpose of this chapter is to discuss possible method of solving for the matrix exponential. Because of the simplicity, accuracy, and ease of implementation, the two power series methods of Taylor and Padé will be discussed. Additionally, methods that improve the basic Taylor and Padé approaches are investigated. Lastly, the amount of computer work associated with each method is identified. Computer work is one of the parameters necessary to compare the efficiency of the solution methodologies.

3.1 The Matrix Exponential

Consider a two-state non-cyclical chain Markov model with transition rate a as shown below

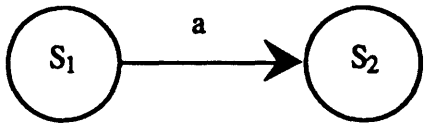


Figure 3-1: Two-State Non-Cyclical Markov Model

Using the notion of probability flow discussed in Section 2.3, the equation for the probability of state 1 can be written from inspection as

$$\frac{dP_1(t)}{dt} = - a P_1(t)$$

The solution of this linear, constant coefficient ordinary differential equation is given by the exponential function.

$$P_1(t) = e^{-at} P(0)$$

where P(0) is the initial condition.

Now consider a general Markov model. For the general Markov model, we have a system of differential equations rather than a single equation, so matrix theory needs to be

applied. It was shown in Chapter 2 that a continuous-transition Markov model results in a set of differential equations of the form

$$\begin{bmatrix} \frac{dP_1(t)}{dt} \\ \frac{dP_2(t)}{dt} \\ \cdots \\ \frac{dP_m(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\sum_j \lambda_{1j} & \lambda_{12} & \cdots & \lambda_{1m} \\ \lambda_{21} & -\sum_j \lambda_{2j} & \cdots & \lambda_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{m1} & \lambda_{m2} & \cdots & -\sum_j \lambda_{mj} \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ \cdots \\ P_m(t) \end{bmatrix}$$

Defining the above matrix to be the transition matrix A, this equation can alternatively be written as

$$\frac{dP(t)}{dt} = A\ P(t)$$

By appealing to the scalar analogy, the solution to the above equation is again the exponential function. In this case, however, the constant coefficient of the exponential function is now a matrix made up of constant coefficient components. The solution of this set of differential equations is called the matrix exponential.

$$P(t) = e^{At} P(0)$$

The exponential function in the scalar case can be defined by its convergent infinite Taylor series

$$e^{at} = 1 + at + \frac{(at)^2}{2!} + \frac{(at)^3}{3!} + \ldots + \frac{(at)^k}{k!} + \ldots$$

In another analogy to the scalar case, the matrix exponential can be formally defined by its convergent infinite Taylor series

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \ldots + \frac{(At)^k}{k!} + \ldots$$

where I is the identity matrix. The focus of this chapter is to find an accurate yet inexpensive method to compute the matrix exponential.

Many methods have been suggested for the solution of the matrix exponential. Because of the simplicity of implementation, we will concentrate on power series solutions.

Moler and Van Loan [Moler and Van Loan, 1978] have found that Taylor and Padé power series have some desirable qualities in the solution of the matrix exponential. Used properly, these power series have been shown to be two of the most accurate and efficient methods known. This chapter will therefore concentrate on these power series solutions to compute the exponential of the matrix.

3.2 Taylor Series Approximations

The Taylor series for the matrix exponential was given above to be the definition That is, the Taylor series for the matrix exponential, denoted by T(At), is

$$T(At) = e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \ldots + \frac{(At)^k}{k!} + \ldots$$

The exponential function is irrational. Therefore, the definition of the matrix exponential must be an infinite series. Since we would like to use computers to calculate the matrix exponential, we will only be able to approximate the matrix exponential with a truncated Taylor series of k terms. A truncated Taylor series will be denoted by $T_k(At)$, where the subscript k will represent the highest power of the matrix At in the truncated series.

$$e^{At} \approx T_k(At) = \sum_{i=0}^{k} \frac{(At)^i}{i!}$$

So as to allow comparison of possible methods, we will define the the order of approximation. The order of the approximation is equivalent to the number of derivatives of the function that are matched by the approximation. For the Taylor series, the order of the approximation is seen to be the highest power of the truncated Taylor series, and will be denoted by k. This quantity will be used to compare methods that are approximating the same function. Comparisons cannot be made for approximations of different functions. Also, the order of the approximation is only a gauge of the accuracy of the approximation. There are many other factors that can affect the accuracy of a solution technique. Lastly, the order of the approximation does not necessarily reflect the amount of work required to obtain the approximation.

The number of terms necessary to achieve an accurate estimate for the matrix exponential is dependent on the values for A and t. It is convenient when discussing the values of the components of a matrix to introduce the concept of a matrix norm. The matrix norm of (At) is denoted as $\|At\|$. There are three common matrix norms, known as the 1-, 2- and infinity norms. These norms are defined as follows.

$$||At||_1 = \max_j \sum_{i=1}^{m} |(At)_{ij}|$$

$$||At||_2 = \max_i |\sigma_i|$$

$$||At||_\infty = \max_i \sum_{j=1}^{m} |(At)_{ij}|$$

where $(At)_{ij}$ represents the element in the ith row and jth column of matrix At, and σ_i represents the i^{th} characteristic value of At. The matrix norm gives an idea of the size of the elements of the matrix. It also gives a measure on the amplification power of the matrix when multiplied by a vector or another matrix. Thus, the matrix norm will be used to give a measure on the magnitude of the components of the transition matrix A. Any of the above definitions is acceptable for our purposes. But because of the ease of calculation, the 1-norm will be used unless otherwise noted.

There are a few properties of the matrix norm that are important to recognize. We list these properties without proof.

1) $\|\|At\|\| = \|\|A\|\| \, \|t\|$	Scalar multiplicative
2) $\|\|A_1 + A_2\|\| \leq \|\|A_1\|\| + \|\|A_2\|\|$	Triangle inequality
3) $\|\|A_1 \, A_2\|\| \leq \|\|A_1\|\| \, \|\|A_2\|\|$	Matrix multiplicative

These properties will be used in determining conditions for using various solution techniques to compute the matrix exponential. They will also be used in determining bounds on certain error propagation patterns.

If the norm of (At) is less than unity ($\|At\| < 1$), then all the terms of the product (At) are less than unity. In this case, the ensuing terms of $T_k(At)$ decrease in magnitude because they are raised to increasingly higher powers. Therefore, the absolute error of the approximation decreases monotonically for each additional term in the series. For $\|At\| < 1$, the number of terms required to achieve reasonable accuracy is relatively small.

On the other hand, if the norm of (At) is greater than unity ($\|At\| > 1$) some terms of the product (At) may be greater than unity. In this case, the ensuing terms of $T_k(At)$ may increase in magnitude before they decrease in magnitude. The magnitudes of the powers of (At) in the numerators initially outpace the factorials of these powers in the denominators.

Eventually, the magnitudes of the factorials surpass the powers of (At) and the terms begin to decrease in magnitude. A large number of terms may therefore be necessary before the approximation gives reasonable accuracy.

Additionally, this method may suffer from severe cancellation if implemented on a digital computer. For Markov models, two consecutive terms of the series will have opposite signs. Yet these terms may have similar magnitudes greater than the precision of the computer. As such, the summation of these terms will result in severe cancellation. Lastly, the number of computer operations necessary to compute the matrix exponential given the large number of terms in the series clearly leads to a highly inefficient algorithm.

Given below is a table for the relative errors of the scalar exponential. Two cases are presented; first, the coefficient is less than unity and second, the coefficient is greater than unity. The terms in the series are counted including the initial unity value. That is, 1+at counts as two terms in the series. Similar trends can be expected for the solution to the matrix exponential.

Table 3-1: Relative Error for the Scalar Exponential (e^{at})

Terms in Series	at = 0.1	at = 10
1	0.1051709	2.202546×10^{4}
2	0.0053462	1.982391×10^{5}
3	0.000179681	9.030840×10^{5}
4	0.000004514	2.767993×10^{6}
5	0.000000091	6.409700×10^{6}
10	$< 2.755731 \times 10^{-17}$	3.112658×10^{7}
15	$< 7.647163 \times 10^{-28}$	1.026922×10^{7}
20	$< 4.110317 \times 10^{-39}$	6.102730×10^{5}
30	$< 3.769987 \times 10^{-63}$	6.266462×10^{1}

The table implies that the truncated Taylor series can be yield an accurate estimate with a reasonable amount of work for cases where the terms of (At) are small. In fact, depending on the elements of the transition matrix, the truncated Taylor series may be the simplest and fastest way to compute the matrix exponential to achieve a desired accuracy.

Unfortunately, the table also implies that the truncated Taylor series may require a very large amount of work and still not yield an acceptable accuracy. Clearly in the second

case, the truncated series is an unreliable and inefficient method to compute the matrix exponential. Additionally, theses examples leave open the question of where the series should be truncated. The first example could be truncated after five terms to give an approximation good to seven significant digits. However, the second case does not reach an acceptable approximation even after thirty terms in the truncated series. For the general case, another method is undoubtedly warranted.

3.3 Padé Series Approximations

Another type of power series that can be used to approximate the matrix exponential is the Padé series. The Padé series is a rational fraction approximation R(x) to a function f(x). This ratio of two polynomials is given in the form

$$R_{pq}(x) = \frac{N_p(x)}{D_q(x)}$$

where

$$N_p(x) = \sum_{i=0}^{p} A_i x^i$$

$$D_q(x) = \sum_{i=0}^{q} B_i x^i$$

The integer pair (p,q) gives the degree of the numerator polynomial and the degree of the denominator polynomial, respectively. The coefficient terms A_k and B_k are chosen so that the first p+q derivatives of the Padé series match the first p+q derivatives of the given function at a designated point. For convenience and standardization, this point is chosen to be zero.

$$R(0) = f(0);\ R'(0) = f'(0);\ \ldots;\ R^{(p+q)}(0) = f^{(p+q)}(0)$$

The full (p,q) array of Padé approximations, known as the Padé table for the given function f, is shown below

$$\begin{array}{cccc} R_{00} & R_{10} & R_{20} & \cdots \\ R_{01} & R_{11} & R_{21} & \cdots \\ R_{02} & R_{12} & R_{22} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{array}$$

provided that all the entries exist. We can see that the top row of the table is the set of Taylor series approximations, T(At). We can also see that the first column is the set of reciprocal Taylor series approximations, 1/T(At).

For the exponential function e^x, the numerator and denominator of the Padé approximation are given by

$$N_p(x) = \sum_{i=0}^{p} \frac{p!\,(p+q-i)!}{(p+q)!\,i!\,(p-i)!} x^i$$

$$D_q(x) = \sum_{i=0}^{q} \frac{q!\,(p+q-i)!}{(p+q)!\,i!\,(q-i)!} (-x)^i$$

Using the above coefficients, the rational fraction $R_{pq}(x) = \frac{N_p(x)}{D_q(x)}$ matches the first p+q derivatives of the exponential function. Clearly then, the order of the Padé approximation is p+q. Because of the nature of the Taylor series, this is equivalent to saying the Padé approximation matches the first p+q terms of the Taylor series for e^x.

We again appeal to the analogy of the scalar case to determine the approximation for the matrix exponential. Applying the matrix theory notion of the matrix inverse, the Padé approximation for the matrix exponential e^{At} is given as

$$R_{pq}(At) = [D_q(At)]^{-1} N_p(At)$$

with

$$N_p(At) = \sum_{i=0}^{p} \frac{p!\,(p+q-i)!}{(p+q)!\,i!\,(p-i)!} (At)^i$$

$$D_q(At) = \sum_{i=0}^{q} \frac{q!\,(p+q-i)!}{(p+q)!\,i!\,(q-i)!} (-At)^i$$

Non-singularity of $D_q(At)$ is assured if p and q are large enough, or if the eigenvalues of (At) are negative [Moler and Van Loan, 1978].

Diagonal Padé approximations are those approximations that have equal degrees on the numerator and the denominator, p = q. They derive their name in that they are on the main diagonal of the Padé table. For example, the third diagonal Padé approximation for e^x is given below

$$R_{33}(x) = \frac{1 + \frac{x}{2} + \frac{x^2}{10} + \frac{x^3}{120}}{1 - \frac{x}{2} + \frac{x^2}{10} - \frac{x^3}{120}}$$

The diagonal approximations are significant because they are the most efficient Padé approximation in terms of the amount of work versus the accuracy obtained. It can be

easily seen that they yield the highest order of approximation for a given power of x. Section 3.5 deals with this concern in more detail.

Padé series approximations suffer from some of the same problems as the Taylor series approximations. The number of terms needed to achieve an accurate estimate is again dependent on the product (At) and may be prohibitively large as ||At|| becomes large. Additionally, cancellation in a similar manner to the Taylor approximations may occur if ||At|| is greater than unity.

One disadvantage that is unique to the Padé series approximations is the need for a matrix inversion. Moreover, the denominator matrix D_q (At) may be poorly conditioned with respect to the inversion process. This poor conditioning occurs particularly when (At) has widely spread eigenvalues [Moler and Van Loan, 1978]. In many cases within reliability analysis, the transition rates may differ by orders of magnitude (especially if repairs are being made), resulting in widely spread eigenvalues and a poorly conditioned denominator matrix. The possibility of a poorly conditioned inverse can make the numerical results of the matrix exponential suspect.

Another disadvantage of the Padé approximation is the limited freedom to add terms to the series. If it is desired to add a term to the Taylor series, the additional term is simply added on to the current total. Thus, Taylor approximations allow the freedom to change the order of the approximation with relative ease. With Padé approximations, however, the coefficients for each term in the numerator and the denominator need to be recalculated. New numerator matrices and new denominator matrices must be created if we want to add an additional term to either the numerator or the denominator. This lack of flexibility requires that more *a priori* knowledge be known with greater confidence to insure that the proper order of the approximation is used.

Nevertheless, the diagonal Padé approximations have some advantages when compared to the Taylor series. The most important advantage is that the Padé approximations converge more rapidly than the Taylor approximations. The order of the approximation is given by the number of derivatives of the exponential function that are matched by the approximation. For the Taylor series, this is denoted by k as above. For the Padé series this is given as p+q. For diagonal Padé approximations the order of the approximation is 2p. Therefore, for the same powers of the product (At), the Padé approximations can have an order twice as large as the Taylor approximations.

Overall, Padé series approximations suffer from the same problems as Taylor series approximations. They have the same problems with cancellation and convergence for norms of (At) greater than unity. Additionally, the Padé approximations require a matrix inversion that causes uncertainty in the process and may lead to inaccurate results. On the other hand, the Padé approximations have the advantage that they have an order of approximation that can be as much as twice the order of approximation for the Taylor series for an equivalent number of terms in the power series, although the amount of computer work required may not be equivalent. In the same manner as the Taylor approximations, the Padé approximations are applicable to certain cases, but they fail as a method for the calculation of the matrix exponential for general systems. Another method is still warranted.

3.4 Scaling and Propagating Methods

The problems of cancellation and convergence can be remedied with an integration method known as scaling and propagating. Scaling and propagating exploits a fundamental identity of the exponential function and the time invariance of the matrix A.

$$e^{At} = (e^{At/n})^n$$

Now either Taylor series or Padé approximations can be used to calculate the quantity inside the brackets ($e^{At/n}$), the scaled matrix exponential. This quantity can then be raised to the appropriate power n to obtain the desired e^{At}. The advantage of the scaling methods is that the scaled transition matrix can be made to have a norm less than unity, thus avoiding the problems of cancellation and convergence. Many authors have suggested this method or variations on it [Ward, 1977; Shah, 1971].

3.4.1 Scaling of the Base Matrix

We will define the matrix M to be the base matrix, or the scaled matrix exponential.

$$M = e^{(At/n)}$$

We will also define the scaled time to be called the time step, Δt. Note that the scaling factor multiplied with the time step yields the final time. In addition note that the scaling factor is also the number of time steps required to reach the final time.

$$n\Delta t = t$$

Thus, we are now calculating the matrix exponential for the scaled time Δt

$$M = e^{A\Delta t}$$

Using the Taylor series definition of the exponential function, we can see that the base matrix M can be defined as

$$M = I + A\Delta t + \frac{(A\Delta t)^2}{2} + \ldots + \frac{(A\Delta t)^k}{k} + \ldots$$

Again, it is apparent that the base matrix must be approximated by the truncated power series. We will denote the degree of Taylor approximation to the base matrix by the subscript k, M_k. We can also use a Padé approximation to determine the base matrix. The degree of the Padé approximation will be denoted with two subscripts pq, M_{pq}. For the diagonal Padé approximations, this will be denoted as M_{pp}.

The time step should be chosen to insure that the base matrix norm is less than unity, $\|At/n\| = \|A\Delta t\| < 1$. This criterion guarantees quick convergence of the approximation of the base matrix. Quick convergence allows for a lower degree of approximation than would otherwise be necessary. One rule of thumb that has been suggested [Moler and van Loan, 1978] is to make n the smallest power of two for which $\|At\| / n < 1$. While this selection meets the norm-less-than-unity criteria, it will be shown in Chapter 8 that this may not be the most efficient choice.

3.4.2 The Stepping Algorithm

Once the base matrix has been determined, it must be propagated through time to obtain e^{at}. There are two basic types of scaling and propagating routines. The simplest type is the stepping algorithm. Either a Taylor or Padé approximation is used to determine the base matrix ($e^{At/n}$). The state vector is then 'stepped' through time via the base matrix to obtain the state vector at the final time. The alternative method is a squaring algorithm. This method also uses either a Taylor or Padé approximation to determine the base matrix. However, the base matrix is now 'squared' through time to obtain the matrix exponential at the final time $(e^{At/n})^n$.

For the stepping algorithm, the state vector P(t) is stepped through time via the base matrix to obtain a new state vector. This process is continued until the final time is reached and the final state vector has been determined. We start with the base matrix and the initial conditions. Multiplying these together, we arrive at the new state vector for the time Δt.

$$P(\Delta t) = M\ P(0)$$

If we then treat the state vector $P(\Delta t)$ to be the new initial conditions, we can continue to step through time repeatedly defining the matrix exponential problem over the time step Δt. Thus, at the end of each step a new state vector is calculated. Continuing the algorithm, we get

$$P(2\Delta t) = M\ P(\Delta t)$$

$$P(3\Delta t) = M\ P(2\Delta t)$$

$$\vdots$$

$$P(n\Delta t) = M\ P((n-1)\Delta t)$$

The stepping methodology is a simple, iterative approach that travels linearly through time. It operates on the state vector while the base matrix remains unchanged.

Primarily, there are two instances that the stepping algorithm is desirable. If it is appealing to have a linear time history of the state probabilities of the Markov model, the stepping algorithm can easily give intermediate state probabilities at multiple intervals of the time step Δt. The desired intervals for the time history should be taken into consideration when choosing the time step.

The second reason for using a stepping algorithm addresses memory considerations. Depending on the size of the Markov model, the base matrix M can become large. M is an (mxm) matrix, where m is the number of states in the Markov model. Often, the matrix M is very sparse and can be stored effectively. The stepping algorithm allows the matrix M to remain sparse by only operating on the state vectors. If, on the other hand, the matrix M were operated on as in the squaring method, it would begin to fill up. If m is very large, memory problems may result. A stepping algorithm may avoid this problem.

3.4.3 The Squaring Algorithm

An alternative to the stepping algorithm is the squaring algorithm. Instead of operating on the state vector, the base matrix is now operated on directly to calculate the transition matrix at the final time. The transition matrix is squared to travel logarithmically

through time. Because of this logarithmic travel, the squaring method can be much more efficient that the stepping algorithm.

The fundamental equation for the scaling and propagation methods is

$$e^{At} = (e^{At/n})^n$$

Written in state vector form this is

$$P(t) = (e^{At/n})^n \, P(0)$$

Substituting in the base matrix gives

$$P(t) = M^n \, P(0)$$

Clearly, we can operate on the base matrix to calculate M^n in order to determine the state probability vector.

The determination of M^n depends on the integer n. The most effective method is to choose n to be an integer power of two. If $n = 2^\alpha$, we can square the base matrix α times until the final time is reached.

$$P(t) = \underbrace{\left[\left[\left[\left[M^2\right]^2\right]^2\right]\cdots\right]^2}_{\alpha \text{ times}} \, P(0)$$

In this manner only α matrix multiplications are necessary instead of the n matrix multiplications necessary for direct calculation.

For example, if a time step of one and a final time of 100 are used, M^n can be determined by the following two ways

$$M^{100} = \underbrace{M\,M \ldots M}_{100 \text{ times}}$$

$$M^{100} = M^{64}\,M^{32}\,M^4\,M^2$$

If, however, the time step is redefined to be 0.78125, the scaling factor becomes 128. The final transition matrix can then be determined by only seven squarings of the base matrix ($2^7 = 128$).

$$P(t) = \underbrace{\left[\left[\left[\left[\left[\left[M^2\right]^2\right]^2\right]^2\right]^2\right]^2\right]^2}_{7 \text{ times}} P(0)$$

The proper combination of squared matrices required a minimum of 9 matrix multiplications (6 to reach the highest power of M and 3 to combine the matrices of lower powers), in addition to additional memory space. Undoubtedly, chosing the scaling factor to be an integer power of two yields a more efficient routine.

The main difference between the squaring algorithm and the stepping algorithm is that the squaring one operates on the base matrix while the stepping one operates on the state vector. In general, the squaring method is more efficient. It travels through time on a logarithmic scale, while the stepping routine travels through time on a linear scale. While the squaring algorithm does not give intermediary results in linear time, it does produce intermediary results in logarithmic time.

3.4.4 The Combined Algorithm

A third approach that presents itself is a combination of the squaring and stepping algorithms. In this way, one can get the efficiency of the squaring algorithm with the flexibility of the stepping technique. Essentially, the idea is to use the squaring method until the first intermediary time, and then to use the stepping option for the rest of the time. It will be shown in section 3.5 that the combined method may actually require less computer work.

Let us define s to be the number of steps completed until the point at which the algorithm switches from squaring to stepping. Thus, we have $\alpha_s = \frac{\ln s}{\ln 2}$ squarings.

$$P(s\Delta t) = M^s P(0)$$

We then need to do $\frac{n}{s}$ steps to reach the final time.

$$P(2s\Delta t) = M^s P(s\Delta t)$$

$$\vdots$$

$$P(n\Delta t) = M^s P((n-s)\Delta t)$$

The output now can be given every ($s\Delta t$) of time. The combined method, then, has the benefit of the flexibility of the stepping algorithm and the efficiency of the squaring algorithm.

Clearly, there are two parameters that must be chosen for any scaling and propagating algorithm. The first parameter is the scaling factor n, or equivalently the time step Δt. The second parameter to be defined is the degree of the approximation, k or p. Additionally, the method of power series approximation needs to be decided, as well as the integration algorithm. The goal is to choose a scale factor and a degree of approximation that result in an efficient and accurate algorithm. More information on the selection of Δt and k (or p) will be given in Chapter 8, after the error propagation patterns are discussed.

3.5 Computer Work Associated with the Solution Techniques

So far, two power series methods of approximating the base matrix have been introduced. Additionally, three integration methods have been discussed. One of the bases for comparison of these options is the amount of computer work each one requires. This section will contrast the various choices for a general square matrix of dimension m. The matrix is assumed full, so no sparse matrix routines are considered.

First, we define the amount of computer work necessary for basic matrix operations. A basic unit of computer work is the floating point operation, or *flop*. A flop is defined as the amount of time necessary for a given computer to execute the FORTRAN statement

$$A(I,J) = A(I,J) + T*A(I,J)$$

This definition contains one floating point multiplication, one floating point addition, some index and subscript calculations, and a few storage references, thereby incorporating most of the basic mathematical and computer operations. Notably, this type of operation is frequently used in many matrix operations.

Using the above definition, pre-multiplying an (mxm) matrix with a m-column vector requires m^2 floating point operations. This can be seen easily in that each column of the matrix must multiply one element in the column vector. Since there are m elements in the vector, the total number of operations is m^2. The same result would hold true if a row vector pre-multiplied an (mxm) matrix. If one were to multiply an (mxm) matrix with another (mxm) matrix, the process requires m^3 operations. This can be understood if one

considers one of the matrices to be a series of m vectors. Thus we just have m matrix-multiply-vector operations for a total of m^3 operations.

3.5.1 Computer Work for the Base Matrix

Consider now the truncated Taylor series approximations as the means to determine the base matrix. The minimum amount of work necessary to calculate the first k+1 terms of the power series (including the identity matrix) is approximately given by $(k-1)m^3$ operations, one m^3 operation for each squaring of the transition matrix. This value assumes that the lower ordered matrices are saved and used to calculate the higher order matrices. That is, the $(At)^2$ matrix is saved and used to calculate the $(At)^3$ matrix, rather than calculating $(At)^3$ only from (At). The additional work necessary to multiply the matrices by the coefficients and to add the matrices has been neglected because of their lower order, m^2.

By comparison, consider the diagonal Padé approximation as the means to calculate the base matrix. The diagonal Padé approximations require $(p+1)m^3$ operations. This value includes $(p-1)m^3$ operations for the (p+1) terms in the numerator and denominator. It also includes m^3 operations for a matrix inversion and m^3 operations for the matrix multiplication of the numerator and the denominator. Again, this value assumes that the lower ordered matrices are saved and used to calculate the higher order matrices. Also, it is assumes that the algorithm can simultaneously calculate the numerator and denominator. Lastly, the lower ordered operations have again been neglected.

One can see that if p = k, the Taylor approximation requires less work to calculate than the corresponding Padé approximation. However, the Padé approximation has an order of 2p, while the Taylor approximation only has an order of k. The two methods require similar amounts of work and result in the same order of approximation if k=4 and p=2. For values less than these, Taylor series yield a higher order of approximation for equivalent computer work. On the other hand, for values greater than these, Padé approximations yield a higher order of approximation for equivalent work.

3.5.2 Computer Work for the Stepping Algorithm

The amount of work necessary to compute the matrix exponential using a stepping algorithm is directly dependent on the number of steps required. Each step is a matrix-multiply-vector operation. Thus, if we have n steps, the amount of work required is $n(m^2)$. This is again assuming a full matrix and a full vector. The amount of work

required to step the base matrix is obviously independent of the power series that computed the base matrix.

To discern the total amount of computer work for the stepping algorithm, the work necessary to calculate the base matrix must also be included. Using the information in section 3.5.1 above, we see that the amount of work using a stepping method with Taylor approximations is

$$(k-1)\, m^3 + nm^2$$

For Padé approximations, the amount of work necessary using a stepping algorithm is

$$(p+1)\, m^3 + nm^2$$

It is important to note that the values of k, p, and n will change depending on the accuracy sought. Thus, a direct comparison of work should not be made without considering the associated order of the approximation and the roundoff and integration errors. Chapter 7 contrasts various methods according to the amount of computer work necessary to achieve a certain accuracy. according to the amount of computer work necessary to achieve a certain accuracy.

3.5.3 Computer Work for the Squaring Algorithm

The computer work for the squaring algorithm is dependent on the number of squarings necessary. Instead of performing n steps, we now perform $\frac{\ln n}{\ln 2}$ squarings. Each squaring is a matrix-multiply-matrix operation. Thus the computer work necessary is $\left(\frac{\ln n}{\ln 2}\right)$. Again, these values are assuming a full transition matrix.

The total amount of computer work for any method must include the computer work necessary to calculate the base matrix. Including the amount of work necessary to determine the base matrix using Taylor and Padé, the total work is

$$(k-1)\, m^3 + \frac{\ln n}{\ln 2}\, m^3 \qquad \text{(Taylor)}$$

$$(p+1)\, m^3 + \frac{\ln n}{\ln 2}\, m^3 \qquad \text{(Padé)}$$

Again, it is important to note that a direct comparison of work cannot be made without considering the associated order of approximation and numerical errors.

The values derived in these two sections suggest that there is a matrix dimension at which the stepping algorithm is more efficient that the squaring one. We can do a direct comparison between the squaring and the stepping methods to determine the conditions on m that squaring more efficient than stepping. Ignoring the work required to compute the base matrix, the following equation gives the relationship for when stepping requires less work than squaring.

$$nm^2 < \frac{\ln n}{\ln 2} m^3 \qquad \text{OR} \qquad \frac{n \ln 2}{\ln n} < m$$

Thus, if the dimension of the matrix is greater than the coefficient $\frac{n \ln 2}{\ln n}$, stepping is more efficient than squaring. The scaling factor tends to be large, so this bound is rarely exceeded.

3.5.4 Computer Work for the Combined Method

The computer work for the combined method is relatively basic. It is simply a combination of the amount of work for the stepping and squaring routines. The scheme uses squaring until the switching point and stepping thereafter, resulting in the following computer work

$$\frac{n}{s} m^2 + \frac{\ln s}{\ln 2} m^3$$

The work necessary to compute the base matrix using either Taylor or Padé can simply be added on to this equation to arrive at the total computer work for the combined method.

It was state previously that the combined method has both the benefits of efficiency and flexibility. In addition, the combined method may also demand less computer work. The factor nm^2 in the stepping technique and the factor $\frac{\ln n}{\ln 2} m^3$ in the squaring technique suggest that a minimum amount of work may be achieved if the two methods are combined. This is in fact the case. Ignoring the amount of work required to create the base matrix, the computer work required (CW) to compute the matrix exponential is given by

$$CW = \frac{n}{s} m^2 + \frac{\ln s}{\ln 2} m^3$$

To find the minimum work, we differentiate with respect to s and set equal to zero.

$$\frac{d}{ds} CW = -\frac{n}{s^2} m^2 + \frac{1}{s \ln 2} m^3 = 0$$

Solving for s gives

$$s = \frac{n \ln 2}{m}$$

Thus, the computer work necessary to compute the matrix exponential can be minimized by using a squaring routine up to the point in time when $s = \frac{n \ln 2}{m}$, and then using a stepping algorithm from s until the final time is reached.

3.6 Recapitulation

This chapter has examined two different power series approximations used to compute the matrix exponential. The Taylor series was used to define the matrix exponential, and the truncated Taylor series was used to approximate the matrix exponential. The order of the approximation was defined as the number of derivatives of the exponential function that are matched by the power series. For Taylor series, the order of the approximation was shown to be k, the highest power of the transition matrix. It was shown for a general class of matrices that the truncated Taylor series was unacceptable because of the slowness of convergence and cancellation problems.

The second power series method introduced was Padé approximations. The Padé approximation is defined as a ratio of two polynomials with the coefficients of the polynomials chosen so the power series matches the derivatives of the exponential function. As such, the order of the approximation was shown to be the sum of the order of the polynomials, p+q. In the case of the diagonal approximations, this value is 2p. The advantage of Padé approximations is that they converge faster than the truncated Taylor series for equivalent powers of the transition matrix At. Despite this fact, Padé approximations still suffer from the slowness of convergence for the general case. Additionally, the rational approximation requires a matrix inversion, injecting further unknown into the solution process. Overall, Padé approximations were also shown to be unacceptable for the general case.

To combat the convergence and cancellation problems of the direct series methods, the notion of scaling was introduced. Scaling and propagating exploits a fundamental concept of the exponential function

$$e^{At} = (e^{At/n})^n$$

Scaling leads to two techniques, stepping and squaring. The stepping algorithm operates on the state vector to produce a linear propagation in time. In contrast, the squaring algorithm operates on the base matrix to give a logarithmic propagation in time. It was shown that the squaring method is generally more efficient. However, if a linear time history of the state probabilities is desired, the stepping algorithm is necessary. Additionally, a combined routine is possible that takes advantage of both the efficiency of the squaring and the flexibility of the stepping.

The computer work associated with each series method as well as the various scaling techniques were given. The results showed that a condition on the size of the transition matrix exists for which stepping is desired over squaring. This condition is rarely met is actual use. The results also showed that there is an algorithm that combines stepping and squaring and results in the minimum amount of computer work for the given methods.

The last two chapters have discussed Markov models and methods of solving for the probabilities associated with Markov models via the matrix exponential. Power series methods were used to approximate the matrix exponential resulting in an order of the approximation. Unfortunately, the order of the approximation does not give the entire story of the numerical error associated with these solution methods. Knowledge of roundoff error and integration error plays an important role in the selection of an efficient and accurate solution technique. The propagation patterns of roundoff error and integration error are the topics of the next three chapters. Chapter 4 examines roundoff error, while Chapters 5 and 6 investigate the integration error associated with Taylor series and Padé series, respectively.

Chapter 4

Roundoff Error

In determining the state probabilities of a Markov model via the matrix exponential, the accuracy of the approximation must be considered. It was shown that the order of the approximation as defined in Chapter 3 gives an indication of accuracy. However, the order of the approximation does not define the total error incurred in the solution of the matrix exponential. If a digital computer is used to solve for the state probabilities, both roundoff error and integration error effect the integrity of the final computed answer.

Integration error is closely related to the order of the approximation. It is function of the approximation method used, the order of the approximation. The causes and propagation of integration error, as well as some results for various techniques, are discussed in detail in Chapters 5 and 6.

Roundoff error is the result of the use of a digital computer to solve for the state probabilities. The computer is not capable of exact representation of some rational numbers or any irrational numbers. This deficiency is evidenced in roundoff error. These inaccuracies are propagated when the values are operated on mathematically. Often times, this propagation of errors can lead to catastrophic results, invalidating the final numerical answer. The problem of roundoff error and its propagation patterns are the subject of this chapter. Some roundoff error reduction techniques are introduced, and the error propagation patterns for some of the integration algorithms are also given.

The goals of this chapter are threefold. First, we wish to define roundoff error, determine its sources, and define its propagation patterns. Once the first step is accomplished, roundoff error reduction methods will be investigated. As best they can, these methods attempt to reduce the influence of the various sources of roundoff error. Lastly, equations that allow an *a priori* determination of the roundoff error for the various solution techniques are derived.

4.1 Computer Approximations

There are two basic data types used in the solution of the matrix exponential, fixed point or integer type representation and floating point or real type representation. The integer data types are whole numbers represented internally as binary numbers. Since all

whole numbers can be represented exactly in binary, there is no error in the internal representation of the integer data type. Since there is no error, the mathematical operations of addition, subtraction, and multiplication are performed exactly, provided the numerical range of the machine is not exceeded. Divisions may not be performed exactly because a whole number divided by a whole number is not constrained to be another whole number, and therefore, may not be exactly represented by the integer data type.

In contrast to the fixed point integer data type, the floating point data type is used to represent numbers that are not necessarily whole numbers. Since we are solving for state probabilities that are less than or equal to unity, this is the data type we will be concerned with. A floating point value is represented in a mantissa (fraction) and an exponent. The mantissa gives the values, and the exponent indicates where to place the decimal point. For example, 625 would have a mantissa of 0.625 and an exponent of 3--$0.625 \times 10^3 = 625$.

In a digital computer, both the mantissa and the exponent are stored internally as binary numbers. The number of binary bits, or word length, used to store the mantissa and exponent vary from computer to computer. Typical values range from twenty-four to fifty-six bits for the mantissa, and from seven to ten bits for the exponent. One bit is usually reserved for the sign of the mantissa In many computers, single precision is given by a twenty-four bit mantissa and a seven bit exponent, yielding between six and seven digits of accuracy. Double precision, on the other hand, uses a fifty-six bit mantissa and a ten bit exponent. This yields an accuracy of greater than fifteen but less then sixteen digits.

The problem arises in that certain numbers are not able to be stored exactly in the mantissa. The finite word length implies there exists a maximum (and minimum) value for the exponent and that not all numbers can be represented by the mantissa. Definitely a repeating fraction, such as $2/3 = 0.66666\ldots$, cannot be represented exactly by the mantissa and must be truncated. Additionally, there are other numbers that cannot be represented exactly due to the finite word length. For example, 1.2 cannot be stored exactly using the floating point data type and must be approximated. Because of the truncations and the approximations, these data types carry an inherent amount of uncertainty, referred to here as roundoff error.

In mathematics a real number is written as 7.39, or symbolically as x. For digital computers, this number should be rewritten to include the possible roundoff error, $x \pm \Delta x$ where Δx is the uncertainty in x due to the finite memory space. Δx is also known as the absolute roundoff error of x. Let $\tilde{x}$ denote the approximation to the value x. That is,

$$\tilde{x} = x \pm \Delta x$$

Thus, we define the absolute roundoff error (ARE) of x as the difference between the approximate value and the exact value.

$$ARE = x - \tilde{x} = \Delta x$$

This approximate value can be equivalently written as

$$\tilde{x} = x(1 \pm \varepsilon_x)$$

where x is the desired real number and ε_x is the relative roundoff error (RRE). For example, a digital computer with seven digits of accuracy for single precision would have $\varepsilon_x = 10^{-7}$. A digital computer with sixteen digits of accuracy for double precision would have $\varepsilon_x = 10^{-16}$. Comparing the above equations, we see that Δx is equivalent to $x\,\varepsilon_x$. We can also see that the definition for ε_x can be given by

$$\varepsilon_x = \frac{x - \tilde{x}}{x}$$

In words, the relative error is defined as the difference between the exact value and the approximate value divided by the exact value. Alternatively, relative error can be defined as the absolute error divided by the exact value.

Roundoff error is propagated when these data are operated upon mathematically. Derived below are the equations needed to calculate the roundoff error for the mathematical operations of both addition and multiplication which are the primary operations used in the matrix exponential solution techniques. The equation for the propagation of roundoff error when subtraction is used is also given. Since division is rarely used, the propagation of roundoff error in this case has been omitted, although a similar analysis can be done to give similar results.

The equation for relative roundoff error gave the error as either positive or negative because approximating a number with a digital representation can yield either positive or negative error depending on the number. The following derivations assume only positive relative roundoff error. In this way, the relative roundoff error at the end of the mathematical operation is the maximum error that could be incurred. Therefore, the equations developed represent a conservative upper bound on the actual relative roundoff error experienced.

ADDITION:

$$\tilde{x} + \tilde{y} = \tilde{z}$$

$$x(1+\varepsilon_x) + y(1+\varepsilon_y) = z(1+\varepsilon_z)$$

$$x + x\varepsilon_x + y + y\varepsilon_y = z(1+\varepsilon_z)$$

$$(x+y)\left(1 + \frac{x\varepsilon_x + y\varepsilon_y}{x+y}\right) = z(1+\varepsilon_z)$$

$$\left(\frac{x\varepsilon_x + y\varepsilon_y}{x+y}\right) = \varepsilon_z$$

From the above derivation one can see that if the relative error of both x and y are equal, they are also equal to the relative error of z. Further, if x is much larger than y and ε_x and ε_y are similar orders of magnitude, the relative error of x is approximately the relative error of z.

MULTIPLICATION:

$$\tilde{x} \cdot \tilde{y} = \tilde{z}$$

$$x(1+\varepsilon_x) \cdot y(1+\varepsilon_y) = z(1+\varepsilon_z)$$

$$(x + x\varepsilon_x) \cdot (y + y\varepsilon_y) = z(1+\varepsilon_z)$$

$$xy + xy\varepsilon_x + xy\varepsilon_y + xy\varepsilon_x\varepsilon_y = z(1+\varepsilon_z)$$

$$(xy) \cdot (1 + \varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y) = z(1+\varepsilon_z)$$

$$\varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y = \varepsilon_z$$

neglecting the higher order terms

$$\varepsilon_x + \varepsilon_y = \varepsilon_z$$

One can see from the above equation that the relative error of a product is approximately the sum of the relative errors of the multipliers. The term $\varepsilon_x \varepsilon_y$ is negligible until the the roundoff error becomes so great as to completely corrupt the result.

Besides addition and multiplication, subtraction may be used in the solution techniques. It is used whenever the conservation of system probability is used. Below is given the relative roundoff error equation for the subtraction operation. To give a conservative bound for the error, opposite signs are used for the relative errors of the terms being subtracted.

SUBTRACTION:

$$\tilde{x} - \tilde{y} = \tilde{z}$$

$$x(1+\varepsilon_x) - y(1-\varepsilon_y) = z(1+\varepsilon_z)$$

$$x + x\varepsilon_x - y + y\varepsilon_y = z(1+\varepsilon_z)$$

$$(x-y)\left(1 + \frac{x\varepsilon_x + y\varepsilon_y}{x-y}\right) = z(1+\varepsilon_z)$$

$$\left(\frac{x\varepsilon_x + y\varepsilon_y}{x-y}\right) = \varepsilon_z$$

The equation for the relative error due to the subtraction operation is very similar to the relative error due to the addition operation with one significant difference. The denominator is now the difference of the two numbers. This equation shows why there is a danger of inducing a lot of roundoff error with the subtraction operation. If x and y are similar numbers, their difference can be very small. This small difference causes a large relative roundoff error in the result.

4.2 Roundoff Error Reduction Techniques

Now that the equations for the propagation patterns of roundoff error have been developed, we look at the major sources of the roundoff error and possible means to reduce these errors. There are two primary sources of roundoff error when the series solution techniques are used to solve for the state probabilities via the matrix exponential. The first source is the inherent uncertainty in the machine precision of digital computers as describe

above. These uncertainties grow when acted upon numerically, and they can affect the integrity of the results. The second source of error is due to the addition or subtraction of values of widely varying magnitude and to the subtraction of similar numbers. Often times, significant portions of the variables are lost due to roundoff error.

For example, if we wish to add a very small number (say 10^{-9}) to a large number (say unity) in single precision, we would get

$$1.000000 - 10^{-9} = 1.000000$$

This yields a relative roundoff error of only 10^{-9}. If, however, we now subtract unity from this result, we get zero when the actual value should be 10^{-9}. This results in a relative error of 1. Furthermore, if this second result is to be multiplied by a final time of 10^{6} the computer would still yield zero, while the desired answer would be 10^{-3}.

Described below are two methods that help reduce the roundoff error associated with the solution techniques given in Chapter 3. The first method, variable renormalization, uses the conservation of system probability to retard the growth of the roundoff error in mathematical operations due to the inherent uncertainty of the machines. The second method, accumulator algorithms, accumulates the roundoff error due to the addition and subtraction of numbers widely varying in magnitude. These accumulated errors are added back to the appropriate quantity once they have become within range of the computer's precision.

4.2.1 Variable Renormalization

The process of variable renormalization capitalizes on the time invariance of the base matrix and on the conservation of system probability. The requirement of conservation of system probability is that each column of the base matrix must sum to unity. Additionally, the sum of the probability state vector must also be unity. These conditions hold regardless of the number of times the base matrix is squared or the number of times the state vector is stepped through time. It was shown in Chapter 2 that the conservation of system probability can be used to calculate the probability of a certain state if the probabilities of all the other states are known. Variable renormalization uses this premise to calculate the element with the largest state probability.

An element is renormalized when it has been computed using the conservation of system probability. That is, the (i, j) element is renormalized when it is determined by summing all the other elements in that column, and then subtracting this sum from one.

$$(i, j) = 1 - \sum_{\substack{k=1 \\ k \neq i}}^{m} (k, j)$$

The term variable is used because because the element being renormalized is the largest in value in the column. The largest element may vary position with column and with time, causing the term being renormalized to be variable.

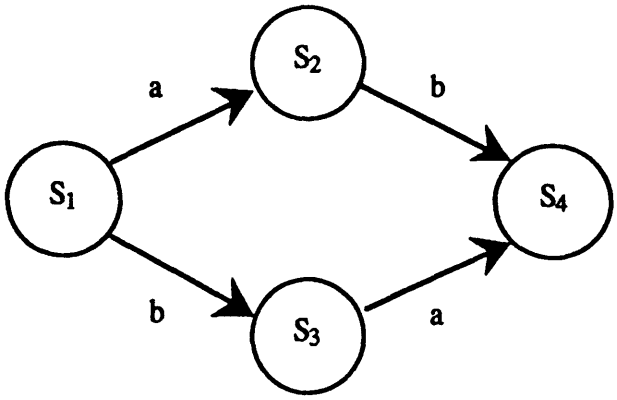
The element that is renormalized is the largest element in each column. The state probability that has the largest magnitude is nominally the diagonal element, and the roundoff error of the diagonal element has a great amount of influence on the roundoff error of the other elements. Thus, renormalizing the diagonal element reduces a large proportion of the roundoff error of that element, thus reducing the error of the other elements in the matrix.

The reasons that the largest element must be renormalized and not just the diagonal element can be seen from the equation for the propagation of roundoff error through subtraction. The equation shows that if two numbers are similar in magnitude, their difference will have a large relative roundoff error. If on the other hand, two numbers are very different in magnitude, their difference will have a relative roundoff error similar to the relative error of the larger number. If an element is not the largest in a column (say, 10^{-3}), the rest of the column will sum to a value greater than the element being renormalized (sum = 0.999). If the problem was run in single precision, the very best that could be achieved is a relative roundoff error of 10^{-4}. The roundoff error of the sum will dominate the subtraction process and the benefits of the renormalization are lost.

Suppose, on the other hand, the element being renormalized in a certain column is very close to one. The other elements in that column will then be much less than one. Even the sum of these other elements will still be much less than one. When we then subtract this sum from unity, the relative roundoff error of the product will be similar to the relative error of unity, which is the machine precision ε. Thus, by renormalizing the largest element in each column, we reduce the relative roundoff error of that element to approximately the machine precision. We also avoid the dangers associated with subtracting two numbers that are similar in magnitude.

This process may be used with either a squaring or a stepping algorithm, although it is much more effective with a squaring algorithm. After each time the matrix is squared, the largest element in each column is renormalized. In this way, the conservation of system probability is guaranteed after each squaring of the matrix, keeping the overall roundoff error propagation down. On the other hand, in a stepping algorithm the largest element in the state vector is renormalized since the base matrix is never operated upon. The largest element in the state vector, however, only has limited influence on the other elements in the vector. Regardless of which method is used, the computer work associated with variable renormalization is negligible. For squaring, renormalization requires m^2 operations. For stepping, renormalization requires only m operations.

To demonstrate the effectiveness of variable renormalization, the following example is presented. Suppose we have the non-cyclical Markov model that was used in the reliability analysis in chapter 2. The Markov model and continuous-transition differential equations are repeated below for convenience.



$$\frac{dP(t)}{dt} = \begin{bmatrix} -(a+b) & 0 & 0 & 0 \\ a & -b & 0 & 0 \\ b & 0 & -a & 0 \\ 0 & b & a & 0 \end{bmatrix} P(t)$$

Figure 4-1: Four-State Continuous-Transition Markov Model

For a this example, we will take the simple base matrix, M_1. That is, we will only use the first term in the Taylor series expansion for the matrix exponential. Using a generic time step of Δt results in the following matrix equation.

$$P(t) = \begin{bmatrix} 1 - (a+b)\,\Delta t & 0 & 0 & 0 \\ a\,\Delta t & 1 - b\,\Delta t & 0 & 0 \\ b\,\Delta t & 0 & 1 - a\,\Delta t & 0 \\ 0 & b\,\Delta t & a\,\Delta t & 1 \end{bmatrix} P(0)$$

Now, suppose we have typical values for the failure rates, $a = b = 10^{-5}\ hr^{-1}$, and for the time step, $\Delta t = 10^{-2}$ hr. Using a single precision representation of seven digits of accuracy, these values give the base matrix shown below.

$$M_1 = \begin{bmatrix} 1.000000 & 0 & 0 & 0 \\ 1.000000\text{x}10^{-7} & 1.000000 & 0 & 0 \\ 1.000000\text{x}10^{-7} & 0 & 1.000000 & 0 \\ 0 & 1.000000\text{x}10^{-7} & 1.000000\text{x}10^{-7} & 1.000000 \end{bmatrix}$$

Notice that the conservation of system probability is violated in that three of the columns do not sum to unity as required. However, because a single precision representation of seven digits of accuracy is being used, the above base matrix does meet the conservation of system probability requirement within the accuracy of the machine.

Still, if we use this matrix in a squaring algorithm, problems will occur. For example, after only four squarings of the matrix ($2^4 = 16$) in single precision, the base matrix becomes

$$M_1^{16} = \begin{bmatrix} 1.000000 & 0 & 0 & 0 \\ 1.600000\text{x}10^{-6} & 1.000000 & 0 & 0 \\ 1.600000\text{x}10^{-6} & 0 & 1.000000 & 0 \\ 2.400000\text{x}10^{-12} & 1.600000\text{x}10^{-6} & 1.600000\text{x}10^{-6} & 1.000000 \end{bmatrix}$$

Clearly the conservation of probability flow is being violated, even within the accuracy of the machine. The diagonal elements would imply that there are no exiting transitions from the corresponding states, yet the rest of the states show the effects of entering transitions. Moreover, each of the first three columns sum to a value greater than one implying a system probability greater than unity. For the integration scheme to be numerically more precise, variable renormalization is performed in this case.

Column 1 will be used as an example to demonstrate the variable renormalization process. The diagonal element is the largest in the first column with a value of 1. Therefore, the other elements in the column are summed in single precision

$$1.600000\text{x}10^{-6} + 1.600000\text{x}10^{-6} + 2.400000\text{x}10^{-12} = 3.200002\text{x}10^{-6}$$

This value is then subtracted from unity to obtain the diagonal element of column 1 by insuring the conservation of probability flow. Again, this process is done as a single precision calculation

$$1.000000 - 3.200002\text{x}10^{-6} = 0.9999968$$

Obviously a discrepancy exists between the regular squaring integration, which gave a value of 1.000000, and the squaring integration with variable renormalization, which returned a value of 0.9999968. Because the method of variable renormalization conserves probability flow, it provides the more accurate results. Applying the renormalization technique to the entire matrix above, one gets

$$M_1^{16} = \begin{bmatrix} 0.9999968 & 0 & 0 & 0 \\ 1.600000\text{x}10^{-6} & 0.9999984 & 0 & 0 \\ 1.600000\text{x}10^{-6} & 0 & 0.9999984 & 0 \\ 2.400000\text{x}10^{-12} & 1.600000\text{x}10^{-6} & 1.600000\text{x}10^{-6} & 1.000000 \end{bmatrix}$$

Note that each of the columns in this matrix now sums to unity using single precision calculations preserving the conservation of system probability

In practice, each column of the matrix would be renormalized after each squaring of the base matrix. In the example above, renormalizing after the first three squarings would not have produced any noticeable results in the diagonal elements. The off diagonal elements would still have been too small in magnitude to effect the subtraction process in renormalization. In this example, renormalization would now be done at all successive squarings of the matrix. The sixteenth power of the matrix, therefore, is given to more fully demonstrate the effectiveness of renormalization.

4.2.2 Accumulator Methods for Stepping Routines

A major drawback of the variable renormalization method is its relative ineffectiveness in reducing roundoff errors for stepping algorithms. Additionally, results and analysis show that renormalization is effective only until the characteristic time of the system even for a squaring routine (see Section 4.3.5). For these reasons, another method of reducing roundoff error was investigated. This class of methods, called accumulator

methods, attacks the same source of roundoff error but approaches the problem in a different manner.

Accumulator methods do not rely on the conservation of system probability to account for the roundoff error due to truncation. Instead they calculate the portion that has been truncated. This truncated portion is then accumulated until it becomes significant enough to be added back in. For example, if we compute the difference of 1 and 10^{-9} with an accumulator we would get.

value: $1.000000 - 10^{-9} = 1.000000$ accumulator: 10^{-9}

If we then subtract this value from unity we would get zero. However, if we add the accumulator since it has become significant, we would get the proper result of 10^{-9}.

If we define z as the accumulator, y as the current value, and dy as the portion to be added (or subtracted) to the current value, the following algorithm can be given

$$u(i) = z(i) + dy(i)$$

$$y(i+1) = y(i) + u(i)$$

$$z(i+1) = u(i) - [y(i+1) - y(i)]$$

The basis of the algorithm exists in the third step. The accumulator is determined by finding the difference between the value that should have been added u(i) and the value that actually was added [y(i+1) - y(i)]. The accumulator is added into the increment of the next step to account for the error in the computation.

This technique of reducing roundoff error is due to [Møller, 1965]. Because the essential idea for this technique can be traced back to Gill, it is sometimes referred to as the Gill-Mϕller algorithm [Butcher, 1987].

The above algorithm is written for a scalar operation. Some alterations are necessary for the matrix operations that are used in the solution of the matrix exponential. As an example, we will develop the accumulator algorithm for a stepping routine using the truncated Taylor series base matrix M_1.

$$M_1 = I + A\Delta t$$

The set of difference equations for the state probabilities is given by

$$P(n\Delta t) = M_1 P((n-1)\Delta t)$$

$$P(n\Delta t) = (I + A\Delta t) P((n-1)\Delta t)$$

$$P(n\Delta t) = P((n-1)\Delta t) + A\Delta t P((n-1)\Delta t)$$

For convenience, the iterative equation for the state probabilities will be written with the time step implicit

$$P(n) = P(n-1) + A\Delta t P(n-1)$$

We now rewrite this equation to demonstrate the possible errors in the state vector.

$$P(n) = \{P(n-1) + Y\} + A\Delta t \{P(n-1) + Z\}$$

where Z and Y are vectors of additive errors. This equation leads us to the accumulator algorithm. Writing the equation in expanded form, we get

$$P(n) = P(n-1) + Y + A\Delta t P(n-1) + A\Delta t Z$$

The points where error may evolve due to addition and subtraction of numbers widely varying in magnitude can now be seen clearly.

We are now prepared to write the stepping algorithm with accumulator. First, we define temporary vectors F and G which help to track the computations. We also define the vector X to be the vector of accumulated errors from the previous steps. The stepping algorithm with accumulator is then given by

Stepping Algorithm with Accumulator for M_1

$$F_n = Z_n + (A\Delta t) Z_n$$

$$G_n = F_n + (A\Delta t) P_n$$

$$Y_n = F_n - \{[(G_n - A\Delta t(1) P_n(1)) - A\Delta t(2) P_n(2)] - \ldots - A\Delta t(m) P_n(m)\}$$

$$P_{n+1} = P_n + G_n$$

$$X_n = G_n - \{P_{n+1} - P_n\}$$

$$Z_{n+1} = Y_n + X_n$$

where $A\Delta t(1)\ P(1)$ represents the product of the first column of the matrix $A\Delta t$ times the first element of the state probability vector P. It is required to perform this line of the algorithm as a series of vector operations rather than a single matrix operation. A series of vector operations accumulates the error that may incur during the addition of terms within the matrix-vector operation.

Accumulator algorithms are base matrix dependent. The algorithm given above is appropriate to use with M_1 only. The basic ideas of the accumulator algorithms, however, are easily applied to situations in which other base matrices are used. The key to any accumulator algorithm is to accumulate the errors that occur in the additions or subtractions. The accumulator algorithm for the base matrix M_2 is developed below so that commonalties among the accumulator methods may be seen.

The second order accumulator has the base matrix M_2

$$M_2 = I + A\Delta t + \frac{(A\Delta t)^2}{2}$$

The set of difference equations for the state probabilities is then given by

$$P_n = M_2\, P_{n-1}$$

$$P_n = (I + A\Delta t + \frac{(A\Delta t)^2}{2})\, P_{n-1}$$

If we consider an additive error to P_{n-1} and expand the equation we get

$$P_n = P_{n-1} + X_n + (A\Delta t)\, P_{n-1} + (A\Delta t)\, Y_n + \frac{(A\Delta t)^2}{2} P_{n-1} + \frac{(A\Delta t)^2}{2} Z_n$$

Once again we have the basis for the stepping algorithm with an accumulator. W, X, Y, and Z are vectors of accumulated error. F, G, H, and J are temporary vectors.

<u>Stepping Algorithm with Accumulator for M_2</u>

$$F_n = \frac{(A\Delta t)^2}{2} Z_n + (A\Delta t)\, Z_n$$

$$^*V_n = \frac{(A\Delta t)^2}{2} Z_n - \{F_n - (A\Delta t)\, Z_n\}$$

$$G_n = F_n + \frac{(A\Delta t)^2}{2} P_n$$

$$^{*}W_n = F_n - \{G_n - \frac{(A\Delta t)^2}{2} P_n\}$$

$$H_n = G_n + Z_n$$

$$X_n = G_n - \{H_n - Z_n\}$$

$$J_n = H_n + (A\Delta t)\, P_n$$

$$^{*}Y_n = H_n - \{J_n - (A\Delta t)\, P_n\}$$

$$P_{n+1} = H_n + P_n$$

$$Z_{n+1} = V_n + W_n + X_n + Y_n$$

The lines of the algorithm preceded by an asterisk (*) must be executed as a series of vector operations as described above. Clearly, the second accumulator algorithm is in the same general format as the first one. Using the base matrix M_2 requires a few more temporary matrices as well as a few more accumulator matrices. This increase in memory would be expected since the M_2 base matrix has one more term of the Taylor series than the M_1 base matrix.

In addition to the increase in memory required, there is an increase in computer work with the accumulator methods. This increase in computer work would also be expected because the accumulator checks all of the addition operations, approximately doubling the computer work. Also, the matrix-times-vector operations have been transformed into a series of vector-times-component operations. Since the base matrix is broken up into its Taylor series, the amount of work could double or triple depending on the base matrix used. The additional computer work required for a stepping algorithm implies that these methods should be used conservatively and only if needed for accuracy.

Overall, the accumulator methods are a means to increase the accuracy of the mathematical computation and essentially create a quasi-double precision from single precision. They can also be used with double precision data types to essentially double the accuracy of the machine. By creating accumulator vectors, the algorithm computes and stores the portion of the value that is lost due to truncation when two numbers of different magnitudes are added. These truncated values are then added back in when they become significant to the computation. Accumulator methods, however, cannot overcome the

inherent inaccuracies due to the finite machine precision. Thus, these errors continue to propagate during the accumulator algorithms.

4.3 Roundoff Error Results

So far this chapter we have developed the roundoff error propagation patterns for the mathematical operations of addition, subtraction, and multiplication. We have also developed two methods for reducing the roundoff error that accumulates. Now, we will use the equations for the propagation of roundoff error to bound the roundoff error in the various solution techniques developed in chapter 3. Both the base matrix and the state vector have an associated error matrix and error vector. These error matrices are determined at each stage in the integration. The results of these studies for the various solution techniques are given in the sections below.

4.3.1 Base Matrix Errors

The calculation of the base matrix has initial errors as was discussed above. These errors are reflected in the associated error matrix. We begin with the continuous-time transition matrix A, and its associated relative error matrix E_A for the two component system described in Section 4.2.1.

$$A = \begin{bmatrix} -(a+b) & 0 & 0 & 0 \\ a & -b & 0 & 0 \\ b & 0 & -a & 0 \\ 0 & b & a & 0 \end{bmatrix} \qquad E_A = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 1\varepsilon & 1\varepsilon & 0 & 0 \\ 1\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 1\varepsilon & 1\varepsilon & 0 \end{bmatrix}$$

where ε is the machine precision. The -(a+b) term in the first diagonal has a relative error of 1ε because both a and b have errors of 1ε. The roundoff error equation for addition shows that if the error of the terms to be summed are equal to each other, they are also equal to the error of the sum.

We can now determine the relative error matrices associated with the various base matrices using Taylor series approximations. The base matrix M_1 for the continuous-time matrix given above and its associated relative error matrix are given below

$$M_1 = \begin{bmatrix} 1 - (a+b)\Delta t & 0 & 0 & 0 \\ a\Delta t & 1 - b\Delta t & 0 & 0 \\ b\Delta t & 0 & 1 - a\Delta t & 0 \\ 0 & b\Delta t & a\Delta t & 1 \end{bmatrix} \qquad E_{M_1} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix}$$

The off-diagonals have an error of 2ε because they are the product of two numbers (i.e. $a \bullet \Delta t$) each with an individual error of 1ε. The roundoff error equation for multiplication shows that the error for a product is the sum of the individual errors. The diagonals, on the other hand, have an initial relative error of approximately 1ε because the product of the failure rate and the time step is subtracted from unity. This yields to the following equation for the relative roundoff error of the diagonal term

$$\text{RRE (diagonal element)} = \frac{[1 \bullet \varepsilon] + [(a+b)\Delta t \bullet \varepsilon]}{1 - (a+b)\Delta t} \approx 1\varepsilon$$

If we assume that the quantity $(a+b)\Delta t$ is small compared to unity, the relative error of the diagonal element is 1ε. This is usually a good assumption since Δt is chosen to be small to insure proper convergence.

The second order base matrix M_2 for the same continuous-time matrix and its associated relative error matrix are given below.

$$M_2 = \begin{bmatrix} 1-(a+b)\Delta t - \frac{(a+b)^2}{2} & 0 & 0 & 0 \\ a\Delta t - \frac{a[(a+b)+b](\Delta t)^2}{2} & 1-b\Delta t + \frac{b^2(\Delta t)^2}{2} & 0 & 0 \\ b\Delta t - \frac{b[(a+b)+a](\Delta t)^2}{2} & 0 & 1-a\Delta t + \frac{a^2(\Delta t)^2}{2} & 0 \\ \frac{2ab(\Delta t)^2}{2} & b\Delta t - \frac{b^2(\Delta t)^2}{2} & a\Delta t - \frac{a^2(\Delta t)^2}{2} & 1 \end{bmatrix}$$

$$E_{M_2} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 4\varepsilon & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix}$$

Again, the diagonals have a relative error of approximately 1ε because they are the difference between two widely varying numbers and by the error equation for subtraction, the resulting error is approximately 1ε. The off-diagonals have errors greater than 1ε because they are the product of various components. Most of the off diagonals are the difference of two numbers and the error of the greater component dominates the error calculation.

The roundoff errors for the base matrix using the Padé approximations are slightly different than those evidenced with Taylor series approximations. The error matrix for the

numerator and the error matrix for the denominator would be the same as the error matrix for a Taylor series base matrix because they vary only by some multiplicative constants. The Padé method requires an inversion and then a matrix multiplication to determine the base matrix. The error of the inversion process will be discussed in Chapter 6, when the integration error of the Padé method is given. For roundoff error calculations, however, we will need to consider the additional matrix multiply. Using the roundoff error equations, this would give the following base matrix for a M_{22} base matrix.

$$E_{M_{22}} = \begin{bmatrix} 2\varepsilon & 0 & 0 & 0 \\ 3\varepsilon & 2\varepsilon & 0 & 0 \\ 3\varepsilon & 0 & 2\varepsilon & 0 \\ 5\varepsilon & 3\varepsilon & 3\varepsilon & 2\varepsilon \end{bmatrix}$$

Essentially, Padé base matrices have larger roundoff errors because of the additional matrix multiplication required.

Obviously, the error in the base matrix depends on the base matrix used. It also depends on the actual values that are used. Because of this uncertainty, the matrix of errors for M_1 will be used as a first approximation to the actual base matrix of errors. This approximation is then used to help determine the proper solution technique for the system. Depending on the solution technique implemented, the base matrix roundoff errors may or may not be significant. Nevertheless, these errors should be calculated and then used to compute the actual roundoff error once the solution technique is known. This way it can be assured the accuracy requirements for the system are met. For the rest of the roundoff error results in this chapter, the base matrix M_1 and its associated error matrix are assumed to be used.

4.3.2 Stepping Algorithms

By examining the actual mathematical operations that take place in the stepping algorithm, we can see how the initial roundoff error in the base matrix and the state vector propagate and affect the solution. We apply the roundoff error equations that were developed in section 4.1 where they are appropriate in the stepping algorithm to calculate the error and track the propagation. It is desirable to detect a pattern or derive an equation that would describe the error propagation so as to be able to predict the roundoff error without executing the stepping algorithm. Such an error estimate will be essential to the selection of the proper time step for the stepping algorithm.

For each stage of the stepping algorithm, the base matrix is multiplied by the state vector to produce a new state vector. This new state vector is then used in the next step to multiply the base matrix. Obviously then, we must determine the error incurred in the matrix-times-vector operation. We will use the four state M_1 given above and a typical initial state vector to demonstrate the error propagation of the stepping algorithm.

$$M_1 = \begin{bmatrix} 1 - (a+b)\Delta t & 0 & 0 & 0 \\ a\Delta t & 1 - b\Delta t & 0 & 0 \\ b\Delta t & 0 & 1 - a\Delta t & 0 \\ 0 & b\Delta t & a\Delta t & 1 \end{bmatrix} \qquad P(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The associated error matrices are

$$E_{M_1} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix} \qquad E_{P(0)} = \begin{bmatrix} 1\varepsilon \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

If we compute the first step and calculate the associated errors we get

$$P(\Delta t) = M_1 P(0)$$

$$P(\Delta t) = \begin{bmatrix} (1)(1) - (a+b)\Delta t \\ a\Delta t \\ b\Delta t \\ 0 \end{bmatrix} \qquad E_{P(\Delta t)} = \begin{bmatrix} 2\varepsilon \\ 3\varepsilon \\ 3\varepsilon \\ 0 \end{bmatrix}$$

The first term of the state vector is the product of two terms each with an error of 1ε. By the roundoff error equation for multiplication, the error of the product is the sum of the individual terms errors. The second and third terms are the product of two terms, one with an error of 1ε and the other with an error of 2ε. Thus the product has an error of 3ε.

We compute the second step and the errors so as to have three examples from which to draw a pattern. The state vector and its error vector after the second step are

$$P(2\Delta t) = \begin{bmatrix} (1 - (a+b)\Delta t)(1 - (a+b)\Delta t) \\ (1 - (a+b)\Delta t)(a\Delta t) + (1 - b\Delta t)(a\Delta t) \\ (1 - (a+b)\Delta t)(b\Delta t) + (1 - a\Delta t)(b\Delta t) \\ (a\Delta t)(b\Delta t) + (a\Delta t)(b\Delta t) \end{bmatrix} \qquad E_{P(2\Delta t)} = \begin{bmatrix} 3\varepsilon \\ 4\varepsilon \\ 4\varepsilon \\ 5\varepsilon \end{bmatrix}$$

This time the first element is a product of two terms, one with an error of 2ε and one with an error of 1ε. The second and third elements are products of terms with errors of 1ε and 3ε, resulting in a relative error of 4ε. The fourth element is the product of terms with errors

of 2ε and 3ε, resulting in a relative error of 5ε. The summations in the calculations of the second and third terms have little effect on the error since the terms are roughly the same value.

It is clear that a pattern is developing. The diagonals of the transition matrix are the dominant values in the both the rows and columns and also have the least amount of error. For the matrix-times-vector operation, a row of the matrix is multiplied by the state vector. This operation is dominated by the diagonals of the matrix. Since when two numbers are multiplied, their errors add, the diagonals add an error of 1ε each multiplication. Thus at every step, the error in each of the terms in the state vector increases by 1ε. There are other terms that may be added in, but since the diagonal is the dominant value in the matrix its error usually dominates the error propagation.

The pattern implies that the error increases 1ε for every step in the algorithm. Thus, an equation for the error for the stepping routine is

$$\text{RRE (stepping)} = n * \varepsilon$$

where n is the number of steps and ε is the machine precision. Alternatively, the equation can be written in terms of the time step (Δt) and the final time (t_{max}).

$$\text{RRE (stepping)} = \frac{t_{max}}{\Delta t} * \varepsilon$$

Both of these equations could be modified to account for the initial errors in the state vector, but since these are usually small, they are negligible. This equation implies that the relative error after 256 steps should be 256ε. For the above four-state example, the actual roundoff errors would be

$$E_{P(256\Delta t)} = \begin{bmatrix} 256\varepsilon \\ 257\varepsilon \\ 257\varepsilon \\ 258\varepsilon \end{bmatrix}$$

It can be seen that the roundoff errors converge to the value given by the prediction above.

Although the example given above is a non-cyclical model, the propagation pattern for the roundoff error is applicable to cyclical models as well. This can be seen in that the same mathematical operations are done in both cases. Additionally, the summations generally do not significantly affect the error because the terms are of similar orders of

magnitude. The slight differences that may occur are dwarfed by the error effects of the stepping routine.

Given below is an example of the relative roundoff error propagation patterns using a stepping routine with M_1 as the base matrix. The example given above was used as the model with the following system parameters. It is given on a log by log plot.

$a = 10^{-3}$ /hour; $b = 10^{-4}$ /hour; $tmax = 2x10^4$ hours;
$n = 100$ steps; $\Delta t = 200$ hours

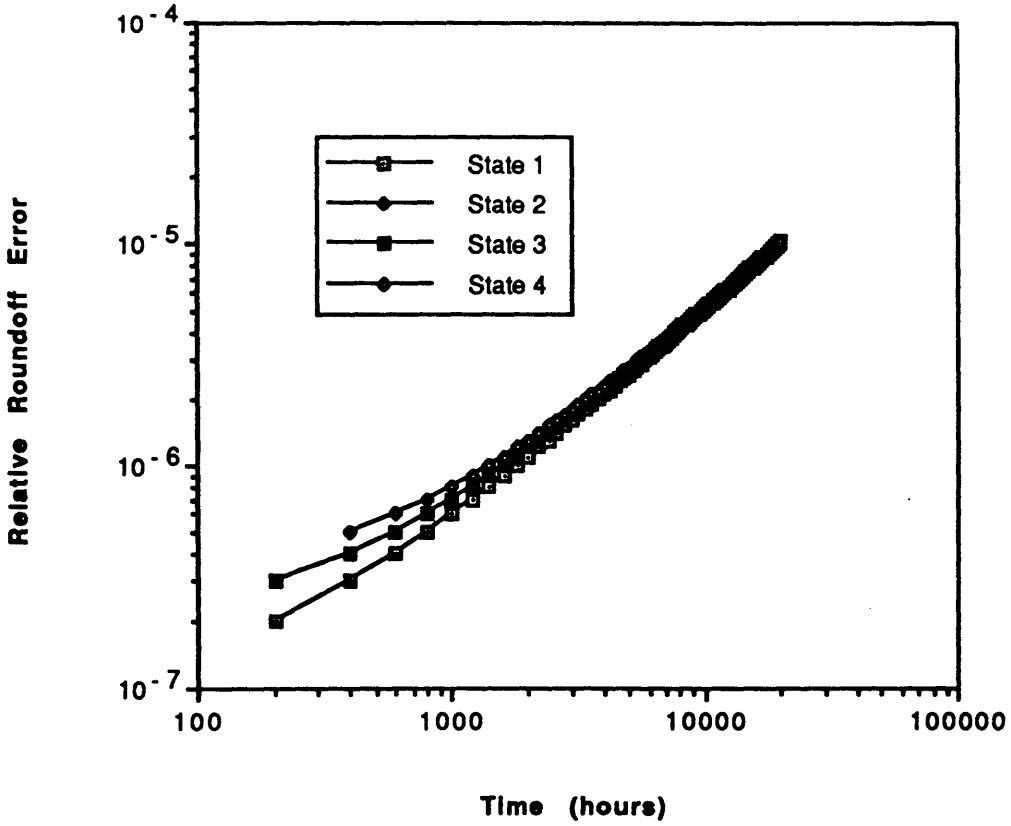


Figure 4-2: The Relative Roundoff Error using a Stepping Algorithm

Notice that the roundoff errors quickly converge to the same value. Also, the error grows directly as the number of steps increases. Clearly, the graph agrees with the roundoff error equation developed.

Consequently, an equation that determines the relative roundoff error for the stepping algorithm has been developed for both non-cyclical and cyclical models. This equation allows for the *a priori* prediction of the roundoff error given a time step. Accordingly, the equation will be essential in determining the solution technique and time step that will produce an acceptable error with a minimum amount of computer work.

4.3.3 Squaring Algorithms

In a manner similar to the stepping algorithms, we apply the roundoff error equations of section 4.1 to the mathematical operations in the squaring algorithm. Again, it is desirable to detect a trend or derive an equation that describes that error propagation pattern associated with the squaring algorithms. This error equation will be used to determine *a priori* the time step that produces an acceptable error with the minimum computer work.

In the squaring algorithm, we have a series of matrix-times-matrix operations with a single matrix-times-vector operation to compute the final state vector. Obviously, the error propagation of these operations need to be understood. In contrast to the stepping algorithm, squaring operates on the matrix. Thus, the relative errors of the transition matrix change while the relative errors of the state vector remain the same. Again, we will use the four-state M_1 base matrix to demonstrate the error propagation. The base matrix M_1 and its error matrix are repeated below for convenience.

$$M_1 = \begin{bmatrix} 1 - (a+b)\Delta t & 0 & 0 & 0 \\ a\Delta t & 1 - b\Delta t & 0 & 0 \\ b\Delta t & 0 & 1 - a\Delta t & 0 \\ 0 & b\Delta t & a\Delta t & 1 \end{bmatrix} \qquad E_{M_1} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix}$$

If M_1 is now squared as in the first squaring of the algorithm, the transition matrix and its error matrix become

$$M_1^2 = \begin{bmatrix} (1 - (a+b)\Delta t)(1 - (a+b)\Delta t) & 0 & 0 & 0 \\ (a\Delta t)[(1 - (a+b)\Delta t) + (1 - b\Delta t)] & (1 - b\Delta t)(1 - b\Delta t) & 0 & 0 \\ (b\Delta t)[(1 - (a+b)\Delta t) + (1 - a\Delta t)] & 0 & (1 - a\Delta t)(1 - a\Delta t) & 0 \\ (a\Delta t)(b\Delta t) + (a\Delta t)(b\Delta t) & (b\Delta t)[(1 - b\Delta t) + 1] & (a\Delta t)[(1 - a\Delta t) + 1] & 1 \end{bmatrix}$$

$$E_{M_1^2} = \begin{bmatrix} 2\varepsilon & 0 & 0 & 0 \\ 3\varepsilon & 2\varepsilon & 0 & 0 \\ 3\varepsilon & 0 & 2\varepsilon & 0 \\ 4\varepsilon & 3\varepsilon & 3\varepsilon & 2\varepsilon \end{bmatrix}$$

The diagonals have errors of 2ε because they are the product of two terms each with an error of 1ε. The terms with errors of 3ε are the product of two terms, one with an error of 1ε and the other with an error of 2ε. The one term with the error of 4ε is the product of terms with errors of 2ε. Because many of the terms are of the same order of magnitude, the summation of these terms does not have a great effect on the roundoff error.

If we continue the squaring process so as to better detect a pattern, we get a transition matrix that is not very insightful when written symbolically. However, the error matrix is easily determined and of primary interest.

$$E_{M_1^4} = \begin{bmatrix} 4\varepsilon & 0 & 0 & 0 \\ 5\varepsilon & 4\varepsilon & 0 & 0 \\ 5\varepsilon & 0 & 4\varepsilon & 0 \\ 6\varepsilon & 5\varepsilon & 5\varepsilon & 4\varepsilon \end{bmatrix}$$

The diagonal terms have errors of 4ε because they are now the product of two terms, each with errors of 2ε. The terms with errors of 5ε are primarily the product of two terms with errors of 2ε and 3ε. Lastly, the term with an error of 6ε is the product of two terms with errors of 3ε. Again, the summations do not have a great effect on the error propagation pattern since the terms are of similar order in magnitude.

Clearly, a pattern for the propagation of the errors exists with the squaring algorithm as well. With each squaring of the matrix, the relative error of each diagonal element is doubled since the diagonal element is multiplied by itself. Additionally, the diagonal element is multiplied by the other elements in the column. Thus, these errors also add. Overall, the errors of the matrix approximately double for each squaring of the matrix. Since the power of the matrix doubles for each squaring of the matrix, the errors increase directly as the power of the matrix increases. This observation leads to the following equation for the propagation of roundoff error using squaring methods.

$$\text{RRE (squaring)} = n * \varepsilon$$

Rewriting this equation in terms of the time step yields

$$\text{RRE (squaring)} = \frac{t_{max}}{\Delta t} * \varepsilon$$

Again, the equations could be slightly modified to account for the initial differences in the error of the base matrix, but these initial differences are usually negligible. The implication

is that that the error after eight squarings of the matrix would converge to 256ϵ ($2^8 = 265$). The actual error matrix after eight squarings is very close to the expected result.

$$E_{M_1^{256}} = \begin{bmatrix} 256\epsilon & 0 & 0 & 0 \\ 257\epsilon & 256\epsilon & 0 & 0 \\ 257\epsilon & 0 & 256\epsilon & 0 \\ 258\epsilon & 257\epsilon & 257\epsilon & 256\epsilon \end{bmatrix}$$

Once the transition matrix for the final time has been determined, it must be multiplied by the initial state vector to determine the state vector at the final time. Most often the initial condition is a probability of one in the first state. If this is the case, the final state vector is the first column of the transition matrix and no additional roundoff error need be incurred. On the other hand, if there is a general initial state vector, the additional matrix-times-vector will incur some additional roundoff error. The additional error, however, will normally be negligible compared to the rest of the error matrix.

The equations given above are again applicable to cyclical models as well as non-cyclical ones. The error propagation is dominated by the diagonal element error. This error doubles regardless of the cyclical nature of the model since it is always multiplied by itself. In a cyclical model, many of the terms are multiplied by other terms with comparable errors, resulting in the same doubling pattern.

It is interesting to note that the roundoff error propagation pattern for the squaring routine is approximately the same as the pattern for the stepping routine. The differences are negligible. Thus, a stepping routine does not have any advantage over a squaring routine in terms of roundoff error, both routines will yield the same accuracy. The squaring routine, however, still has the advantage that it usually requires less computer work. This also implies that the roundoff error will be the same for a combined stepping/squaring algorithm as it would be for an algorithm using one or the other.

The figure below gives the relative roundoff error for the four states of the model used in the example with the following system parameters. A squaring algorithm with M_1 as the base matrix was used. Again, the data is presented on a log by log plot.

$a = 10^{-3}$ /hour; $b = 10^{-4}$ /hour; $tmax = 2x10^{4}$ hours;
$n = 33{,}554{,}432$ (2^{25}) steps; $\Delta t = 5.960x10^{-4}$ hour

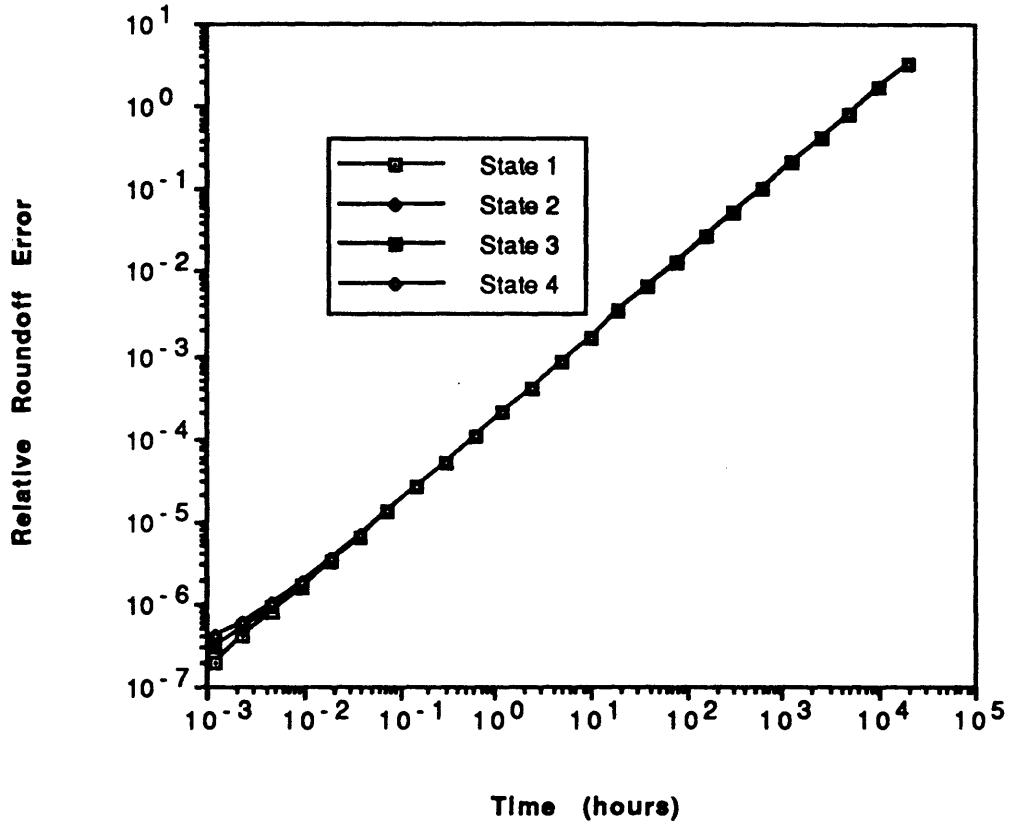


Figure 4-3: The Relative Roundoff Error using a Squaring Algorithm

Again, the error patterns for the various states quickly converge to the same result. The overall patterns goes as the number of states even though the squaring algorithm was used. The pattern evidenced on the graph agrees with the error propagation equation developed.

Clearly, a pattern for the propagation of errors in a squaring technique has been developed. This pattern is applicable to a general cyclical and a general non-cyclical Markov model. The error equation can be used to *a priori* bound the roundoff error. This information can then be combined with the integration error data to determine a time step and integration methodology that results in an acceptable error while minimizing computer work.

4.3.4 Stepping Algorithms with Variable Renormalization

As it was stated before, the variable renormalization technique does not work very well with stepping operations. In a stepping algorithm, the state vector is multiplied by the base matrix. The largest element in the state vector is then determined through renormalization of the state vector. The relative errors in the base matrix are unaltered by the renormalization process as well as the other elements in the state vector. Because variable renormalization only affects one element in the state vector, it is relatively ineffective in a stepping algorithm.

It was shown above that after each matrix-times-vector operation, the errors in the state vector increase by 1ε yielding a final roundoff error of approximately $n\varepsilon$. We will now take the same example as in section 4.3.2 using renormalization. The base matrix and initial state vector and the associated error matrices are repeated here for convenience

$$M_1 = \begin{bmatrix} 1-(a+b)\Delta t & 0 & 0 & 0 \\ a\Delta t & 1-b\Delta t & 0 & 0 \\ b\Delta t & 0 & 1-a\Delta t & 0 \\ 0 & b\Delta t & a\Delta t & 1 \end{bmatrix} \qquad P(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$E_{M_1} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix} \qquad E_{P(0)} = \begin{bmatrix} 1\varepsilon \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The state vector and its error vector after the first time step are

$$P(\Delta t) = \begin{bmatrix} 1-(a+b)\Delta t \\ a\Delta t \\ b\Delta t \\ 0 \end{bmatrix} \qquad E_{P(\Delta t)} = \begin{bmatrix} 2\varepsilon \\ 3\varepsilon \\ 3\varepsilon \\ 0 \end{bmatrix}$$

We can now renormalize the first element (the largest) in the state vector.

$$P_1(\Delta t) = 1 - [P_2(\Delta t) + P_3(\Delta t) + P_4(\Delta t)]$$

$$P_1(\Delta t) = 1 - [a\Delta t + b\Delta t]$$

Clearly, in symbolic terms, this is the quantity we already had for the first element. In terms of the relative error, however, there is a difference. Calculating the relative error (RE) of the first element, we get.

$$RE(sum) = \frac{(a\Delta t)\,3\varepsilon + (b\Delta t)\,3\varepsilon}{(a\Delta t) + (b\Delta t)} = \frac{(a\Delta t) + (b\Delta t)}{(a\Delta t) + (b\Delta t)}\,3\varepsilon = 3\varepsilon$$

$$RE(P_1(\Delta t)) = \frac{(1)\,1\varepsilon + (a\Delta t + b\Delta t)\,3\varepsilon}{1 - (a\Delta t + b\Delta t)} \approx 1\varepsilon$$

Thus, variable renormalization will reduce the error of the first (largest) element in the state vector provided that the other elements are small by comparison. The renormalized state vector and its associated state vector are, where the renormalized vectors are designated with a superscript R.

$$P(\Delta t)^R = \begin{bmatrix} 1 - (a+b)\Delta t \\ a\Delta t \\ b\Delta t \\ 0 \end{bmatrix} \qquad E^R_{P(\Delta t)} = \begin{bmatrix} 1\varepsilon \\ 3\varepsilon \\ 3\varepsilon \\ 0 \end{bmatrix}$$

Unfortunately, if we calculate the state vector at the end of the second time step, we see no appreciable difference in the errors of the state vector save the element being renormalized.

$$P(2\Delta t) = \begin{bmatrix} (1 - (a+b)\Delta t)\,(1 - (a+b)\Delta t) \\ (1 - (a+b)\Delta t)\,(a\Delta t) + (1 - b\Delta t)\,(a\Delta t) \\ (1 - (a+b)\Delta t)\,(b\Delta t) + (1 - a\Delta t)\,(b\Delta t) \\ (a\Delta t)\,(b\Delta t) + (a\Delta t)\,(b\Delta t) \end{bmatrix} \qquad E_{P(2\Delta t)} = \begin{bmatrix} 2\varepsilon \\ 4\varepsilon \\ 4\varepsilon \\ 5\varepsilon \end{bmatrix}$$

We can again renormalize the state vector to get a relative roundoff error of 1ε in the first element of the state vector But, as can be seen, the renormalization process does not affect the other elements in the state vector.

$$E^R_{P(2\Delta t)} = \begin{bmatrix} 1\varepsilon \\ 4\varepsilon \\ 4\varepsilon \\ 5\varepsilon \end{bmatrix}$$

Despite the apparent failure of variable renormalization to reduce most of the roundoff errors in the state vector, one might be tempted to use the stepping routine with variable renormalization if only interested in the probability of the first state. Indeed, this would be an acceptable approach in certain cases. The renormalization process, however, only works on the largest element in the state vector. Thus, the first element will be renormalized only until the characteristic time. At the characteristic time, an element other than the first one will be the largest element in the vector. This element will then begin to be renormalized. The error of the first element, on the other hand, reverts to an increase of

1ε for every step. Clearly then, for systems run past the characteristic time, variable renormalization in a stepping routine may yield large roundoff errors.

Shown below is a graph of the errors associated with the four-state model that has been used many times thus far. The system parameters are listed below. In a manner similar to the previous results, a base matrix of M_1 was used, and the data is presented on a log by log plot.

$a = 10^{-3}$ /hour; $b = 10^{-4}$ /hour; tmax = $2x10^4$ hours;
n = 100 steps; $\Delta t = 200$ hours

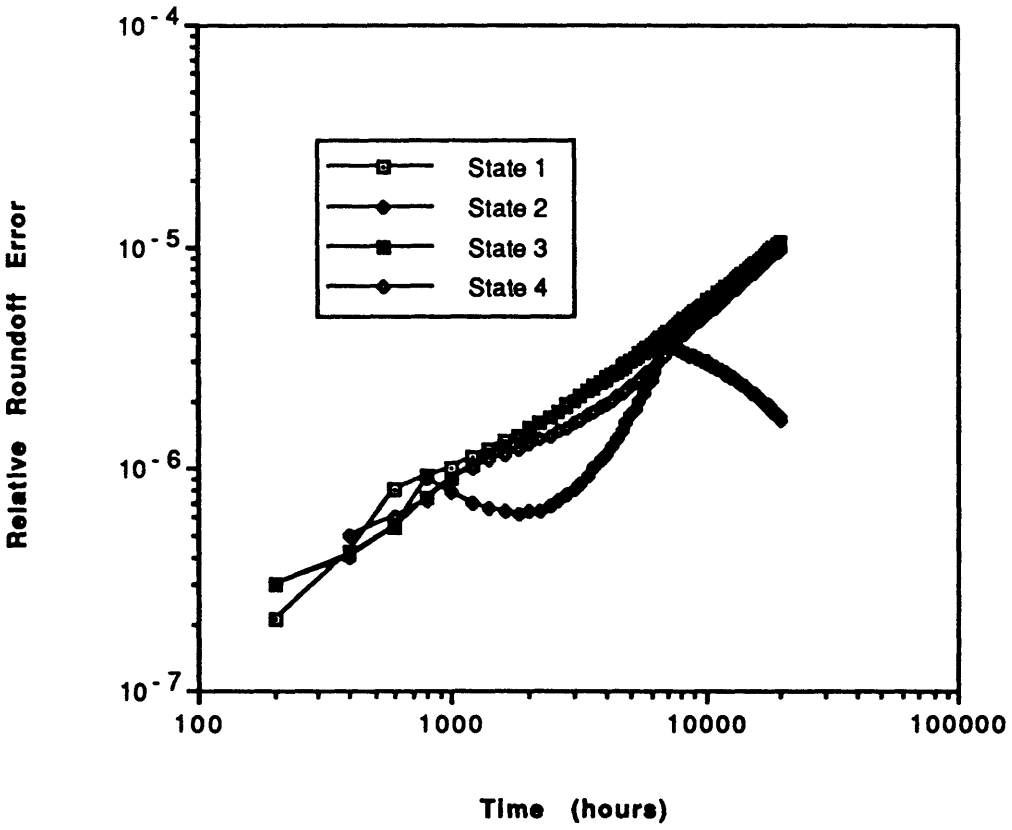


Figure 4-4: The Relative Roundoff Error using a Stepping Algorithm with Variable Renormalization

Notice that the first state is initially growing at a slower rate until the characteristic time is reached at approximately 1000 hours. After this point, the first element is no longer being

renormalized, as is evidenced by the slight decrease in the error of state 2 which is now being renormalized. State 4 eventually becomes the renormalized element around 10,000 hours because the system is a pure decay model. Overall, the variable renormalization does not have a significant effect on the propagation of the roundoff error.

The results shown here for the stepping routine with variable renormalization are similar to what would be expected for the roundoff error given a cyclical model. There are some slight differences, however, that can easily be understood. Since the states of a cyclical model are allowed to have non-zero steady-state probability, the element that is renormalized may be any of the elements in the state vector. In fact, if the largest two of the steady-state values are close in magnitude, renormalization does not significantly reduce the error of either element. At this point, renormalization has no effect on the error propagation. Additionally, steady-state may be reached before the characteristic time, in which case the first element would be the only one to be renormalized. Regardless of the renormalization, the elements in the state vector that are not being renormalized will still increase at a rate of 1ε for every step. As such, cyclical Markov models and non-cyclical models have essentially the same relative roundoff error propagation patterns in a stepping routine with variable renormalization.

As we did for the other cases, we wish to express the relative roundoff error propagation as a function of the system parameters. Clearly before the characteristic time, the propagation of the errors for all the states save the renormalized one is the same as without renormalization. After the characteristic time, which element is renormalized depends on the transition rates and on the time and cannot be accurately determined. The uncertainty of the variable renormalization causes the upper bound on the roundoff error to be conservative. Since we are looking for an upper bound on all the relative errors in the system, we will use error propagation bound that was used for a stepping algorithm without variable renormalization.

$$\text{RRE (stepping with variable renormalization)} = n * \varepsilon$$

$$\text{RRE (stepping with variable renormalization)} = \frac{t_{max}}{\Delta t} * \varepsilon$$

We have an equation that is an upper bound for the relative roundoff error associated with a stepping routine using variable renormalization. The equation developed yields the same result as either a stepping or squaring routine without variable renormalization. Yet, the variable renormalization may produce slightly more accurate

results. Nevertheless, in terms of determining the amount of error and the integration time step, variable renormalization is of little benefit to the stepping routines. The real benefit of the variable renormalization technique will be witnessed when it is applied to the squaring routines in the next section.

4.3.5 Squaring Algorithms with Variable Renormalization

The real benefit of variable renormalization is evidenced in the squaring algorithms. In contrast to the stepping algorithms, the base matrix is multiplied by itself to propagate it through time. The variable renormalization is therefore applied to the transition matrix and not the state vector. The largest element in each column of the state vector is determined through renormalization. In this manner, the relative errors of all elements of the transition matrix are greatly affected by renormalization process, making it an effective routine for reducing roundoff error.

It was shown in Section 4.3.3 that the relative roundoff error of a squaring algorithm without variable renormalization increases with the power of the squarings. This pattern results in an amount of error equivalent to the stepping routine without variable renormalization--nε. However, when variable renormalization is applied, the roundoff error can be significantly reduced.

We will again use the simple example given before in Section 4.3.2. In this case, however, we will demonstrate the effectiveness of variable renormalization applied to a squaring procedure. The base matrix and its error matrix are

$$M_1 = \begin{bmatrix} 1 - (a+b)\Delta t & 0 & 0 & 0 \\ a\Delta t & 1 - b\Delta t & 0 & 0 \\ b\Delta t & 0 & 1 - a\Delta t & 0 \\ 0 & b\Delta t & a\Delta t & 1 \end{bmatrix} \qquad E_{M_1} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 2\varepsilon & 1\varepsilon & 0 & 0 \\ 2\varepsilon & 0 & 1\varepsilon & 0 \\ 0 & 2\varepsilon & 2\varepsilon & 1\varepsilon \end{bmatrix}$$

The transition matrix and error matrix after one squaring are

$$M_1^2 = \begin{bmatrix} (1 - (a+b)\Delta t)\,(1 - (a+b)\Delta t) & 0 & 0 & 0 \\ (a\Delta t)\,[(1 - (a+b)\Delta t) + (1 - b\Delta t)] & (1 - b\Delta t)\,(1 - b\Delta t) & 0 & 0 \\ (b\Delta t)\,[(1 - (a+b)\Delta t) + (1 - a\Delta t)] & 0 & (1 - a\Delta t)\,(1 - a\Delta t) & 0 \\ (a\Delta t)\,(b\Delta t) + (a\Delta t)\,(b\Delta t) & (b\Delta t)\,[(1 - b\Delta t) + 1] & (a\Delta t)\,[(1 - a\Delta t) + 1] & 1 \end{bmatrix}$$

$$E_{M_1^2} = \begin{bmatrix} 2\varepsilon & 0 & 0 & 0 \\ 3\varepsilon & 2\varepsilon & 0 & 0 \\ 3\varepsilon & 0 & 2\varepsilon & 0 \\ 4\varepsilon & 3\varepsilon & 3\varepsilon & 2\varepsilon \end{bmatrix}$$

We can now normalize the diagonal element in each of the columns of the transition matrix. The diagonal element is being renormalized here under the assumption that it is the largest element in the column. We will see later where this assumption is no longer valid.

$$M_I^2(1,1) = 1 - [M_I^2(2,1) + M_I^2(3,1) + M_I^2(4,1)] = 1 - (\text{sum})$$

$$\text{RE (sum)} = \frac{M_I^2(2,1)\ 3\varepsilon + M_I^2(3,1)\ 3\varepsilon + M_I^2(4,1)\ 4\varepsilon}{M_I^2(2,1) + M_I^2(3,1) + M_I^2(4,1)} \approx 3\varepsilon$$

$$\text{RE}\,(M_I^2(1,1)) = \frac{(1)\ 1\varepsilon + (\text{sum})\ 3\varepsilon}{1 - (\text{sum})} \approx 1\varepsilon$$

The same process would be repeated for the other three columns. Thus, in a manner similar to that shown in the previous section, the roundoff error of the renormalized diagonal elements is approximately 1ε. After renormalization, then, the error matrix would be

$$E^R_{M_I^2} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 3\varepsilon & 1\varepsilon & 0 & 0 \\ 3\varepsilon & 0 & 1\varepsilon & 0 \\ 4\varepsilon & 3\varepsilon & 3\varepsilon & 1\varepsilon \end{bmatrix}$$

The above matrix shows only a slight improvement over the error matrix without variable renormalization. However, the promise of improved performance can be readily seen. When a matrix is squared, each element in the squared matrix has two terms that were multiplied by a diagonal. When two values are multiplied together, their relative roundoff errors add. Thus, with the diagonals having an error of only 1ε, the increase of the roundoff error is only 1ε. Additionally, once the values are multiplied together, they must be added to determine the new element of the squared matrix. It was shown previously that the addition of two numbers favors the relative error of the larger one. This relation further increases the influence of the diagonal since it nominally is the largest element in the column. The error propagation pattern we expect, therefore, is an increase of 1ε for every squaring of the matrix instead of every step in the solution.

A few examples of the error matrix for different levels of squaring are given below to demonstrate the effects of the variable renormalization.

$$E^R_{M_I^4} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 4\varepsilon & 1\varepsilon & 0 & 0 \\ 4\varepsilon & 0 & 1\varepsilon & 0 \\ 5\varepsilon & 4\varepsilon & 4\varepsilon & 1\varepsilon \end{bmatrix}$$

$$E^{R}_{M_1^{16}} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 6\varepsilon & 1\varepsilon & 0 & 0 \\ 6\varepsilon & 0 & 1\varepsilon & 0 \\ 7\varepsilon & 6\varepsilon & 6\varepsilon & 1\varepsilon \end{bmatrix}$$

$$E^{R}_{M_1^{256}} = \begin{bmatrix} 1\varepsilon & 0 & 0 & 0 \\ 10\varepsilon & 1\varepsilon & 0 & 0 \\ 10\varepsilon & 0 & 1\varepsilon & 0 \\ 11\varepsilon & 10\varepsilon & 10\varepsilon & 1\varepsilon \end{bmatrix}$$

Section 4.3.3 showed that the equivalent system without variable renormalization would have errors of 256 ε. Clearly, the variable renormalization greatly reduced the relative roundoff error.

An equation describing this error propagation pattern is easily seen. The relative error increases 1ε for every squaring of the matrix. Additionally, the base matrix starts out with some initial error. Lastly, we can see that the errors increase due to the failure level. This last effect is due to an initial multiplication where the renormalized elements do not have any effect. These three effects lead to the following equation for the relative roundoff error for scaling and squaring using variable renormalization.

$$\text{RE (squaring with varible renormalization)} = \left(\frac{\ln n}{\ln 2} + 1 + f \right) \varepsilon$$

where f is the failure level. Often the number of steps is large enough that the other two effects can be neglected, resulting in the simple equation

$$\text{RE (squaring with varible renormalization)} \approx \left(\frac{\ln n}{\ln 2} \right) \varepsilon$$

The pattern developed so far assumes that the diagonal is the largest element in the column. Unfortunately, this constraint is often violated at some point in the integration run. The point at which this constraint is violated is around the characteristic time of the system. At the characteristic time of the system, the probability of the first state is approximately one-half. This corresponds to the probability of the first diagonal being approximately one-half. Thus, close to this point, other states in the system (or elements in the column) contain the other one-half of the probability. Once another state has a larger probability value than the diagonal, its corresponding element in the column is being normalized.

The effects of an element other than the diagonal being normalized can be readily seen. Earlier, it was stated that the errors were kept low because the diagonals with errors

of 1ε were dominating the multiplication process. Despite the fact that the element being renormalized has switched, the diagonal elements still continue to dominate the multiplication process. Thus, as the errors of the diagonal grow, the errors of the rest of the matrix grow. It is no longer necessary that elements of the squared matrix be comprised of components that were normalized (diagonals). In essence, the benefits of the variable renormalization have been lost for the general matrix.

Once past the characteristic time, the system acts as though variable renormalization was no longer taking place. A return to the error doubling pattern described in Section 4.3.3 is evidenced. The diagonal has an initial error of 1ε, and for every squaring after the switching point, this error doubles. Eventually, the diagonals infect the other elements in the matrix (except the ones being normalized) to produce a converging pattern for the relative roundoff error.

The complete relative error propagation pattern is the combination of the two individual error patterns. The first pattern runs until the the characteristic time while the second pattern is in effect thereafter. Note that the second pattern begins the doubling pattern with the relative error that is evident in the system at the switching point. Thus, the equation for the relative roundoff error propagation is given in two parts, with n' indicating the switching point (n'Δt = char time).

$$\text{RRE (squaring with variable renormalization)} = \begin{cases} \dfrac{\ln n}{\ln 2}\,\varepsilon & \text{for } n\Delta t < \text{char time} \\ \\ (n - n')\left(\dfrac{\ln n'}{\ln 2}\right)\varepsilon & \text{for } n\Delta t > \text{char time} \end{cases}$$

Below is given an graph that demonstrates that roundoff error propagation patterns. It was generated using the non-cyclical four-state Markov model given previously. The two asymptotic patterns are clearly visible, as well as the transition area.

$a = 10^{-3}$ /hour; $b = 10^{-4}$ /hour; $t_{max} = 2 \times 10^{4}$ hours
$n = 33{,}554{,}432$ steps; $\Delta t = 5.960 \times 10^{-4}$ hour

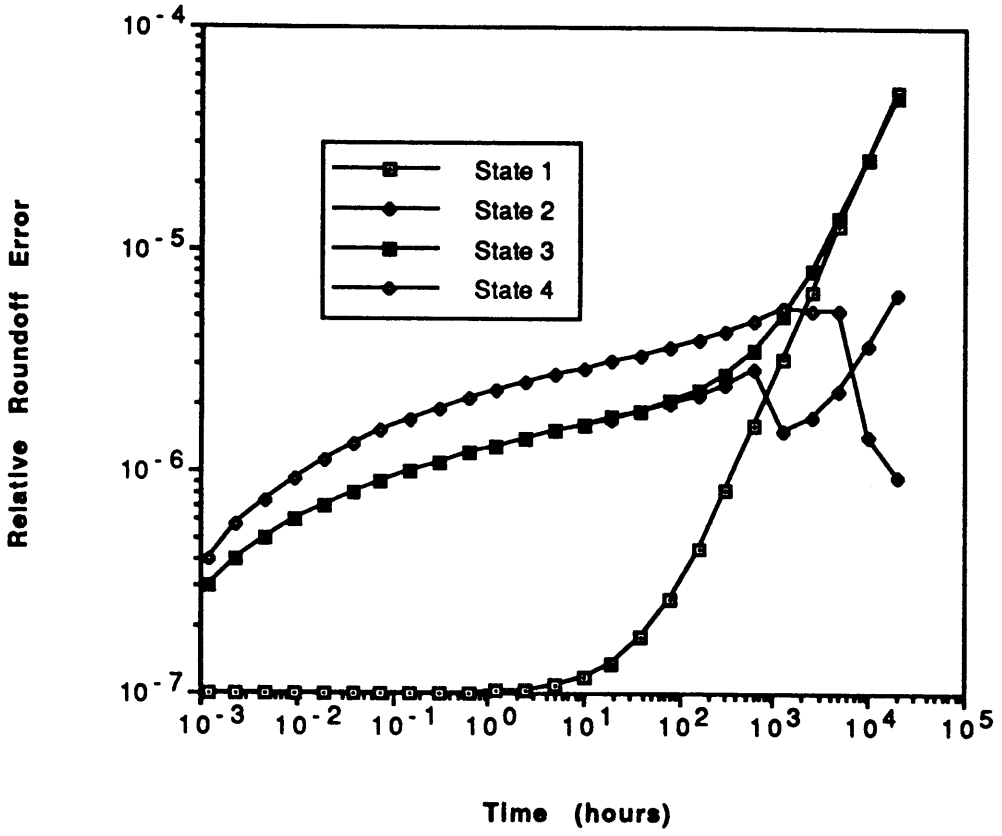


Figure 4-5: The Relative Roundoff Error using a Squaring Algorithm with Variable Renormalization

Obviously, the first state is initially being normalized so its error remains close to the machine precision (10^{-7}). But once the characteristic time is approached at 1000 hours, it is clear that a different state is being renormalized. Initially after the characteristic time, State-2 is being renormalized. However, State 4 eventually becomes the largest element, and so its roundoff error decreases accordingly. Also, after the characteristic time, the error doubling pattern of squaring without renormalization has returned. Again, the graph agrees with the developed equations.

It is important to note that the actual error propagation pattern is a smooth function between the two types of propagation. The definitions given above are asymptotic

approximations to the actual propagation path. There is a smooth transition period that joins the two asymptotic patterns. During this transition period, the two propagation patterns both have influence in the overall pattern. The amount of influence of either depends on the parameters of the system. To remain conservative, then, it may be desirable to consider the switching point at one or two squarings before the characteristic time. This rule of thumb will generally bound the errors in the transition period, but it also leads to relatively loose bound once in the error doubling portion. If the final time is near the characteristic time of the system, it is suggested that the conservative approach be used. On the other hand, if the final time is well into the doubling period the asymptotic approach should be sufficient.

Again, an equation that describes the error pattern for a general model using the solution technique of squaring with variable renormalization has been developed. The equation shows that variable renormalization may greatly reduce the roundoff error depending on the number of steps and the characteristic time of the system. Overall, scaling and squaring with variable renormalization is one of the most effective integration methods discussed. Nevertheless, there are times when the characteristic time is very short compared to the final time or many intermediate values of high accuracy are warranted. In these cases, it may be necessary to move to a stepping algorithm with an accumulator.

4.3.6 Stepping Algorithms with Accumulator Methods

The stepping method with accumulator is an attempt to increase the accuracy of the machine by accumulating the errors caused by truncation. These accumulated values are then added back in when they become significant. In this manner, the relative roundoff error is reduced. The result is known as quasi-double precision [Møller, 1965].

The roundoff error propagation pattern for the accumulator methods is difficult to exactly define. The roundoff errors initially grow as normally for the stepping algorithm. The accumulator is eventually added in to improve the accuracy. Whereas without the accumulator, the roundoff errors grow monotonically, with the accumulator the errors no longer are monotonic. The actual pattern is more jagged.

Because of the unusual nature of the accumulator algorithm, a general bound will not be developed. Instead, a 'rule-of-thumb' bound will be given. The accumulator, like the regular vectors, has an accuracy of ε. Thus, when it is added in, the most accurate the new value could possibly be is ε^2. However, the new value would quickly be represented

in single point precision, reducing the accuracy back to ε. Clearly, then, the accuracy once the accumulator is added in is approximately ε. The problem arises in the times between the additions of the accumulator. The roundoff error continues to grow, but it is difficult to determine how long since the last accumulator addition especially after the first one. Hence, the 'rule-of-thumb' is necessary.

Different approaches are possible. The approach developed here is to assume that the roundoff errors do not grow to more than 100ε between accumulator additions. The factor of 100 was chosen because of its convenience and its seemingly conservative nature. This allows for 100 steps between accumulator additions. Additionally, it allows for an accuracy of 10^{-5} to be reached on typical single precision machines, more than is usually desired. Other factors are acceptable, depending on the degree of conservatism desired. No matter what factor is desired, once the accumulator is added in, the relative roundoff error is assumed to return to 1ε.

The accumulator holds off the encroachment of roundoff error until number of steps reaches the reciprocal of the machine precision. At this point, the roundoff error of the accumulator has completely corrupted the accumulator vectors. The values being added in are now suspect. Thus from this point on, the accumulator is no longer providing any additional information to the integration process. The normal roundoff error propagation pattern of Section 4.3.2 applies from here to the final time. Again, depending on the degree of conservatism, this point of inactivity for the accumulator can be changed.

We can now construct a rule-of-thumb roundoff error equation for the stepping algorithm with an accumulator.

$$\text{RRE (stepping with accumulator)} = \begin{cases} 100\,\varepsilon & \text{for } n < \dfrac{1}{\varepsilon} \\ n\varepsilon - 1 & \text{for } n > \dfrac{1}{\varepsilon} \end{cases}$$

It is important to note that the above equation is not a bound on the relative roundoff error. It is a good approximation to the roundoff error patterns actually experienced. Depending on the degree of conservatism desired, both the factor of 100 and the point at which the accumulator is no longer active may be altered.

An equation that describes the relative roundoff error using a stepping algorithm with accumulator has been developed. The equation is not a bound, however. The

accumulator, while accurate, is difficult to fully describe. Because of this uncertainty and the additional work required with an accumulator, it should be used only when necessary.

4.4 Recapitulation

Chapter 4 dealt with the notion of roundoff error. The specific goals of the chapter were to define roundoff error and its propagation patterns; to investigate methods that would reduce the amount of roundoff error incurred; and to develop equations that predict the roundoff error for various solution techniques.

The first task at hand was to define roundoff error and to develop useful equations that describe the roundoff error propagation patterns. Equations that calculate the roundoff error for the basic mathematical operations of addition, multiplication, and subtraction were developed. These equations were expressed in terms of relative error. It was shown that when two numbers are multiplied, their errors are added. It was also shown that when two numbers are the added or subtracted, the resulting error is a weighted average of the component errors.

After the propagation patterns were defined for the general mathematical operations, two roundoff error reduction techniques were introduced. The first method, variable renormalization, exploits the concepts of the conservation of system probability and the time-invariant nature of the transition matrix. The renormalization method recalculates the largest element in the each column of the transition matrix or perhaps in the state vector by summing the other elements in the column and subtracting the sum from unity. The effect of recalculating the largest element is to reduce its error to approximately that of the machine's precision. The reduction of the error of the largest element then propagates throughout the other elements in the transition matrix.

The second error reduction technique introduced was the accumulator method. Basically, additional vectors are created that accumulate the values that are lost through truncation. Eventually, the values that were truncated become significant and are added back in. The effect is essentially creating a quasi-double precision because the accumulator has a machine precision equivalent to that of the state vector. Accumulator methods are most easily used with stepping algorithms whereas variable renormalization is more effective with squaring routines.

Once the error reduction techniques were described, the results of the roundoff error study were presented. The results were generated using the roundoff error equations

for the mathematical operations. First, the roundoff error for the base matrix was given. It was shown that the base matrix errors are essentially the same regardless of the power of the truncated power series.

The errors of the base matrix were then propagated using the stepping and squaring algorithms. The analysis showed that both routines yielded identical amounts of roundoff error. The roundoff error went as the number of steps in the integration procedure times the machine precision.

The error propagation patterns were also developed for both stepping and squaring with variable renormalization and for the stepping with accumulator. The results showed that stepping with variable renormalization showed no appreciable decrease in roundoff error except for the element being renormalized. On the other hand, squaring with variable renormalization showed a significant decrease in roundoff error for most cases. The roundoff error went as the number of squarings (instead of the number of steps) during the integration run times the machine precision. Unfortunately, the reduced propagation pattern only lasts until the characteristic time of the system, after which the roundoff error doubling pattern again returns. Lastly, the accumulator method produced a decrease in the roundoff error for a stepping routine. While never actually producing double precision accuracy, the encroachment of the roundoff error was held off to produce an effect that looked like double precision.

In closing, the roundoff error equations that describe the propagation patterns for the various integration techniques have been developed. The amount of roundoff error can be predicted *a priori* given the integration parameters and the system parameters. The first phase of the project has been completed. The second phase, integration error, begins in the next chapter with a discussion on the integration errors experienced when using Taylor series approximations.

Chapter 5

Integration Error--Taylor Series Approximations

In addition to roundoff error, the major contributor to numerical error is integration error. While roundoff error is primarily due to the precision of the machine being used, integration error is the result of the chosen integration scheme. This chapter explores the integration error that results from using the Taylor series method. It also investigates a method of reducing integration error known as Richardson extrapolation. Lastly, it gives numerical results to demonstrate the integration error propagation patterns for the various techniques.

It was convenient to discuss roundoff error in terms of the relative error. This is due to the roundoff error being independent of most system parameters, depending instead on the number of mathematical operations preformed. As such, roundoff error was independent of the final values of the probability. Integration error, on the other hand, is difficult to express in terms of relative error. It is highly dependent on the system transition matrix, and thus, on the final probabilities. The equations developed in this chapter bound the absolute integration error for the system as a whole. Because of this dependence on the final probability, the relative error of a state will vary greatly with the state probabilities. Therefore, it is more convenient when discussing integration error to speak in terms of absolute error. Absolute error will be used for most of this chapter. However, approximations for the relative error of individual states of the Markov model will be developed using the chain model approximations given in Section 2.6.

5.1 Sources of Integration Error

Contrary to roundoff error, integration error is not the result of finite word space on the computer. Rather, integration error is caused by the approximation to the matrix exponential (see Chapter 3). The exact solution to the matrix exponential is an infinite power series written in continuous time.

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \ldots$$

It was shown in Chapter 3 that the calculation of the matrix exponential in this form was impractical and often leads to highly inaccurate results.

The more efficient way to compute the matrix exponential is to use a truncated power series.

$$e^{At} \approx I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!}$$

This method reduces the amount of work necessary to compute the matrix exponential. However, it is clear that the truncated power series introduces integration error. The higher order terms in the power series that are not included in the approximation will yield the error caused by the truncation. This method also poses the problem of selecting the number of terms to include before truncating the series. Unfortunately, even a properly truncated power series may produce gross errors due to catastrophic cancellation (see section 3.2).

To supplement the method of the truncated power series, the idea of scaling and propagating is used. Scaling and propagating utilizes a fundamental property of the exponential function, namely that

$$e^{At} = [e^{A\Delta t}]^n$$

where $n\Delta t = t$. This technique calculates the matrix exponential for a smaller time of interest (i.e. the time step Δt) using a truncated power series. These values are then used to determine the matrix exponential for the final time of interest, t.

The truncated power series method with scaling and propagating has two distinct sources of integration error. The first source of error is still due to the fact that an infinite power series has been truncated. This is true regardless of the time period for which the matrix exponential has been calculated. The second source of error is due to the scaling and propagating. The errors produced by the truncated power series are compounded when the matrix exponential for the time step is propagated by either stepping or squaring to produce the matrix exponential for the final time period of interest.

We can see that the two sources of integration error are dependent on two of the parameters of the integration scheme. The number of terms in the truncated power series and the time step determine the amount of integration error in the scaled matrix exponential. The number of steps, which is directly attributable to the time step, determines the magnification factor of the scaled matrix error. The goal, then, is to choose the two

parameters (the number of truncated power series terms and the time step) such that an acceptable accuracy is achieved in the least amount of computer work.

5.2 Propagation of Integration Error

The previous section explained the causes of integration error. We will now look closer at the actual propagation of the integration error. The notation of the base matrix used in chapter 3 is repeated here for convenience.

$$M_k = I + A\Delta t + \frac{(A\Delta t)^2}{2!} + \ldots + \frac{(A\Delta t)^k}{k!}$$

Additionally, we repeat the constraint that the norm of the matrix $A\Delta t$ be less than unity to insure quick convergence.

$$||A\Delta t|| < 1$$

It is desired now to bound the error due to the loss of terms in the truncated power series. Since the norm of the matrix is less than unity, we know that

$$\left|\left|\frac{(A\Delta t)^k}{k!}\right|\right| > \left|\left|\frac{(A\Delta t)^{k+1}}{(k+1)!}\right|\right|$$

The term on the right hand side of the equation would be the next term in the power series if it were not truncated at k . The right hand side of the equation is thus larger than any single term in the truncated portion of the power series.

Additionally, it can be shown that the terms in the transition matrix alternate sign for each failure level at each additional multiplication of the matrix. For example, suppose A had the following form. The (+) represents a positive quantity, while the (-) represents a negative value.

$$A\Delta t = \begin{bmatrix} - & 0 & 0 & 0 \\ + & - & 0 & 0 \\ 0 & + & - & 0 \\ 0 & 0 & + & 0 \end{bmatrix}$$

This would be the type of transition matrix for a four-state, non-cyclical Markov model. If this transition matrix were now raised to the second and third powers as in the Taylor power series, the matrices would have the following forms.

$$(A\Delta t)^2 = \begin{bmatrix} + & 0 & 0 & 0 \\ - & + & 0 & 0 \\ + & - & + & 0 \\ 0 & + & - & 0 \end{bmatrix} \qquad (A\Delta t)^3 = \begin{bmatrix} - & 0 & 0 & 0 \\ + & - & 0 & 0 \\ - & + & - & 0 \\ + & - & + & 0 \end{bmatrix}$$

Notice that the elements change sign for each higher power of the transition matrix. It can be seen that this yields a power series that converges by Leibnitz's Theorem for alternating power series. Therefore, we can bound the integration error using the next term in the power series. It can be seen that the terms in the power series after the truncation point alternate in sign and decrease in magnitude. Thus, the magnitude of the integration error is always less than the magnitude of the next term in the truncated power series. Finally, we have our first bound for the absolute integration error of truncated power series.

$$\text{AIE (base matrix using Taylor)} \leq \left|\left| \frac{(A\Delta t)^{k+1}}{(k+1)!} \right|\right|$$

Now that we have the integration error for the truncated power series bounded, we need to bound the error for entire interval of interest. Nominally, the integration scheme appears, with $n\Delta t = t$

$$e^{At} = [e^{A\Delta t}]^n \approx (M_k)^n$$

Rewriting this equation to include the additive error E yields

$$e^{At} = (M_k + E)^n$$

Using the binomial theorem and expanding the right hand side, one gets

$$e^{At} = (M_k)^n + n\,(M_k)^{n-1}\,E + \frac{n\,(n-1)}{2}\,(M_k)^{n-2}\,E^2 + \ldots$$

The error matrix E is composed only of powers of the transition matrix A. Thus, the error matrix has the same property as the transition matrix, namely that the terms decrease in magnitude and alternate in sign with each increase in the power of the matrix. We can then bound the integration error using only the second term in the series given above.

$$\text{AIE (Taylor)} \leq ||n\,M_k^{n-1}\,E||$$

We can further expand this equation through the M_k term

$$\text{AIE (Taylor)} \leq \left|\left| n \left[I + A\Delta t + \ldots + \frac{(A\Delta t)^k}{k!} \right]^{n-1} E \right|\right|$$

Since the $A\Delta t$ terms decrease in magnitude and change sign with every increase in power, we can see that the integration error can now be bounded by

$$\text{AIE (Taylor)} \leq ||n\, I\, E||$$

Finally, this can be rewritten knowing that the error matrix E is bounded by the integration error for the truncated power series given above. Therefore, the total integration error for the matrix exponential for the entire period of interest can be bounded by

$$\text{AIE (Taylor)} \leq \left|\left| \frac{n\,(A\Delta t)^{k+1}}{(k+1)!} \right|\right|$$

This last equation bounds the integration error for the truncated Taylor series algorithm. We can easily see that the integration error, of course, depends on the transition rates of the Markov model to be evaluated, A. The error also depends on the number of terms used in the truncated power series, k. It is clear that the greater number of terms in the truncated power series, the lower the integration error. This is evidenced both by the factorial term in the denominator and by the transition matrix raised to a power in the numerator. Remember that the elements in the transition matrix are less than unity, so they decrease in magnitude when raised to a power. The third parameter that the integration error depends upon is the time step. This is visible in both the n and the Δt. With k at least equal to one, a decrease in the time step results in a decrease in the integration error because the power of Δt is always equal to the power of the number of steps, n.

Summarizing then, the integration error is decreased with either a decrease in the time step or an increase in the number of terms included in the truncated power series. It is important to note that, as usual, either of these options to decrease the integration error results in an increase in the amount of computer work necessary to solve the system. The topic of computer work will be covered more completely in Chapter 7.

Unfortunately, the given equation bounding the integration error is not easily calculated. We can, however, use some properties of the matrix norm to massage the equation into a simpler form. The first property, the scalar multiplicative property, allows us to rewrite the equation with the scalar constants outside of the matrix norm.

$$\text{Scalar Multiplicative Propery (n scalar):} \quad ||n\, A|| \leq |n|\, ||A||$$

$$\text{AIE (Taylor)} \leq \frac{n}{(k+1)!}\, ||(A\Delta t)^{k+1}||$$

Another property of the matrix norm is that the norm of a product of matrices is less than or equal to the product of the norms of the individual matrices. This property is commonly referred to as the matrix multiplicative property,

$$\|A\,B\| \leq \|A\|\,\|B\|$$

This property allows us to write the equation as the power of the norm of the transition matrix.

$$\text{AIE (Taylor)} \leq \frac{n}{(k+1)!}\,||(A\Delta t)||^{k+1}$$

The error bound is much easier to calculate in this fashion. We only need to calculate the transition matrix once rather than multiplying the matrix k+1 times. The equation can be further simplified by again applying the scalar multiplicative property. Thus, the time step can be brought outside the matrix norm.

$$\text{AIE (Taylor)} \leq \frac{n\,(\Delta t)^{k+1}}{(k+1)!}\,||A||^{k+1}$$

Once outside the matrix norm, one of the time steps can be combined with the number of time steps to give the error bound equation in its simplest form.

$$\text{AIE (Taylor)} \leq \frac{t\,(\Delta t)^{k}}{(k+1)!}\,||A||^{k+1}$$

Given the system parameters of the transition matrix and the final time, the integration error is now explicitly dependent upon the time step and the number of terms in the truncated power series. This final version is straight forward and easy to calculate.

One note of caution is in order. While each simplification reduces the computational effort, the absolute integration bound becomes increasingly more conservative. Depending on the values of the transition matrix and the number of terms in the truncated series, the the triangle inequality simplification of the integration error bound may be very conservative. In cases where a highly accurate bound on the integration error is desired, it may be better to calculate the bound with the power of the matrix included inside the matrix norm. Another alternative would be to determine the integration parameters using a simpler bounding equation and then verifying the calculation with a more accurate bound. More on the proper selection of the integration parameters will be given in Chapter 8.

5.3 Richardson Extrapolation

Ideally, the integration error would be zero. We can see from the equation above that the integration error vanishes as the time step approaches zero. In this manner, we are calculating the limiting values of the state probabilities at the given time of interest. However, the amount of work increases sharply as the time step decreases. It was also shown that the roundoff error increases as the time step decreases.

A possible procedure to better determine the limiting values was proposed by Richardson and is called 'the deferred approach to the limit' or Richardson extrapolation [Butcher, 1987]. The method consists of evaluating the model once with a full time step, and then evaluating the model a second time with two one-half time steps. By comparing the two results, one can obtain a more accurate estimate of the limiting values.

The algorithm for Richardson extrapolation will be developed by example. We will start with the simplest Taylor series approximation for the matrix exponential, using base matrix M_1.

$$e^{At} \approx [I + A\Delta t]^n$$

The right hand side of this equation can be expanded using the binomial theorem.

$$e^{At} \approx I + n(A\Delta t) + \frac{n(n-1)}{2}(A\Delta t)^2 + \frac{n(n-1)(n-2)}{6}(A\Delta t)^3 + \ldots + \binom{n}{k}(A\Delta t)^k$$

where $\binom{n}{k}$ is the binomial coefficient. Using the relationship $n\Delta t = t$, the first three powers of the expansion can be rewritten as

$$e^{At} \approx I + At + \frac{1}{2}(At)^2 - \frac{n}{2}(A\Delta t)^2 + \frac{1}{6}(A\Delta t)^3 - \frac{(3n^2 - 2n)}{6}(A\Delta t)^3 + \ldots$$

If we now subtract the actual Taylor power series for the matrix exponential, the integration error is

$$\text{AIE (Richardson, Pass 1)} = \frac{n}{2}(A\Delta t)^2 + \frac{n^2}{2}(A\Delta t)^3 - \frac{n}{3}(A\Delta t)^3 + \ldots$$

The basis of the extrapolation method is to integrate the system a second time with half the time step. A time step of one-half requires twice the number of integration steps. This translates to n more steps or one more squaring. Thus, we substitute $\frac{\Delta t}{2}$ for Δt and 2n

for n into the above equation, still maintaining the relationship $2n\frac{\Delta t}{2} = t$. The integration error for the second integration pass is now

$$\text{AIE (Richardson, Pass 2)} = \frac{2n}{2}\left(\frac{A\Delta t}{2}\right)^2 + \frac{3(2n)^2 - 2(2n)}{6}\left(\frac{A\Delta t}{2}\right)^3 + \cdots$$

This can be rewritten into a better form for comparison purposes

$$\text{AIE (Richardson, Pass 2)} = \frac{1}{2}\left[\frac{n}{2}(A\Delta t)^2\right] + \frac{1}{2}\left[\frac{n^2}{2}(A\Delta t)^3\right] - \frac{1}{4}\left[\frac{n}{3}(A\Delta t)^3\right] + \ldots$$

If we now compare the integration error for the two passes, we can see that the second pass (the one with the half time step) is more accurate than the first pass. The second pass can be used as a bound on the accuracy of the first pass. This is a quick and simple means to bound the integration error. However, a closer look at the integration error equations for the two passes shows a unique relationship that can be exploited.

Close examination of the two integration equations shows that the second pass has approximately one-half the integration error of the first pass, assuming the norm of $A\Delta t$ is less than unity. This fact can be used to increase the order of the approximation. For convenience, we introduce a new notation: P' will denote the values obtained after the first pass with the full time step; P" will denote the values obtained after the second pass with the half time step; P'" will denote the values obtained after the third pass with a quarter time step; and so on. In the same fashion, P° will denote the values obtained after the first extrapolation, $P^{\circ\circ}$ the values after the second extrapolation, etc. P* will denote the exact value of the matrix exponential. Lastly, IE(n) will represent the integration error for the n^{th} integration pass using a time step of $\frac{\Delta t}{2^{n-1}}$. With this notation, we have the following two equations for the integration error for the two integration passes.

$$P^* - P' = IE(1)$$

$$P^* - P'' = IE(2) \approx \frac{IE(1)}{2}$$

Subtracting the second equation from the first yields

$$P'' - P' = IE(1) - IE(2) \approx \frac{1}{2} IE(1)$$

We know, however, that P* = IE(1) + P'. Substituting to eliminate IE(1) now gives

$$P^* \approx P^\circ(2,1) = 2[P'' - P'] + P' \approx 2P'' - P'$$

Thus, by integrating the problem twice, once with half the time step, we can use the values of the two integrations to project or extrapolate to the limiting values. Overall, taking twice the second integration pass and subtracting the first integration pass yields an estimate that is closer in value to the desired matrix exponential than either of the original integration passes.

This new approximation can be shown to have an integration error that is of a higher order than either P'' or P'. The power series for IE(1) and IE(2) were given above. It can be clearly seen, that the extrapolation method would yield the following equation for integration error, where IE°(2,1) represents the integration error after the extrapolation.

$$IE^\circ(2,1) = 2IE(2) - IE(1)$$

In terms of the power series for the integration errors, this gives

$$IE^\circ(2,1) = -\frac{1}{2}\left[\frac{n}{3}(A\Delta t)^3\right] + \ldots$$

Since the order of the error is defined as the lowest power of the matrix $A\Delta t$ in the power series, the order of the extrapolation is two. The order of both of the original integration passes was one. This is equivalent to increasing the order of the approximation by one.

This entire process can be continued further. Suppose a third integration pass was conducted with a time step half that of the second pass, or equivalently, one-fourth the time step of the first pass. Using this time step, the integration error would be

$$IE\ (Pass\ 3) = \frac{4n}{2}\left(\frac{A\Delta t}{4}\right)^2 + \frac{3(4n)^2 - 2(4n)}{6}\left(\frac{A\Delta t}{4}\right)^3 + \ldots$$

Again, rewriting this equation into a more convenient form for comparison gives

$$IE\ (Pass\ 3) = \frac{1}{4}\left[\frac{n}{2}(A\Delta t)^2\right] + \frac{1}{4}\left[\frac{n^2}{2}(A\Delta t)^3\right] - \frac{1}{16}\left[\frac{n}{3}(A\Delta t)^3\right]\ldots$$

In a analogous manner to pass 1 and pass 2, pass 3 has an integration error that is approximately one-half the integration error of the second pass. Thus, we can now do an extrapolation with integration passes two and three. This extrapolation would yield a set of equations similar to the previous extrapolation.

$$P^* - P'' = IE(2)$$

$$P^* - P''' = IE(3) \approx \frac{IE(2)}{2}$$

Subtracting the second equation from the first yields

$$P''' - P'' = IE(2) - IE(3) \approx \frac{1}{2} IE(2)$$

We know, however, that $P^* = IE(2) + P''$. Substituting to eliminate IE(2) now gives

$$P^* \approx P^\circ(3,2) = 2[P''' - P''] + P'' = 2P''' - P''$$

Again, the new approximation has less integration error than either of the original integration passes.

This approximation also produces an integration error that is higher in order than either of the integration passes. The power series for the integration error is again given by

$$IE^\circ(3,2) = 2IE^\circ(3) - IE^\circ(2)$$

$$IE^\circ(3,2) = \frac{1}{8}\left[\frac{n}{3}(A\Delta t)^3\right] \ldots$$

Clearly, the order of the error is two while the order of the individual passes was one, giving an increase of one. The integration error for the extrapolation using P'' and P''' is, of course, less than the integration error for the extrapolation using P' and P'' because a smaller time step is used.

We are now prepared to continue the extrapolation method to the next level. It is obvious that the integration error of the second extrapolation is approximately one fourth the integration error of the first extrapolation. We can use this knowledge to conduct another extrapolation to further improve the accuracy of our estimation of the matrix exponential.

The first two extrapolations can be represented by the following equations.

$$P^* \approx P^\circ(2,1) = 2P'' - P'$$

$$P^* \approx P^\circ(3,2) = 2P''' - P''$$

The two new approximations to the matrix exponential can now be used for a second level extrapolation. Since the difference in the integration errors is a factor of four, the extrapolation equation is

$$P^* \approx P^{\circ\circ}(3,1) = \frac{4P^{\circ}(3,2) - P^{\circ}(2,1)}{3}$$

The division by three is required since the first probability is being multiplied by four. This equation yields an integration error that is third order in $A\Delta t$.

$$IE^{\circ\circ}(3,1) = \frac{4\,IE^{\circ}(3,2) - IE^{\circ}(2,1)}{3}$$

$$IE^{\circ\circ}(3,1) = 4 \cdot \frac{1}{8}\left[\frac{n}{3}(A\Delta t)^3\right] - \frac{1}{2}\left[\frac{n}{3}(A\Delta t)^3\right] + \ldots = \emptyset[(A\Delta t)^3]$$

Thus, with two levels of extrapolations we have increased the order of the error by two. In this manner, then, we can successively integrate and extrapolate to decrease the order of the integration error.

Some refinements are necessary to make this approach generally applicable. The extrapolation methods above were conducted using a first order base matrix to approximate the matrix exponential. Often times, a higher order base matrix is used. In these cases, the ratios of the integration errors vary according to the order of the base matrix. A base matrix of second order has an integration error of second order. Obviously then, the factor of two in the extrapolation method to rid the approximation of first order errors is not suitable. More insight to the extrapolation method is necessary to make it generally acceptable.

The multiplication factors for the extrapolation method arise out of the difference between the power of n and the power of Δt in the integration error. For the first extrapolation, the power of n in the largest error term is equal the power of Δt. Thus, the extrapolation parameter for a general integration method is

$$EP = 2^k$$

where k is the highest power of the base matrix.

The second-level extrapolation parameters also result from the difference between the power of n and the power of Δt. However, the first extrapolation increases the power of Δt in the error equation. Thus, the difference between the power of n and the power of

Δt is increased by one. We define a new parameter j to be the number of extrapolation levels completed, where j = 0, 1, 2, ... The extrapolation parameters, EP(j), for a general integration and general extrapolation method are now

$$EP(j) = 2^{k+j} \qquad j = 0, 1, 2, \ldots$$

It is important to remember that the extrapolation parameters are the parameters that are multiplied by the integration pass of the smaller time step. Also, it is important to remember that the first-level extrapolation method as described here can be used only with integration passes whose time steps differ by a factor of two. Each successive extrapolation should be done with adjoining approximations of the matrix exponential.

Additionally, the value of the extrapolation must be divided by an appropriate factor since the probabilities are being magnified in order to do the extrapolation. The method calls for the integration pass with the smaller time step to be multiplied by the extrapolation parameter given above. The other integration pass is then subtracted from this value. Thus, a corrective dividing factor of one less than the extrapolation factor is necessary to produce a value that is again representative of a single state probability.

Finally, we can write a general equation for a probability state vector to describe the extrapolation process.

$$P_{j+1} = \frac{(EP_j \bullet P'_j) - P_j}{EP_j - 1}$$

with it understood that EP_j is the extrapolation parameter for the Taylor series. The prime (') signifies the value with the smaller time step. The probability state vector can be determined using either a squaring or a stepping algorithm.

A general equation that bounds the integration error can be written to complement the probability equation. Including the additive absolute integration error to the probability state vector yields

$$P_{j+1} + AIE_{j+1} = \frac{[EP_j \bullet (P'_j + AIE'_j)] - (P_j + AIE_j)}{EP_j - 1}$$

If the state probabilities are now subtracted out, we have a equation that bounds the absolute integration error for the extrapolation process.

$$AIE_{j+1} \text{ (Richardson with Taylor)} \leq \frac{(EP_j \bullet AIE'_j) - AIE_j}{EP_j - 1}$$

where AE_j and AE_j' are determined by any of the methods described previously. Of course the tightness of this bound depends heavily on the tightness of the bounds for the error of the initial integration passes.

This process can be continued to be used successively as many times as is desired. Each time the full process is used, the order of the integration error will decrease by one. However, one of the goals of this study is to determine an accurate method of integration with relatively small amounts of computer work. The question to be answered, then, is in what situations does the extrapolation method decrease the integration error with minimal increases in computer work, and in what situations is another method with different integration parameters more appropriate.

5.4 Error of the Taylor Series

The integration error of the truncated Taylor series approximation to the matrix exponential is dependent upon the system parameters and the integration parameters. Various bounds for the integration error of a Taylor series method were given above in Section 5.2. The purpose of this section is to present numerical results of the integration error from various Markov models. The dependence on the system parameters, the number of terms in the truncation, and the size of the time step will be shown. Additionally, the tightness of the various bounds for the integration error will be established.

It is difficult to separate the effects of roundoff error and the effects of integration error when running an actual problem. One can approximately separate their individual contributions by minimizing the effects of the one not being studied. A convenient way to minimize the effects of the roundoff error is to keep the number of mathematical operations inconsequential. In this way, we assume that the roundoff error is negligible, and all the error is attributed to integration error. Thus, the integration error can be determined by subtracting the approximate answer from the exact answer. Unfortunately, the exact answer is not easily derived for many of the Markov models studied. In these cases, the problem can be run in a higher precision (i.e. double precision) and then used for comparison. Caution should be taken to insure that the errors for the double precision run are outside the range of the approximation.

The solutions given here will use a scaling technique with variable renormalization in an effort to minimize the roundoff error effects while still demonstrating the integration error patterns. They were run in a double precision of greater than fifteen significant digits.

We will again use the four-state Markov model of previous chapters to describe the integration error patterns. Repeating from Section 2.7, the closed form solution to the system probabilities for this model are

$$
\begin{aligned}
P_1(t) &= e^{-(a+b)t} \\
P_2(t) &= e^{-bt} - e^{-(a+b)t} \\
P_3(t) &= e^{-at} - e^{-(a+b)t} \\
P_4(t) &= 1-e^{-bt} - \frac{b}{a+b}(1-e^{-(a+b)t}) \\
&\quad + 1-e^{-at} - \frac{a}{a+b}(1-e^{-(a+b)t})
\end{aligned}
$$

For this example, the following system parameters were used

$$
\begin{aligned}
a &= 10^{-3}\ /\text{hour} \\
b &= 10^{-4}\ /\text{hour} \\
t_1 &= 100 \text{ hours} \\
t_2 &= 2\text{x}10^4 \text{ hours}
\end{aligned}
$$

Reflecting on the discussion of the characteristic time of the system, it was decided to test the integration error for two final times. The first final time results in a characteristic time less than unity, while the second illustration gives a characteristic time greater than unity. These two final times yield the following state probability vectors

$$
P\ (100 \text{ hours}) = \begin{bmatrix} 0.8958341 \\ 0.0942157 \\ 0.0090033 \\ 0.0009469 \end{bmatrix} \qquad P\ (2\text{x}10^4 \text{ hours}) = \begin{bmatrix} 2.789468\text{x}10^{-10} \\ 0.1353353 \\ 1.782206\text{x}10^{-09} \\ 0.8646647 \end{bmatrix}
$$

Plotted versus the time step below is the most liberal bound for the integration error, using the maximum value for the matrix norm and raising this maximum to the appropriate power for the various base matrices. On these same graphs are plotted the absolute integration error for the four states for comparison purposes. There is a graph for each of the first three base matrices, M_1, M_2, and M_3 at both of the final times.

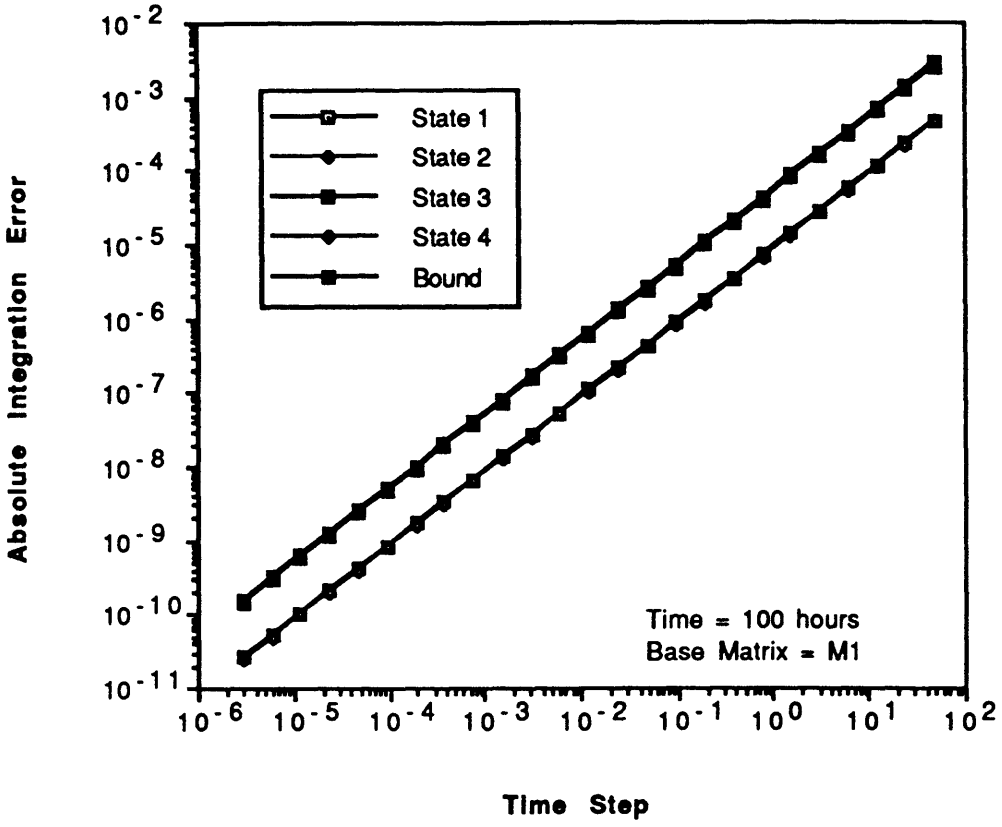


Figure 5-1: Absolute Integration Error using M_1 versus Time Step prior to the Characteristic Time

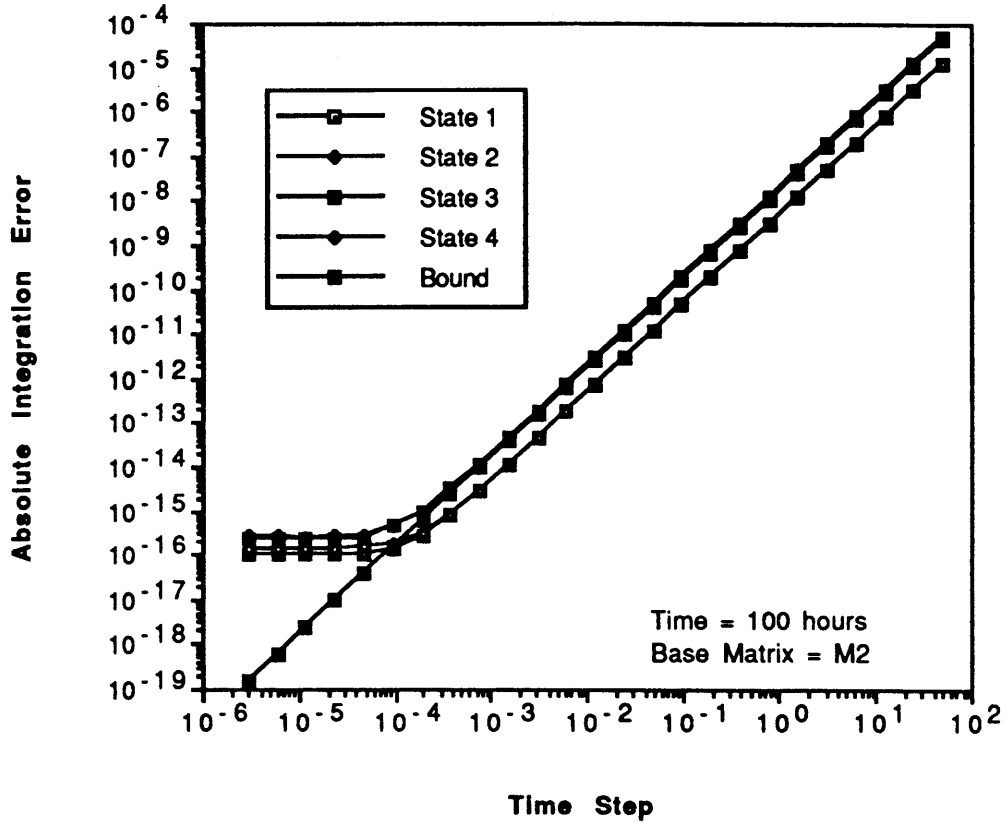


Figure 5-2: Absolute Integration Error using M_2 versus Time Step prior to the Characteristic Time

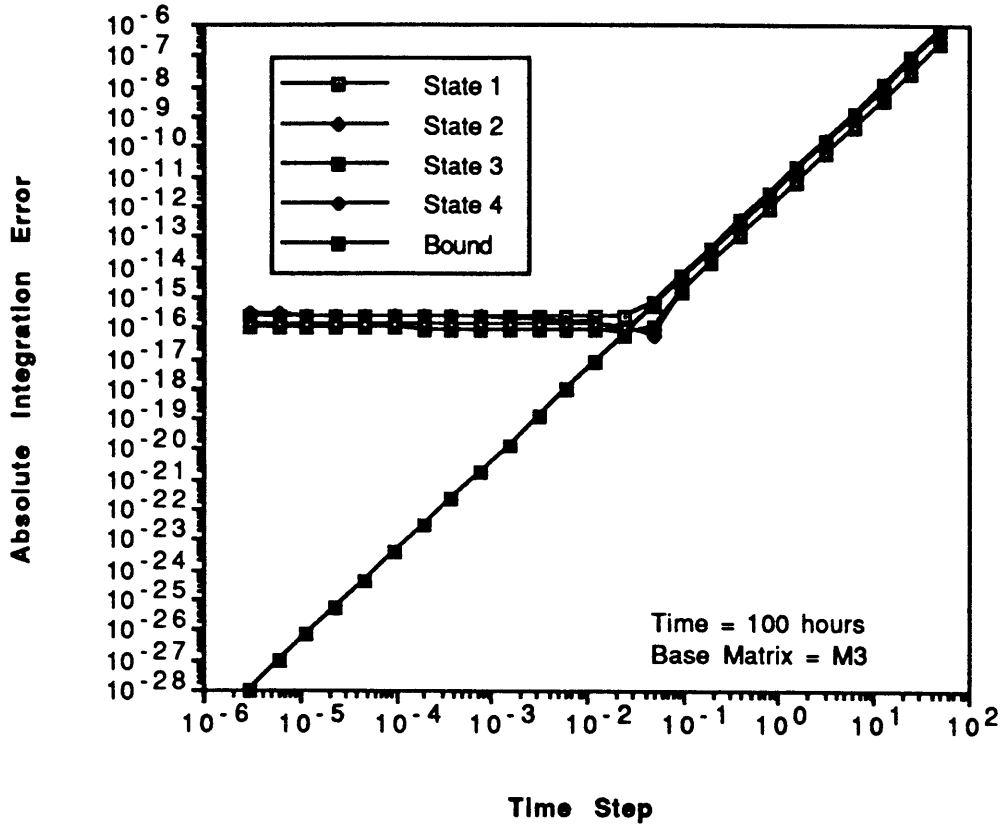


Figure 5-3: Absolute Integration Error using M_3 versus Time Step prior to the Characteristic Time

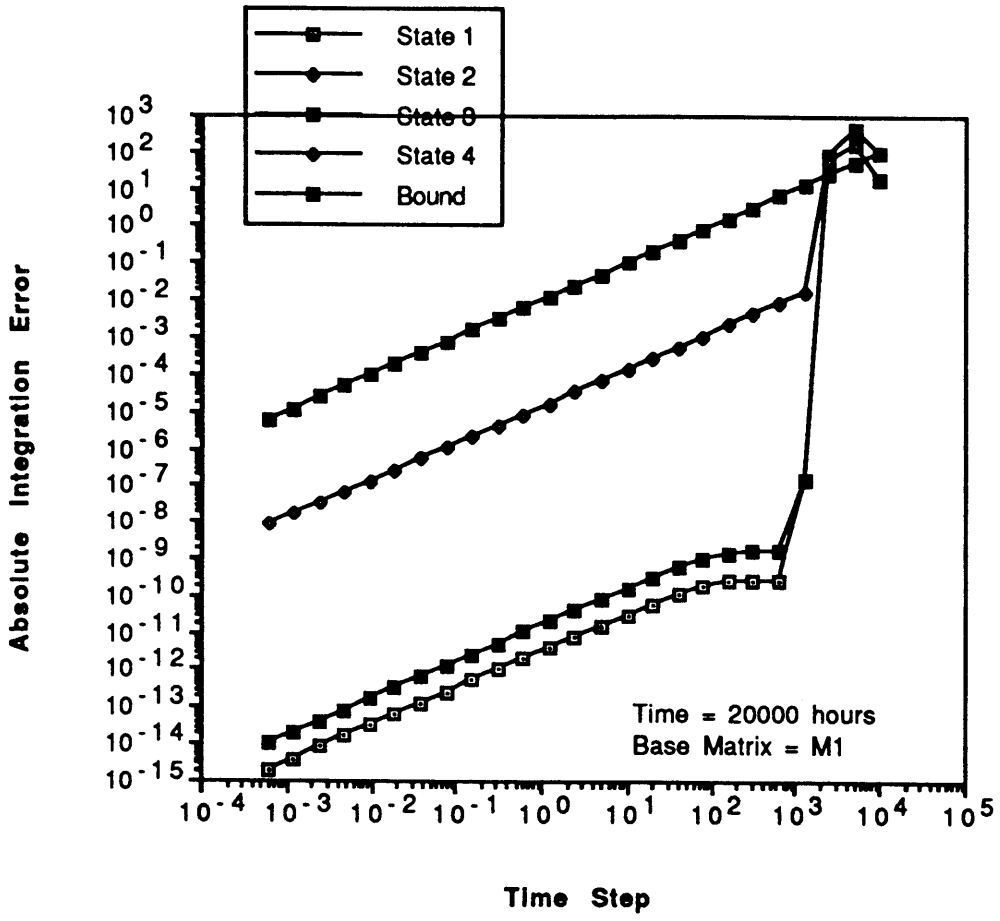


Figure 5-4: Absolute Integration Error using M_1 versus Time Step after the Characteristic Time

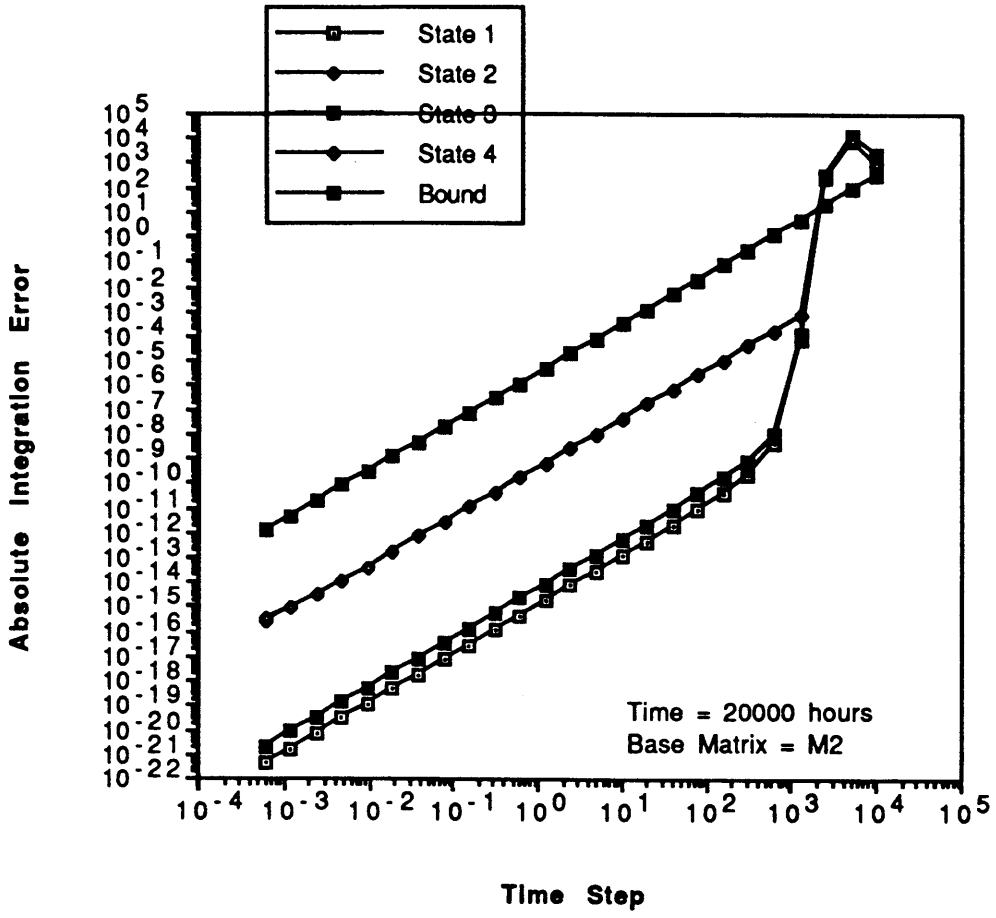


Figure 5-5: Absolute Integration Error using M_2 versus Time Step after the Characteristic Time

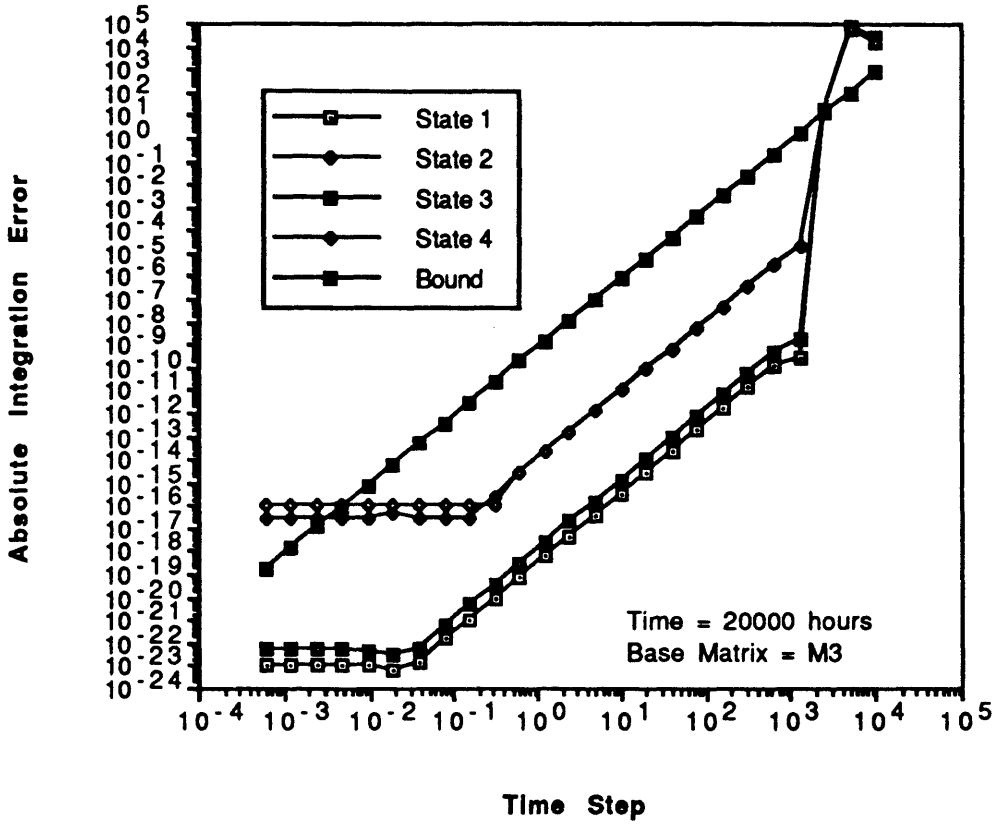


Figure 5-6: Absolute Integration Error using M_3 versus Time Step after the Characteristic Time

The first three graphs (Figures 5-1 to 5-3) show the error propagation patterns for the final time less than the characteristic time. One can see that the error steadily decreases as the time step decreases. Also, examining the errors across the three graphs shows a significant decrease in the errors as the number of terms in the base matrix is increased. For example, if it was desired to have a absolute integration error of 10^{-6}, we would need a time step of $3x10^{-2}$ hours for M_1, or a time step of 6 hours for M_2, or a time step of only 60 hours for M_3. Figures 5-2 and 5-3 appear flat toward the lower values of the time step because the limits of machine precision.

The second three graphs (Figures 5-4 to 5-6) present the absolute integration error for final time greater than the characteristic time. Once more, these show a steady decrease as the time step decreases. Significant changes in the integration error are again evident when looking across the three graphs. An absolute integration error of 10^{-6} would require

a time step of $7x10^{-2}$ hours for M_1; a time step of $6x10^{-1}$ hours for M_2; and a time step of 10 hours for M_3. The unusual error patterns in the region of large time step is due to the norm of the matrix $A\Delta t$ being greater than unity. The flat regions toward the lower end of the scale is again due to the limits of machine precision.

The error bound equation is also plotted on all six graphs. In the first three, the error bound is difficult to discern because of its close proximity to States 1 and 2. In the latter three graphs, the error bound is easily recognized. Clearly, in all cases the error bound does indeed provided an upper limit to the integration errors experienced. However, because the most liberal bound was used, it is not very tight for the integrations run past the characteristic time. This is a common occurrence. If a tighter bound is desired for systems run past the characteristic time, one of the more conservative error bound equations will have to be implemented.

It is important to point out that the roundoff error reduction technique of variable renormalization had little influence on the resulting integration error but was used to reduce the roundoff error. The renormalization process affects the actual numerical values, but it does not affect the integration technique.

5.5 Error of the Taylor Series with Richardson Extrapolation

It was shown above that Richardson extrapolation increased the order of the accuracy of the approximation by one for each level of extrapolation. Numerical results for the extrapolation methods are given below. The time step used for graphing these data is the time step of the first integration pass in the extrapolation process. These results were generated using the system of the previous section. Characteristic times both greater than and less than unity were used, however, only the first two base matrices were the premises of the integration schemes. To keep the graphs simple, only the integration errors for the first state will be shown. Output was generated for two levels of extrapolation to show its effective use when applied successively.

$$a = 10^{-3}\ /\text{hour}$$
$$b = 10^{-4}\ /\text{hour}$$
$$t_1 = 100\ \text{hours}$$
$$t_2 = 2x10^{4}\ \text{hours}$$

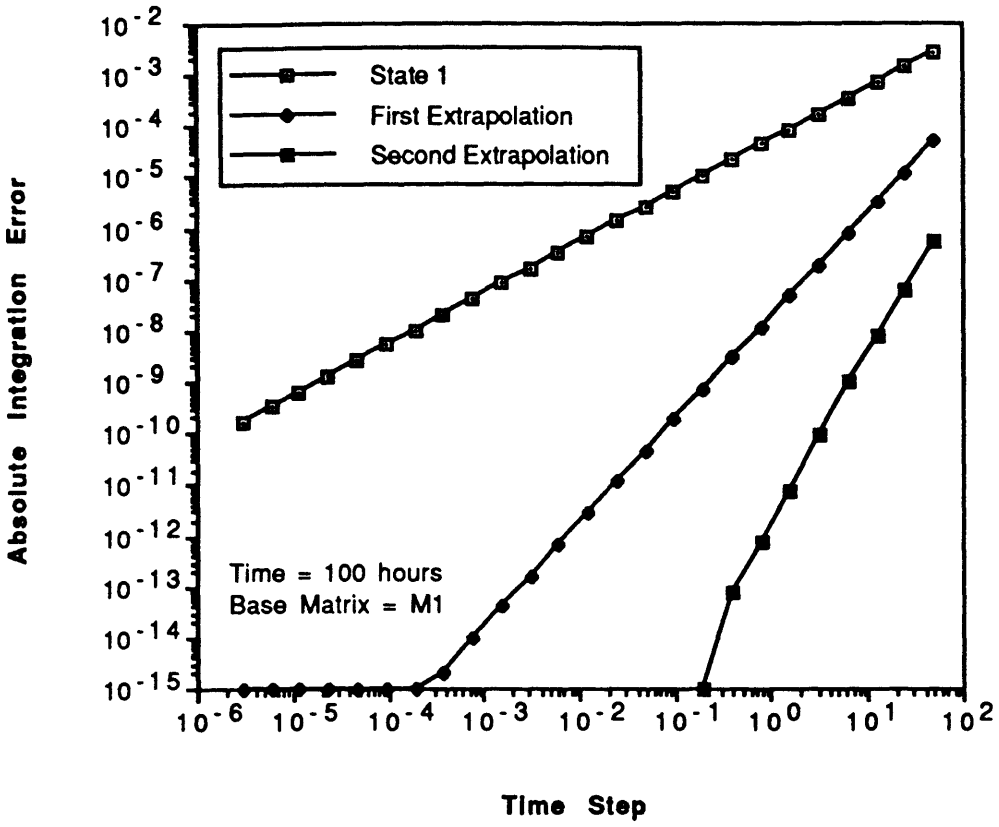


Figure 5-7: Absolute Integration Error using M_1 versus Time Step before the Characteristic Time with Richardson Extrapolation

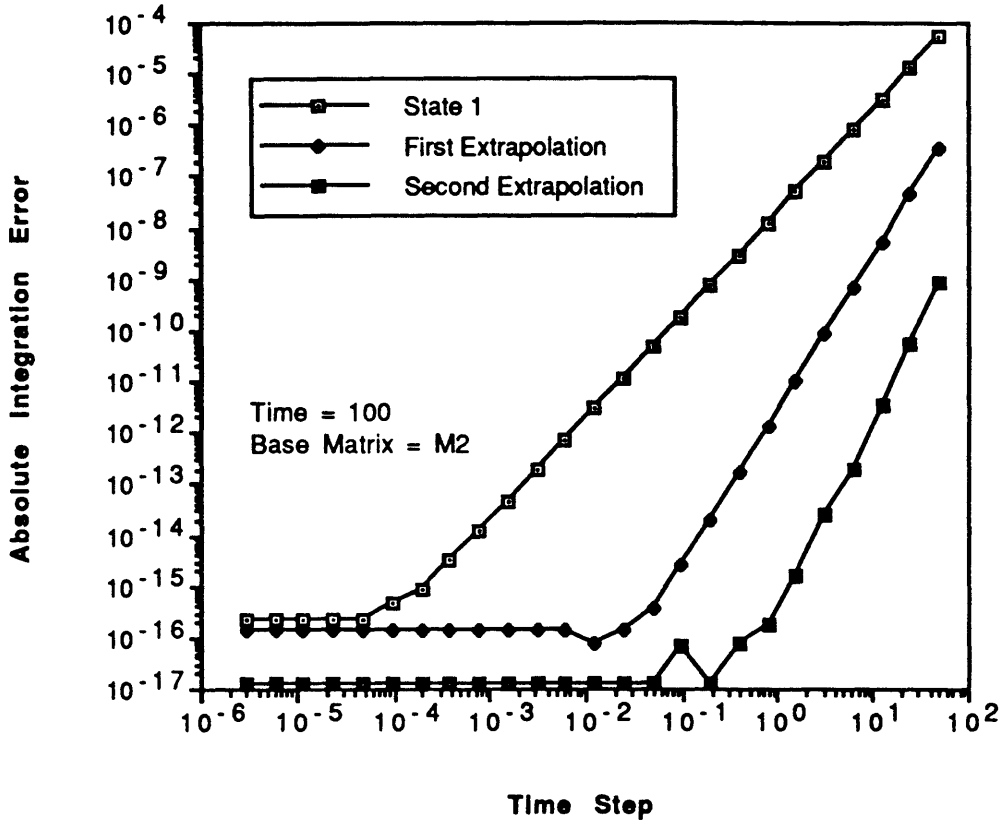


Figure 5-8: Absolute Integration Error using M_2 versus Time Step before the Characteristic Time with Richardson Extrapolation

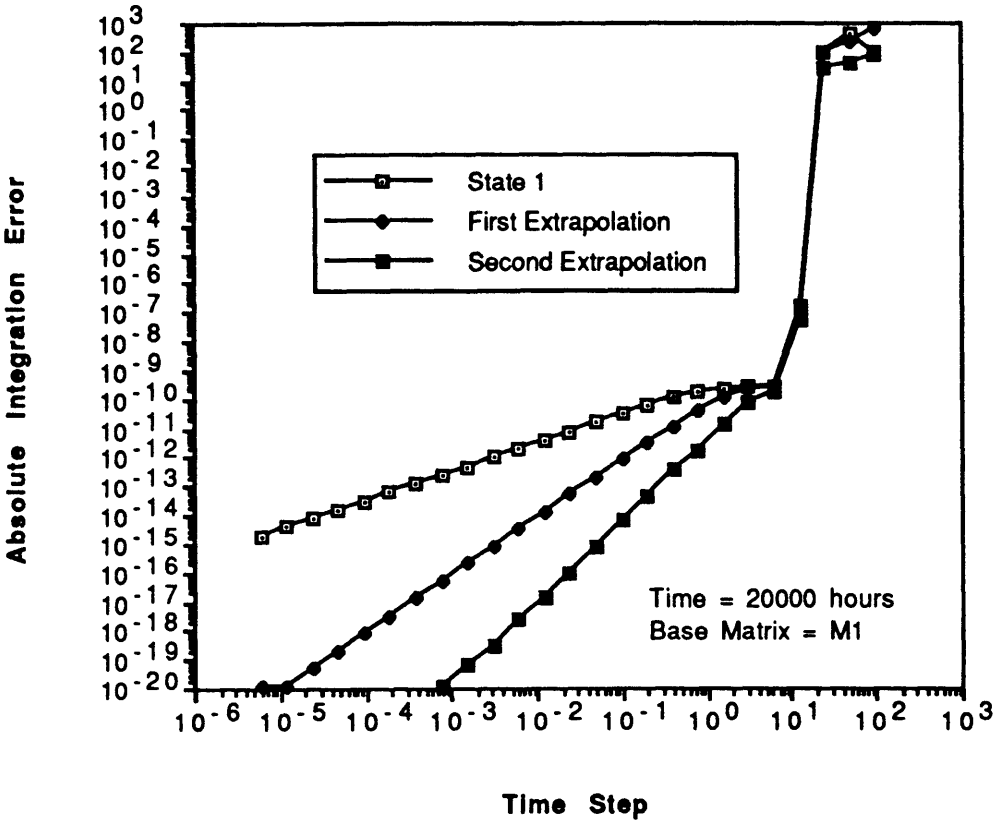


Figure 5-9: Absolute Integration Error using M_1 versus Time Step after the Characteristic Time with Richardson Extrapolation

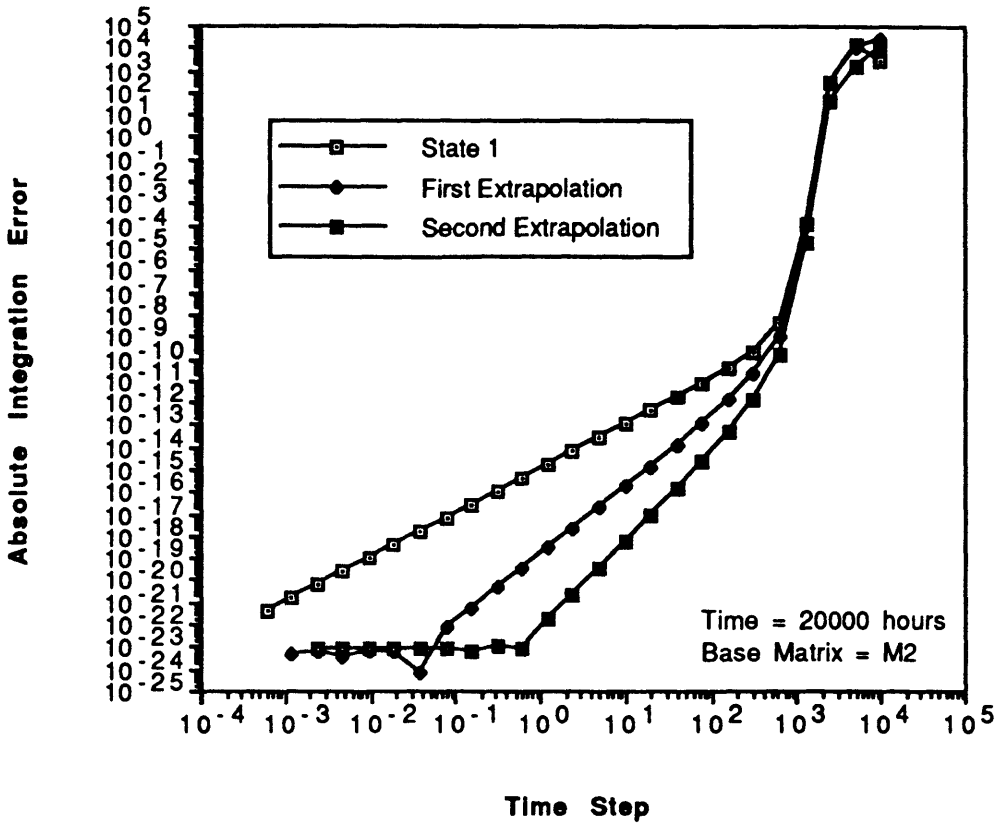


Figure 5-10: Absolute Integration Error using M_2 versus Time Step after the Characteristic Time with Richardson Extrapolation

The first two graphs (Figures 5-7 and 5-8) present the results of Richardson extrapolation before the characteristic time, while the latter two graphs (Figures 5-9 and 5-10) present the results of Richardson extrapolation after the characteristic time. In each case, the extrapolation methods greatly reduce the absolute integration error. However, the relative increase in accuracy is greater for the first extrapolation than for the second one, implying a law of diminishing returns. Note that in all four graphs, the different methods have different slopes. The different slopes attest to the different orders of approximation achieved. The unusual areas toward the larger time steps in Figures 5-9 and 5-10 is due to a transition matrix norm greater than unity.

Again it is important to note that the data for Richardson extrapolation is given versus the time step. This data is misleading if one is primarily concerned about the amount of computer work necessary to compute the values. In most cases, the

extrapolation method requires approximately twice as much work to achieve the additional accuracy. The question of whether this is an efficient method is left open.

5.6 Chain Model Approximations

The majority of this chapter has been spent developing equations that bound the absolute integration error. This analysis has been based on the norm of the transition matrix. One of the drawbacks of this approach is the loss of individuality. While all of the states may be bounded, the bounds may be very conservative for some states while appropriate for others. This is evidenced in some of the graphs shown above.

An alternative approach is to return to the chain model approximations described in Section 2.6. The chain model has the advantage that the state probabilities are easily determined given the system parameters. However, as was stated before, the chain model introduces more approximation errors. Thus, the equations developed here are only approximations and are not guaranteed to bound the integration errors.

The bounds for the integration errors were derived using the norm of the transition matrix. To approximate the errors for various states, the matrix norm must be broken up into the individual components. For the chain model, this is relatively easy.

The integration error for a general Markov model was shown in section 5.2 to be

$$\text{AIE (Taylor)} \leq \left\| \frac{n\,(A\Delta t)^{k+1}}{(k+1)!} \right\|$$

Clearly, we can pull the n and the (k+1)! terms outside of the matrix norm without affecting it, since both values are positive. Thus, we must determine the individual integration errors from the norm of the transition matrix.

Given below are the first three powers of the transition matrix for a five-state non-cyclical chain model.

$$A\Delta t = \begin{bmatrix} -a\Delta t & 0 & 0 & 0 & 0 \\ +a\Delta t & -a\Delta t & 0 & 0 & 0 \\ 0 & +a\Delta t & -a\Delta t & 0 & 0 \\ 0 & 0 & +a\Delta t & -a\Delta t & 0 \\ 0 & 0 & 0 & +a\Delta t & 0 \end{bmatrix}$$

$$(A\Delta t)^2 = \begin{bmatrix} (a\Delta t)^2 & 0 & 0 & 0 & 0 \\ -2(a\Delta t)^2 & (a\Delta t)^2 & 0 & 0 & 0 \\ (a\Delta t)^2 & -2(a\Delta t)^2 & (a\Delta t)^2 & 0 & 0 \\ 0 & (a\Delta t)^2 & -2(a\Delta t)^2 & (a\Delta t)^2 & 0 \\ 0 & 0 & (a\Delta t)^2 & -(a\Delta t)^2 & 0 \end{bmatrix}$$

$$(A\Delta t)^3 = \begin{bmatrix} -(a\Delta t)^3 & 0 & 0 & 0 & 0 \\ 3(a\Delta t)^3 & -(a\Delta t)^3 & 0 & 0 & 0 \\ -3(a\Delta t)^3 & 3(a\Delta t)^3 & -(a\Delta t)^3 & 0 & 0 \\ (a\Delta t)^3 & -3(a\Delta t)^3 & 3(a\Delta t)^3 & -(a\Delta t)^3 & 0 \\ 0 & (a\Delta t)^3 & -2(a\Delta t)^3 & (a\Delta t)^3 & 0 \end{bmatrix}$$

A distinct pattern can be seen to be emerging in the coefficients of the terms in the first column. Examining the means by which these coefficients were calculated, one can see that they are given by the binomial coefficient

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

In terms of the parameters of the integration error equation, the coefficients are given by

$$\binom{k+1}{f} = \frac{(k+1)!}{(k+1-f)!\,f!}$$

where f is the failure level of the state of interest. This leads to an integration error equation that bounds the integration error of the individual states in a chain model.

$$AIE_f \text{ (Taylor chain)} \leq \frac{n\binom{k+1}{f}(\lambda\Delta t)^{k+1}}{(k+1)!}$$

where a is the transition rate used in the chain model. This equation, like many of the previous ones, can be rewritten so that it is only in terms of the time step.

$$AIE_f \text{ (Taylor chain)} \leq \frac{\binom{k+1}{f} t\,(\Delta t)^k \lambda^{k+1}}{(k+1)!}$$

While the above equation may bound the integration error of the chain model, the uncertainties in translating a general Markov model into a chain model casts doubt on this error bound. In some cases, the integration error may still be bound, but this method

cannot guarantee that the error is bounded. Generally, the chain model provides a good approximation to the dynamics of more complex Markov models. Nevertheless, caution must be exercised whenever using a method involving the chain model.

5.7 Relative Error Approximations for the Integration Error

It was stated previously that integration error is easier to describe in terms of absolute error, while the roundoff error is easier to describe in terms of relative error. In order to have a comprehensive numerical error bound, however, the integration error and the roundoff error should be presented in the same manner. It is usually more desirable to speak in terms of relative errors since an *a priori* knowledge of the magnitude of the state probabilities would be necessary to determine appropriate accuracies in terms of absolute errors.

Relative error is defined as the absolute error divided by the exact answer. We have already developed two types of equations for the absolute integration error. Thus, we need a value for the exact answer. Obviously, this is impractical. Therefore, we will use an approximation to the exact answer. The best means we have to approximate a state probability is by means of the chain model approximations.

The general equation for the relative integration error equation will be

$$\text{RIE} \leq \frac{\text{AIE}}{\text{factor}}$$

where AIE represents any of the absolute integration error approximations or bounds developed in this chapter, and where 'factor' is the approximation to the exact state probability. The term 'factor' will be determined using the chain model. Two chain models were given in Section 2.6 that provided lower and upper bounds on a state probability of interest. Since the 'factor' is in the denominator and since we want the relative error to be conservative, we should use the lower bound on the state probability. This was given in Section 2.6 for state three to be

$$\underline{\text{Lower Bound Probability for } S_3}$$

$$P_3(t) = \left(\frac{\lambda_{min}}{\lambda_{max} + \lambda_{min}}\right)\left[\left(\frac{\lambda_{min}}{\lambda_{max} + \lambda_{min}}\right)\left(e^{\lambda_{max}t} - e^{\lambda_{min}t}\right) + \lambda_{min} t\, e^{\lambda_{min}t}\right]$$

For a general state in a chain model, this is easier to represent as

$$\text{factor} = \binom{n^*}{f} (\lambda_{min}\,\Delta t^*)^f\,(1 - \lambda_{max}\,\Delta t^*)^{n^* - f}$$

where n^* is large enough and Δt^* is small enough such that 'factor' is convergent to an acceptable accuracy. The relation $n^*\Delta t^* = t$ is still valid, and n^* and Δt^* have no relation to n and Δt.

Finally, we can determine the approximate relative integration error using the chain model. This now leads to the following two equations; one using the matrix norm technique for the absolute errors, and the other using the chain model approximation for the absolute errors. Other methods of determining the absolute error are also appropriate.

$$\text{RIE (Taylor using matrix norm)} \approx \frac{t\,(\Delta t)^k\,\|A\|^{k+1}}{(k+1)!\binom{n^*}{f}(\lambda_{min}\Delta t^*)^f\,(1 - \lambda_{max}\,\Delta t^*)^{n^* - f}}$$

$$\text{RIE (Taylor using chain model)} \approx \frac{t\,(\Delta t)^k\,\binom{k+1}{f}\,a^{k+1}}{(k+1)!\binom{n^*}{f}(\lambda_{min}\Delta t^*)^f\,(1 - \lambda_{max}\,\Delta t^*)^{n^* - f}}$$

Once again, the relative integration error equations are only approximations because the chain model was used. They tend to be good approximations, but caution is still warranted.

5.8 Recapitulation

This chapter was dedicated to determining the integration error incurred when using the Taylor series method to solve the matrix exponential. The causes of integration error were discussed. It was shown that there are two sources of integration error. These are caused by the truncation of an infinite power series and by the compounding of the first error due to repeated multiplication of the base matrix. It was also shown that these errors are dependent upon the number of terms in the truncated power series and the time step, the two parameters of the integration scheme chosen.

After the sources of error were exposed, it was shown how the errors propagate. Equations that bound the integration error were developed. The various error bounds exhibited different levels of conservatism. Each level of increased accuracy required more computer work. The bounds are dependent on the system parameters of the transition matrix and the final time. The bounds are also dependent on the integration parameters of the number of terms in the truncated power series and the time step.

A method of reducing the integration error called 'the deferred approach to the limit' or Richardson extrapolation was also developed. The basis of this method is to integrate the system twice, once with a full time step and once with a half time step. The two approximations are then extrapolated in such a manner so as to determine a more accurate approximation than either of the individual values. The manner in which the initial approximations are extrapolated depends on the number of terms in the base matrix and the level of extrapolation. Guidelines for repeated extrapolation are developed, along with equations for the extrapolation parameters. The equations for the integration error of the extrapolated values are also given.

Once the integration error has been described for the various methods of solution, numerical results are given. These results, given in the form of graphs, show the benefit of the various methods for characteristic times both greater than and less that unity. The bounds of the integration error are also plotted for comparison purposes. Numerical results were also generated for Richardson extrapolation, with the error bounds again being plotted for comparison purposes.

Integration error equations were developed for the chain model approximations to allow for more flexibility and choices in estimating error. These equations were based on the general equation for the bound of integration error. However, the transition matrix was assumed to describe a chain model. This assumption greatly simplifies the equation, negating the need to calculate a matrix norm. The equations developed can be used to bound the error of the chain model, but caution must be used when this result is applied to the general system modeled by a chain model.

Finally, the last section of this chapter discussed a method of determining the integration error in terms of relative error. The chain model approximation was used to estimate the exact state probabilities. These estimates of the state probabilities were then used in conjunction with the methods of determining the absolute error to arrive at the final relative error equations.

This chapter examined the integration error associated with the Taylor series approximation approach to the solution of the matrix exponential. Equations that describe this error for various situations were developed. The other major solution technique discussed in this paper is the Padé series approximation approach. The following chapter will examine the the integration error associated with this method, and equations that

describe the error for various situations will be developed. Because of the similar nature of the integration error propagation, Chapter 6 will draw heavily on the work presented here.

Chapter 6

Integration Error--Padé Series Approximations

The previous chapter described the integration error associated with the Taylor series approximation as a solution technique to computing the matrix exponential. The focus of this chapter is to describe the integration error associated with the Padé series approximation as a solution technique to computing the matrix exponential. The causes and propagation of integration error as it applies to Padé series will be given. The parameters for Richardson extrapolation using Padé will be developed. Once the integration patterns are understood, a set of equations will be given that bound the integration error incurred using the Padé series approximations. Finally, numerical results will be given for the integration error of a sample Markov model solved using Padé approximations.

In describing propagation patterns associated with the Taylor series approach, Chapter 5 also described many aspects of integration error not unique to Taylor series but generally applicable to power series solutions in general. As such, this chapter will depend on a knowledge of Chapter 5 and borrow heavily from it. A thorough discussion of Richardson extrapolation will not be given, and most of the general features of integration error will not be discussed in detail. The chain model approximations and the relative error approximations are identical in nature to the previous work, and will therefore only be mentioned briefly here. For the reasons given in Chapter 5, the discussion of integration errors will again be in terms of absolute errors.

6.1 Sources of Integration Error

The Padé series was initially described in Chapter 3. There it could be seen that the diagonal Padé is the most effective. As such, only diagonal approximations will be discussed here. The analysis can be appended for the off diagonal methods. Again, for reasons outlined previously, a truncated power series will be used as a base matrix, and then this base matrix will be propagated through time. The truncated Padé series also exploits the scaling property of the exponential function.

$$e^{At} = [e^{A\Delta t}]^n$$

$$e^{At} = [M_{pp}]^n$$

The formula for the Padé approximation to the scaled matrix exponential is given as

$$M_{pp}(At) = [D_p(A\Delta t)]^{-1} N_p(A\Delta t)$$

with

$$N_p(A\Delta t) = \sum_{i=0}^{p} \frac{p!\,(2p-i)!}{(2p)!\,i!\,(p-i)!} (A\Delta t)^i$$

$$D_q(A\Delta t) = \sum_{i=0}^{q} \frac{q!\,(2p-i)!}{(2p)!\,i!\,(q-i)!} (-At)^i$$

The following table gives the first three base matrices using diagonal approximations. Higher level base matrices can easily be computed using the formula above.

Table 6-1: First Three Levels of the Padé Approximation to the Matrix Exponential

Level of Approx.	Numerator	Denominator
1	$I + \frac{A\Delta t}{2}$	$I - \frac{A\Delta t}{2}$
2	$I + \frac{A\Delta t}{2} + \frac{(A\Delta t)^2}{12}$	$I - \frac{A\Delta t}{2} + \frac{(A\Delta t)^2}{12}$
3	$I + \frac{A\Delta t}{2} + \frac{(A\Delta t)^2}{10} + \frac{(A\Delta t)^3}{120}$	$I - \frac{A\Delta t}{2} + \frac{(A\Delta t)^2}{10} - \frac{(A\Delta t)^3}{120}$

From the above table and the equation for matrix exponential, we can see two sources of error similar to that for the Taylor series. First, the integration error results from the errors incurred in determining the base matrix. The second source of error is due to the propagation of the base matrix errors in the stepping or squaring propagation through time. The two sources are again dependent on the two integration parameters. The base matrix errors are due to the number of terms in the truncated power series, and the propagation errors are due to the number of steps necessary to reach the final time. The number of steps required is directly dependent on the size of the time step, the other integration parameter.

In addition to the two sources of integration error that Padé series share with Taylor series, there is another source of error that is unique to Padé series. In order to determine the base matrix for the Padé series, a matrix inversion is required. The matrix inversion

process introduces error. This error is not necessarily integration error and not necessarily roundoff error--it is included here because of its direct relation to the Padé series and the propagation of the base matrix error.

The error of the matrix inversion process is difficult to bound *a priori*. Any bound would depend on the system parameters rather than the integration parameters. The matrix $D_p(A\Delta t)$ may be poorly conditioned with respect to the inversion causing large errors. In particular, widely spread eigenvalues of $A\Delta t$ lead to a poorly conditioned denominator matrix [Møller, 1978]. Also, the relative error for a matrix inversion has not, to the author's knowledge, been bounded *a priori* without explicitly containing the computed inverse in the error bound [Wilkinson, 1961; Ward, 1977].

Relative error bounds for matrix inversion may be determined *a posteriori*. Unfortunately, these relative error bounds make use of the calculated inverse matrix to determine the error. Because of this drawback, most algorithms track the error through the inversion process and then produce the bound [Møller, 1978; Ward, 1977]. One of the main premises of this paper, however, is to develop methods of bounding the error *a priori* so that the final computed result is not a necessary part of the error bound. In this manner, the integration process could be chosen automatically and still insure the integrity of the results. Clearly, the Padé series approach does not meet these criteria for a general case of the matrices encountered in the solution of Markov models because the inversion process has not been bounded *a priori*.

Nonetheless, the Padé approximation still deserves proper attention. Some people may still want to use the Padé approximations in conjunction with the rest of the *a priori* error bounds. One can temporarily ignore the error of the inversion process until the conclusion, and then simply add in the *a posteriori* error bound. For this reason, the analysis for the integration error for the Padé series is presented in general terms. Throughout the analysis, the error for the inversion process will be denoted by the additive error E_i.

6.2 Propagation of Integration Error

One source of the base matrix error is the inaccuracies due to the truncated series approximation to the matrix exponential. It is not clear from inspection what order of the approximation the Padé series yields. It was shown in Chapter 3, however, that the order of the diagonal Padé series is 2p, where p is the highest power in the Padé series. By

appealing to the scalar case analogy and referencing the results of Chapter 5, we can use the following bound for the error of the base matrix.

$$\text{AIE (base matrix using Padé)} \leq \left|\left| \frac{(A\Delta t)^{2p+1}}{(2p+1)!} + E_i \right|\right|$$

where E_i is the additive error due to the inversion process. The scalar analogy was used to assume that the denominator polynomial could be divided into the numerator polynomial. The requirement that the transition matrix norm be less than unity was borrowed from Chapter 5. Thus, the magnitude of the norm decreases as the power of the matrix increases.

$$||(A\Delta t)^p|| > ||(A\Delta t)^{p+1}||$$

Additionally, the equation above uses the fact that the sign of the terms in the matrix $A\Delta t$ changes with each increase in the power of the matrix.

This base matrix error must be propagated through the solution algorithm so that the error for the entire interval of interest will be bounded. Chapter 5 showed that, for the stepping and squaring algorithms, the error can be bounded by

$$\text{AIE (Padé)} \leq ||n\, I\, E||$$

where E is the error of the base matrix. For Padé series, this equation yields an integration error bound of

$$\text{AIE (Padé)} \leq \left|\left| n \left(\frac{(A\Delta t)^{2p+1}}{(2p+1)!} + E_i \right) \right|\right|$$

This error bound is easily manipulated using the properties of the matrix norm discussed in the previous chapter. Using the triangle inequality and the scalar multiplicative property, the integration error becomes

$$\text{AIE (Padé)} \leq \frac{n\,(\Delta t)^{2p+1}}{(2p+1)!} ||A||^{2p+1} + n||E_i||$$

It can also be rewritten in terms of the final time instead of the number of time steps.

$$\text{AIE (Padé)} \leq \frac{t\,(\Delta t)^{2p}}{(2p+1)!} ||A||^{2p+1} + n||E_i||$$

Thus, we now have an integration error bound for the truncated Padé series approximation. The bound is dependent on the two integration parameters of the time step and the number of terms in the truncated power series. It is also dependent on the transition matrix A. Lastly, the error bound depends on the error of the inversion process. As was stated earlier, this error may be ignored initially and added back in once it has been determined *a posteriori*.

6.3 Richardson Extrapolation

The method of Richardson extrapolation for Padé series is based on the same premise as the method for Taylor series. The idea is to integrate the problem twice--once with the full time step and once with a half time step. The two results are then combined to give a better approximation than either of the individual results. It is a means of approximating the results in the limit as the time step goes to zero.

Section 5.3 showed that the extrapolation parameters for a Taylor series approach was

$$EP_j \text{ (Taylor)} = 2^{k+j} \qquad j = 0, 1, 2, \ldots$$

where k is the order of the approximation for the truncated Taylor series, and j is the number of extrapolation levels completed. However, Padé approximation achieve twice the order of approximation for an equivalent power of the base matrix. Accounting for this difference in the order of approximation, we arrive at the extrapolation parameters for the Padé series approach to Richardson extrapolation.

$$EP_j \text{ (Padé)} = 2^{2(p+j)} \qquad j = 0, 1, 2, \ldots$$

Again, the extrapolation must be divided by a factor one less than the extrapolation parameter to insure that the value is only one state probability. The general equation for the probability state vector using Richardson extrapolation with a Padé approximation is

$$P_{j+1} \text{ (Richardson with Padé)} = \frac{EP_j \bullet P'_j - P_j}{EP_j - 1}$$

with it understood that EP_j is the extrapolation parameter for the Padé series. The prime (') signifies that value has the smaller time step. The absolute integration error would then be

$$AIE_{j+1}\ (\text{Richardson with Padé}) \leq \frac{\left[EP_j \bullet AIE'_j\right] - AIE_j}{EP_j - 1}$$

AIE_j and AIE_j' can be determined using the methods described in section 6.2.

Richardson extrapolation for Padé approximations is very similar to the extrapolation using Taylor approximations. We have developed the extrapolation parameters necessary and have also bound the absolute integration error for extrapolation with Padé. The next two sections will present some numerical results for the various solution techniques using Padé and Richardson.

6.4 Error of the Padé Series

Section 6.2 developed the error bounds for the Padé approximations. These bounds are functions of the time step and the number of terms in the base matrix power series. The error bounds do not include the error due to the inversion process since we are primarily concerned with *a priori* bounds. The purpose of this section is to give some numerical results for a sample system which is solved using Padé approximations. The error bound calculation is also included to provide a reference point. In a like manner to the error for the Taylor series approach, the number of steps is small so that the roundoff error is kept to a minimum. Also, the integration was run in double precision to nullify the roundoff error effects.

The sample system is the four state Markov model that is familiar by now. The transition matrix is the same as used for Taylor series.

$$a = 10^{-3}\ /\text{hour}$$
$$b = 10^{-4}\ /\text{hour}$$
$$t_1 = 100\ \text{hours}$$
$$t_2 = 20000\ \text{hours}$$

Again, final times both before and after the characteristic time were run for comparison purposes. The chosen solution method was squaring algorithm with variable renormalization. The first two base matrices M_{11}, M_{22} were used.

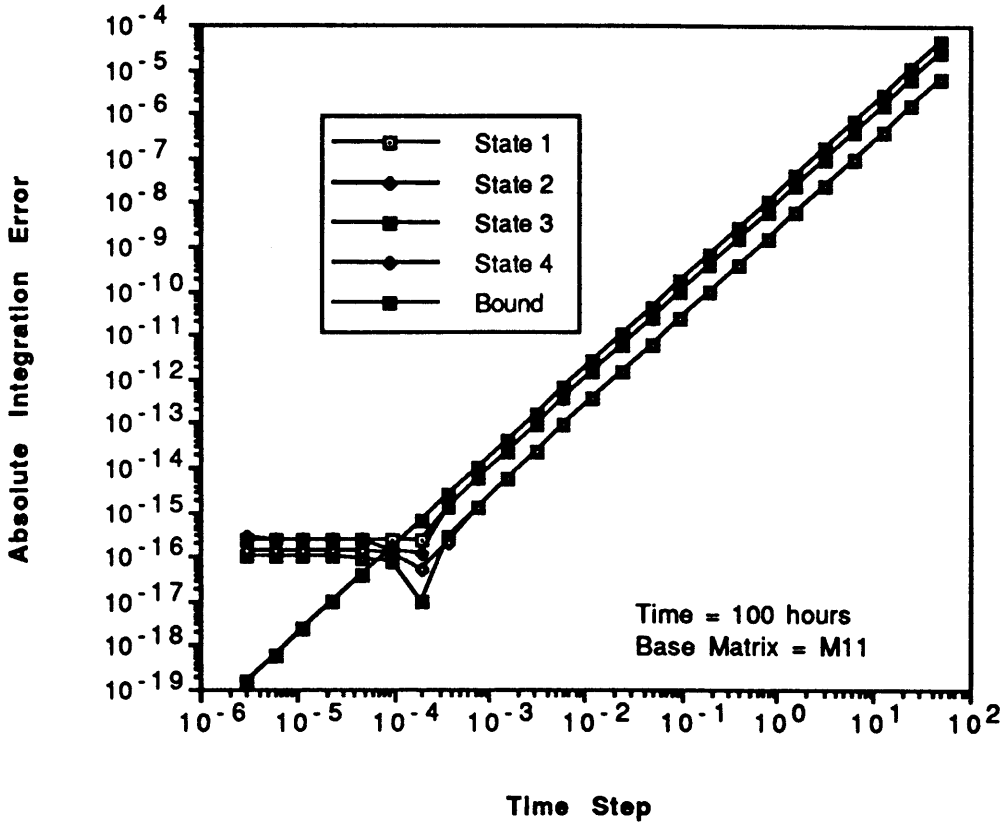


Figure 6-1: Absolute Integration Error using M_{11} versus Time Step before the Characteristic Time

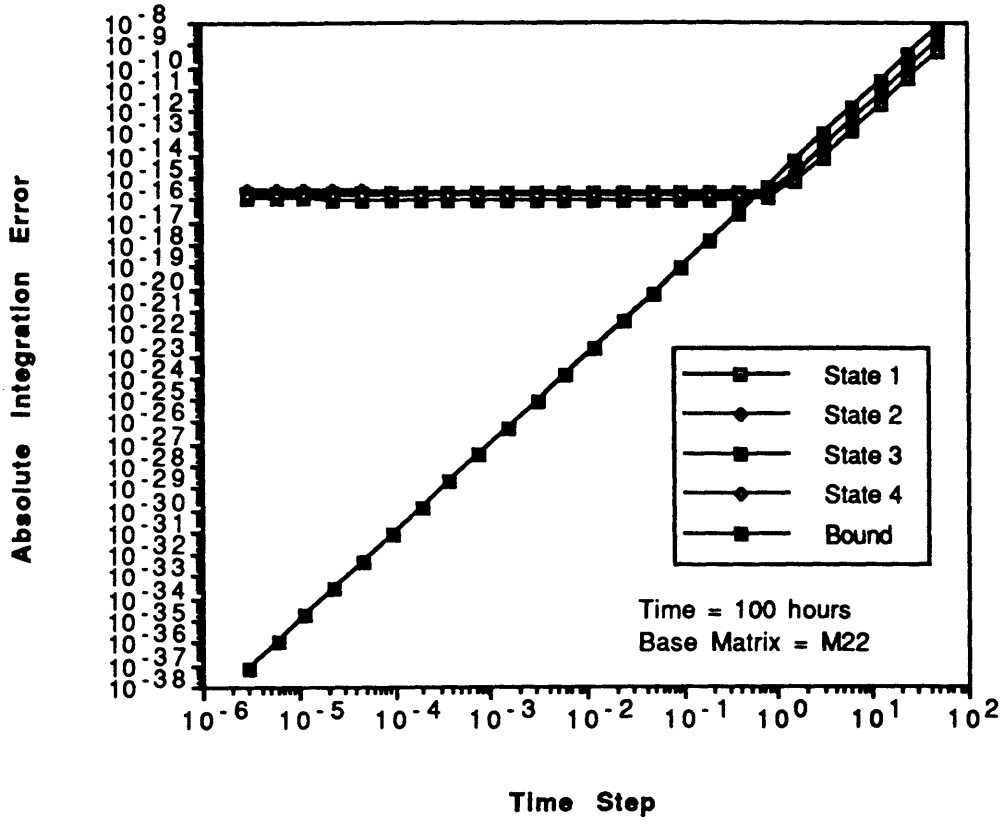


Figure 6-2: Absolute Integration Error using M_{22} versus Time Step before the Characteristic Time

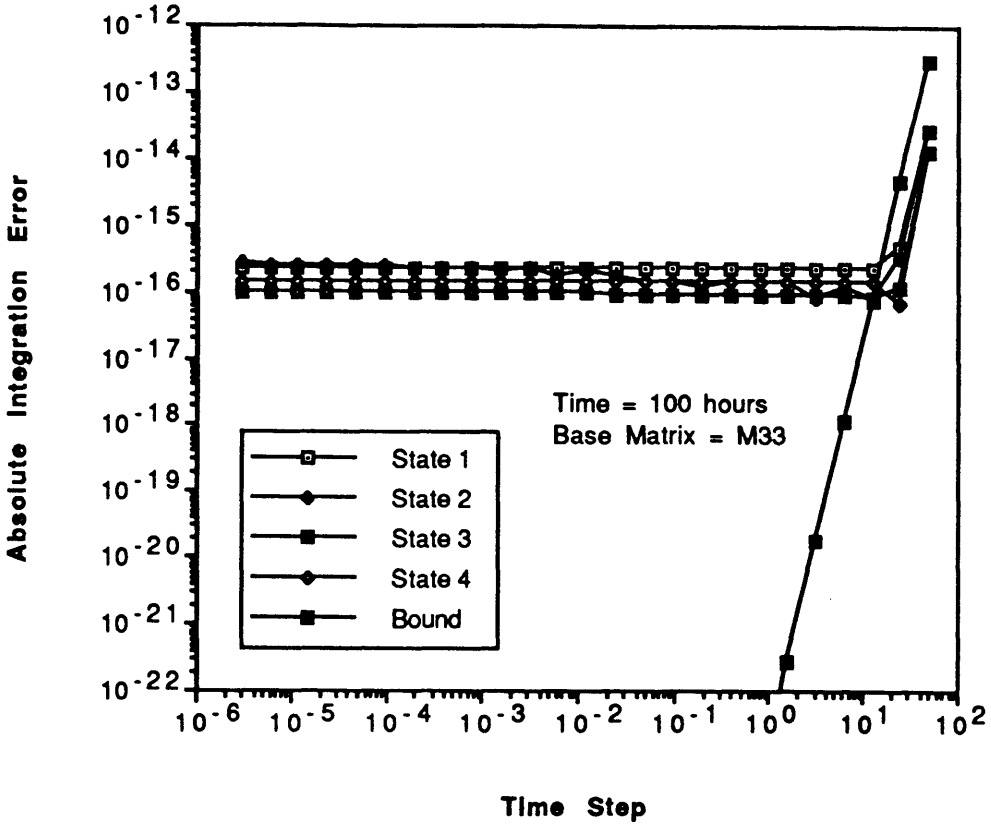


Figure 6-3: Absolute Integration Error using M_{33} versus Time Step before the Characteristic Time

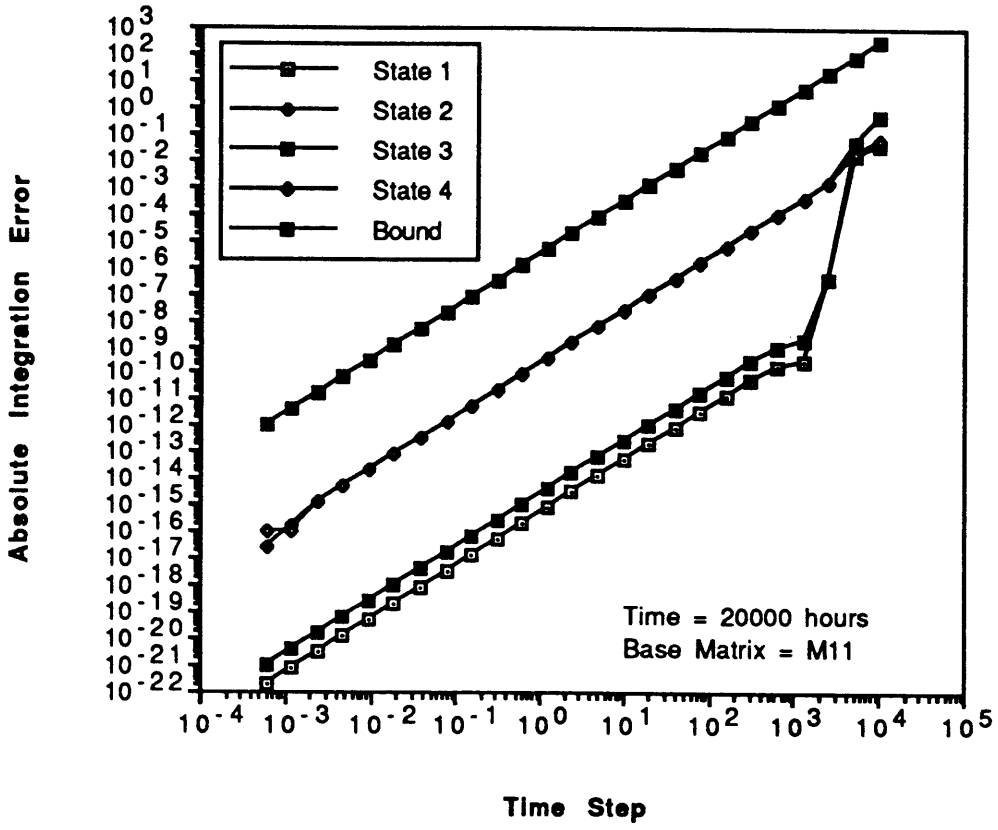


Figure 6-4: Absolute Integration Error using M_{11} versus Time Step after the Characteristic Time

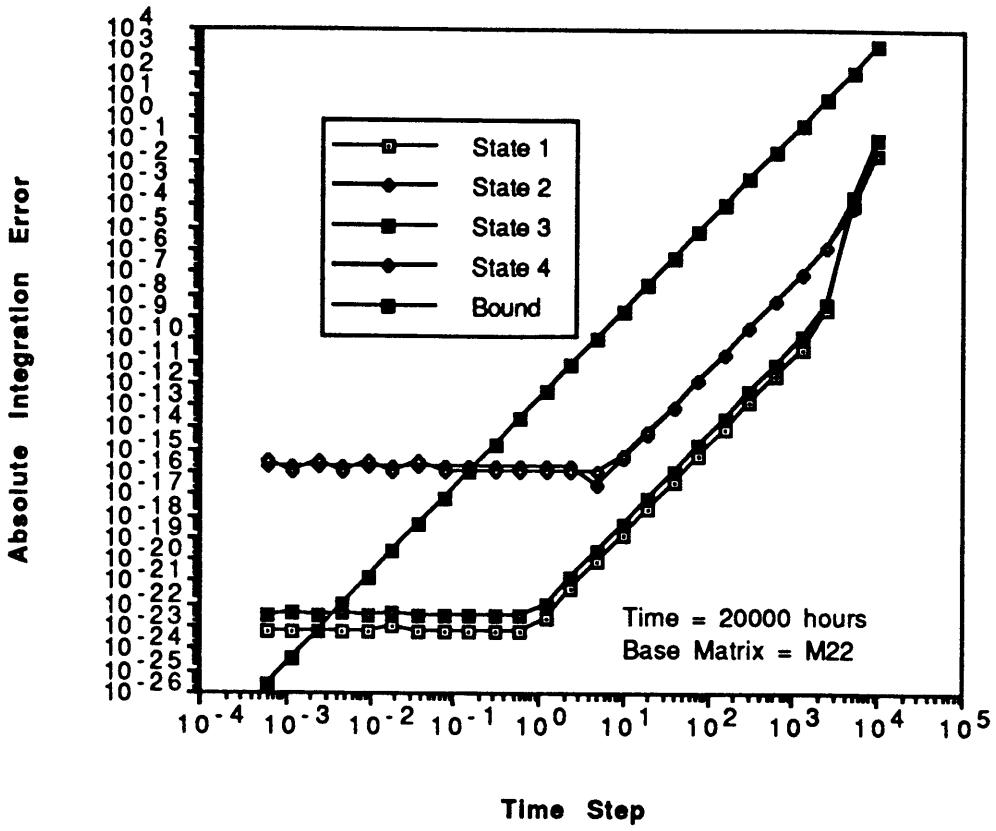


Figure 6-5: Absolute Integration Error using M_{22} versus Time Step after the Characteristic Time

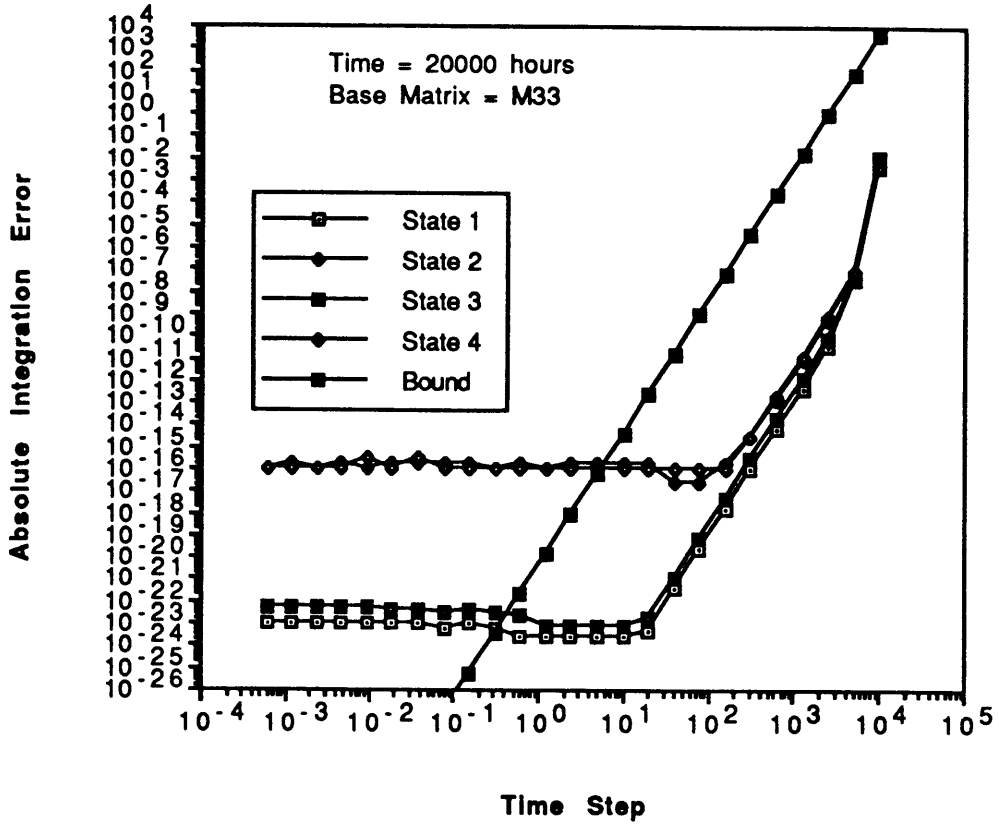


Figure 6-6: Absolute Integration Error using M_{33} versus Time Step after the Characteristic Time

The first three graphs (Figures 6-1, 6-2 and 6-3) display the results for an integration with a final time less than the characteristic time. Clearly, increasing the base matrix from M_{11} to M_{22} or M_{33} greatly reduces the amount of integration error. For example, a time step of 10 hours gives an absolute integration error of 10^{-6} using a M_{11} base matrix. This same time step using a M_{22} base matrix only produces and integration error of 10^{-11}, while the machine precision of 10^{-16} is reached if M_{33} is used. The flat portion of the three graphs indicates that the machine precision has been reached at these values for a time step.

The latter three graphs (Figures 6-4, 6-5 and 6-6) present the results for an integration with a final time greater than the characteristic time. Again, increasing the base matrix from M_{11} to M_{22} or M_{33} greatly decreases the absolute error. The change in base matrix causes a significant change in integration error because an increase of one in the

power of the truncated series for the base matrix translates into an increase of two in the order of the approximation. The flat portions of Figures 6-5 and 6-6 indicate that the machine precision has been reached. The unusual pattern at the larger time steps indicates that the norm of the base matrix was not less than unity.

An error bound is plotted in all four graphs. The equation used is the more conservative one, with the matrix norm raised to a power. In all cases the bound provided and upper limit to the absolute integration error actually experienced. Because of the inequalities in attaining the bounding equation, the bounds are much tighter for systems with a final time less than the characteristic time. If tight bounds are desired for a system with a final time greater than the characteristic time, one of the less conservative equations should be used. Invariably, this means an increase in the amount of computer work.

6.5 Error of the Padé Series with Richardson Extrapolation

Numerical results for the extrapolation methods are given below. Again, the time step used for these data is the time step in the first integration pass of the extrapolation process. These results were generated using the system of the previous section. Characteristic times both greater than and less than unity were used. Again, only the first two base matrices (M_{11} and M_{22}) were the premises of the integration schemes. To keep the graphs simple, only the integration errors for State 1 is shown. If possible, output was generated for two levels of extrapolation to show its effective use when applied successively.

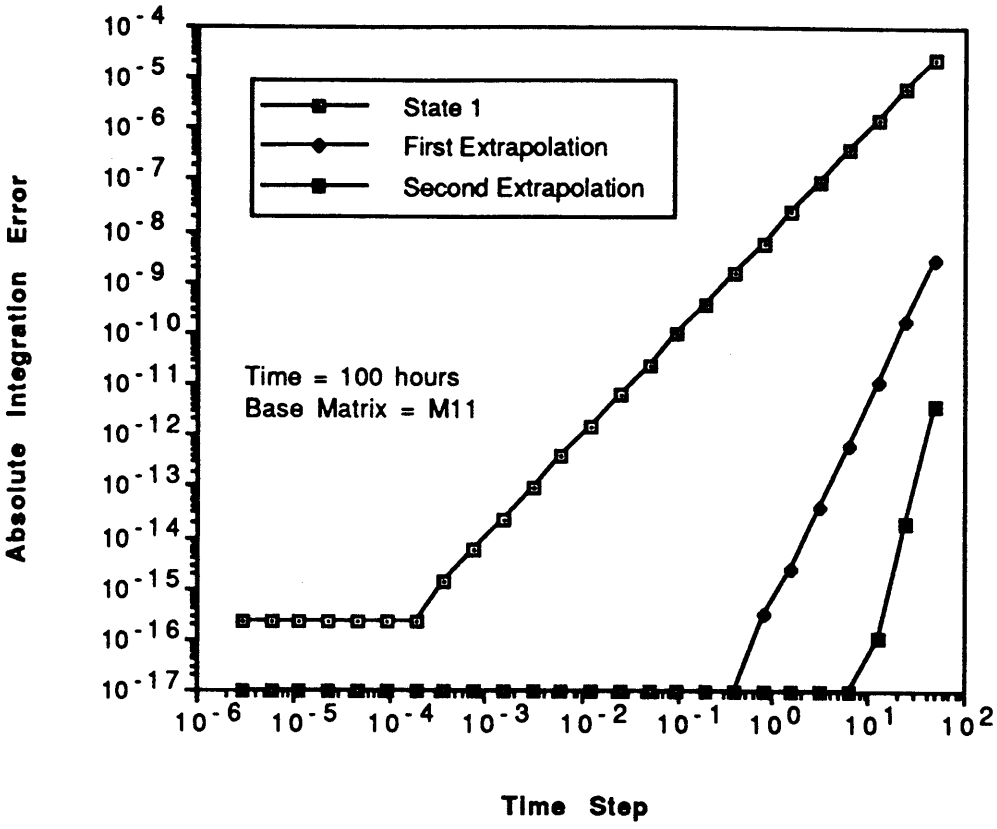


Figure 6-7: Absolute Integration Error using M_{11} versus Time Step before the Characteristic Time with Richardson Extrapolation

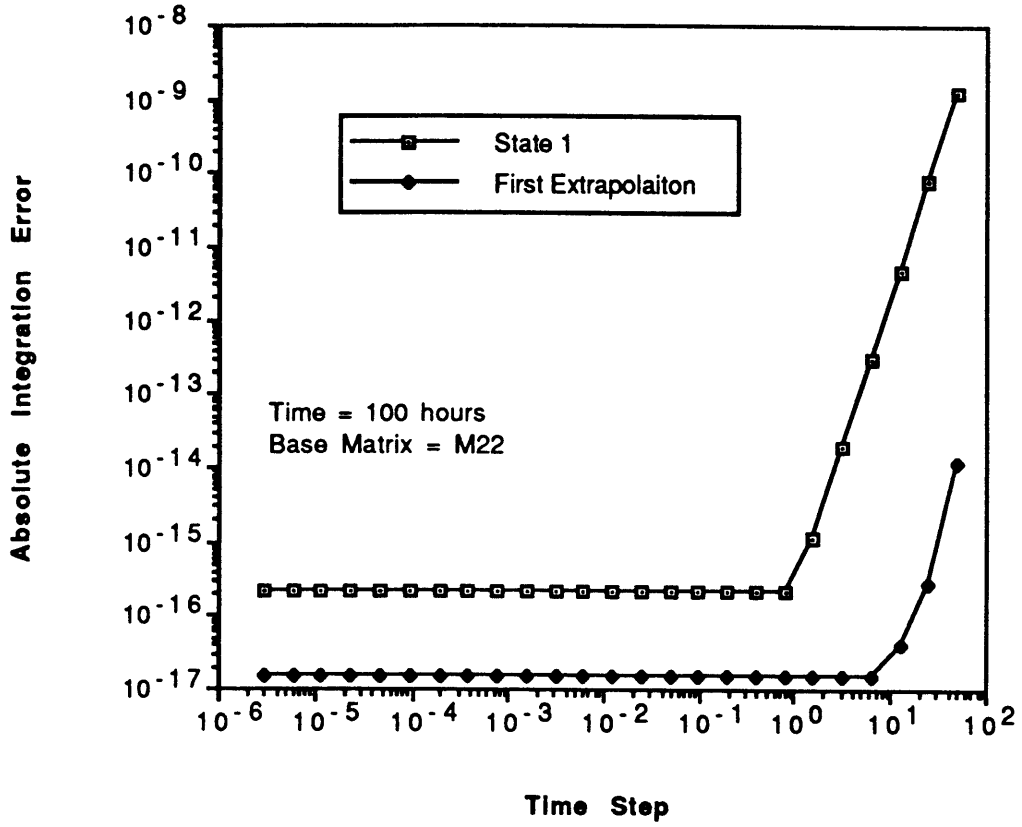


Figure 6-8: Absolute Integration Error using M_{22} versus Time Step before the Characteristic Time with Richardson Extrapolation

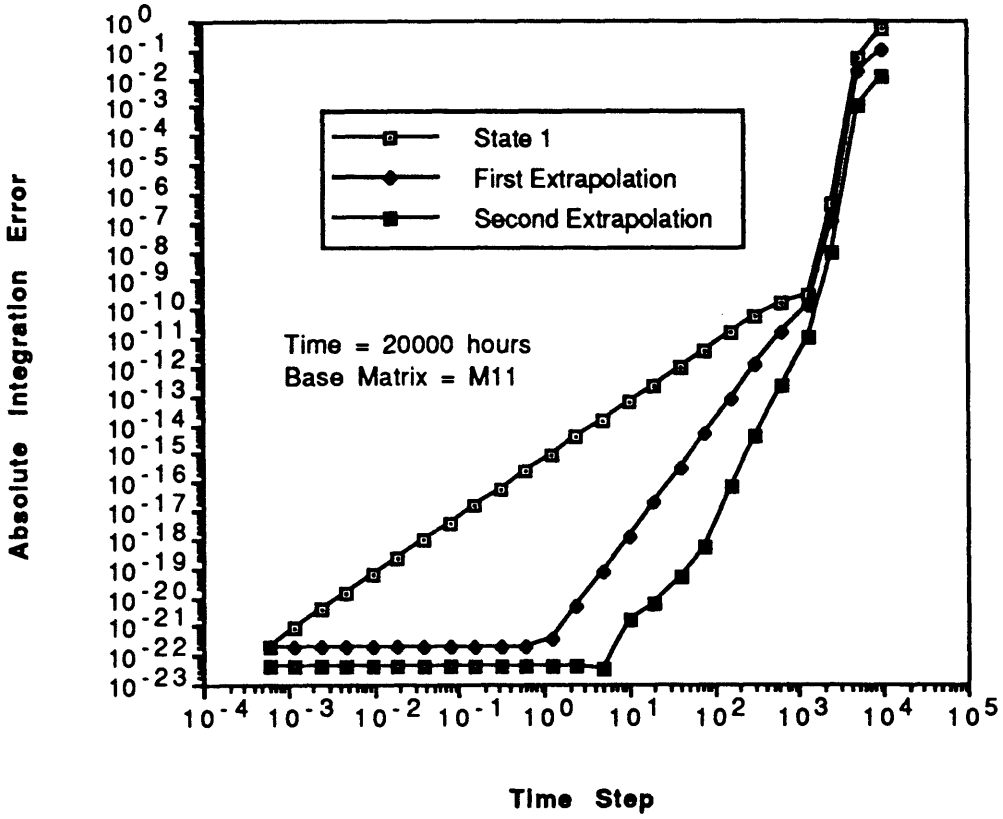


Figure 6-9: Absolute Integration Error using M_{11} versus Time Step after the Characteristic Time with Richardson Extrapolation

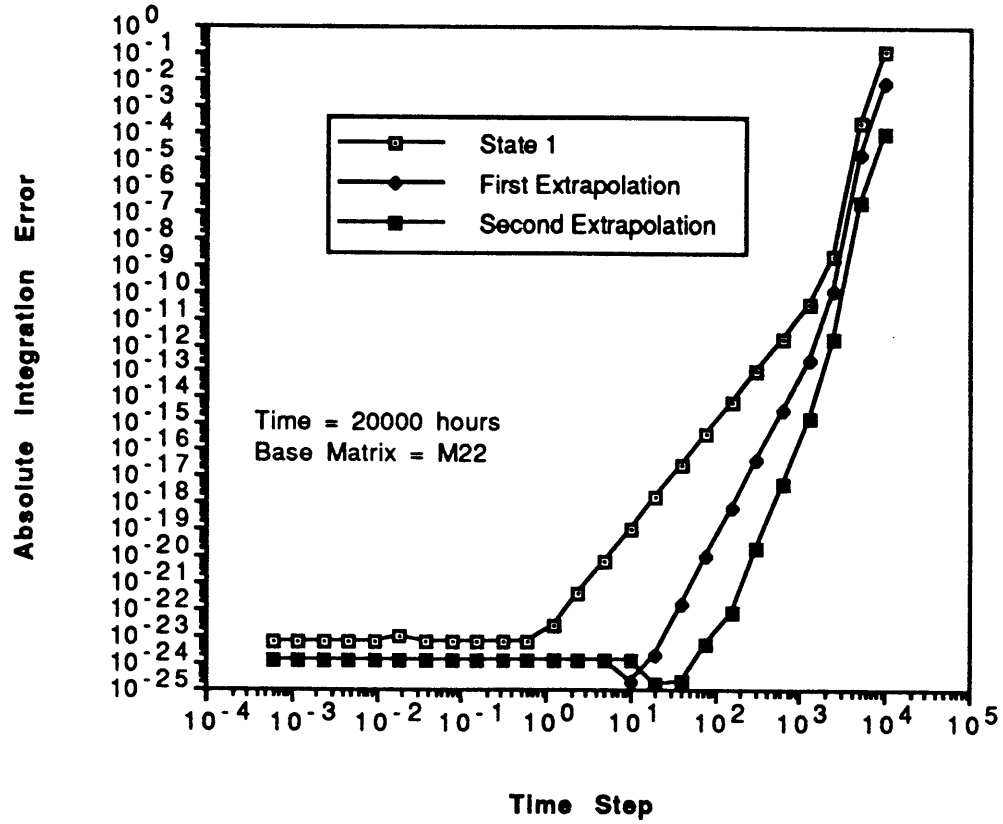


Figure 6-10: Absolute Integration Error using M_{22} versus Time Step after the Characteristic Time with Richardson Extrapolation

These four graphs present the absolute integration error for an integration algorithm using Padé approximations with Richardson extrapolation. The first two (Figures 6-7 and 6-8) present the integration error for a system whose final time is less than the characteristic time of the system. The extrapolation method is very effective in these cases due to the nature of the Padé approximation. A second extrapolation could not be conducted for the case where M_{22} was used because of the limits of machine precision. The varying slopes of the extrapolation indicate the different orders of approximation.

The latter two graphs (Figures 6-9 and 6-10) present the integration error for a system whose final time is greater than the characteristic time of the system. Because of the longer time, an second extrapolation was possible for both M_{11} and M_{22}. The different slopes again demonstrate the different orders of approximation achieved for the different methods.

Note that the data for Richardson extrapolation is given versus the time step. This data is misleading if one is primarily concerned about the amount of computer work necessary to compute the values. In most cases, the extrapolation method requires approximately twice as much work to achieve the additional accuracy. The question of whether this is an efficient method is left open. The following chapter discusses the issue of accuracy versus computer cost in detail.

6.6 Chain Model Approximations

Chain model approximations can be used with Padé approximations in the same manner they were applied to the Taylor approximations (see Section 5.6). The same transition matrix is used for the chain model.

$$A\Delta t = \begin{bmatrix} -a\Delta t & 0 & 0 & 0 & 0 \\ +a\Delta t & -a\Delta t & 0 & 0 & 0 \\ 0 & +a\Delta t & -a\Delta t & 0 & 0 \\ 0 & 0 & +a\Delta t & -a\Delta t & 0 \\ 0 & 0 & 0 & +a\Delta t & 0 \end{bmatrix}$$

It was shown that the coefficient for the terms after increased matrix multiplication fit the pattern of the binomial coefficient. In terms of the error bound for the Padé approximations, this becomes

$$\begin{pmatrix} 2p+1 \\ f \end{pmatrix} = \frac{(2p+1)!}{(2p+1-f)!\ f!}$$

where f is the failure level in the chain model for the state of interest. The chain model approximation then leads to the following equation for the integration error of the f^{th} failure level

$$AIE_f \text{ (Padé chain)} \leq \frac{n\begin{pmatrix} 2p+1 \\ f \end{pmatrix}(\lambda\Delta t)^{2p+1}}{(2p+1)!} + n E_i$$

where λ is the transition rate used in the chain model. This equation, like many of the previous ones, can be rewritten so that it is only in terms of the time step.

$$AIE_f \text{ (Padé chain)} \leq \frac{\begin{pmatrix} 2p+1 \\ f \end{pmatrix} t\,(\Delta t)^{2p}\,(\lambda)^{2p+1}}{(2p+1)!} + n E_i$$

We now have error bound equations for the Padé approximation that are very similar to the bounds developed for the Taylor approximation. Note, however, that the error of the inversion process has been neglected in the above derivation and should be considered. This unknown error causes these chain model approximations to be even more suspect.

6.7 Relative Error Approximations for the Integration Error

This chapter has described the integration error in terms of absolute error for the Padé series approximations to the matrix exponential. While it is easier to describe the integration error in terms of absolute error, it is often more useful to give the integration error in terms of relative error as was discussed in Section 5.7.

Relative error is defined as the absolute error divided by the exact answer. Since equations for the absolute error have already been developed, we need a value for the exact answer. The best means we have to *a priori* approximate a state probability is by means of the chain model approximations. Rather than repeat the derivation of the relative error, we simply state that the process for determining the relative error approximations for Padé series parallels the discussion given in Chapter 5 for the relative error approximations for Taylor series approximations.

The lower bound for the probability of a state in the chain model is given below.

$$\text{factor} = \binom{n^*}{f} (\lambda_{min}\,\Delta t^*)^f\,(1 - \lambda_{max}\,\Delta t^*)^{n^*-f}$$

where n* is large enough and Δt^* is small enough such that 'factor' is convergent to an acceptable accuracy. The relation $n^*\Delta t^* = t$ is still valid. Note that this is not a Padé approximation to the chain model, rather it is a Taylor approximation. The calculation of 'factor' is an approximation to the state probability. Therefore, it is not important which method is used, just that a suitable accuracy can be achieved.

Using the above equation for an approximation to the probability for the state of interest yields the following equation for the relative integration error using the Padé series approach.

$$RIE_f\,(\text{Padé}) \le \frac{t\,(\Delta t)^{2p}\,\|A\|^{2p+1} + n\,E_i}{(2p+1)!\binom{n^*}{f}(\lambda_{min}\Delta t^*)^f\,(1 - \lambda_{max}\,\Delta t^*)^{n^*-f}}$$

Thus, we now have a complete set of integration error equations for the Padé series approach to the solution of the matrix exponential.

6.8 Recapitulation

This chapter was devoted to a discussion of the integration error incurred when the truncated Padé power series is used to determine the matrix exponential. Because chapter 5 dealt with this topic for the case of the Taylor series, much of the material was borrowed from the previous chapter. Often the details were left out and the reader was referred to the preceding work.

The first task was to recognize the sources of error. Two sources of error similar to those discussed for the Taylor series were identified. These sources were shown to be directly dependent upon the two integration parameters. The number of terms in the truncated power series defines the error of the base matrix, and the size of the time step defines the propagation of this base matrix error. In addition to these sources of error, the base matrix has error due to the inversion process required by the Padé series. This error has not been bounded *a priori* to the author's knowledge. As such, it was left as a general error and carried through the analysis. In most cases, the inversion error is ignored and assumed to be negligent for the purposes of selecting the integration parameters. The inversion error can be determined once the inversion has taken place. At this point, the error should be added back in to insure the proper accuracy.

The parameters for Richardson extrapolation were developed for the specific case of Padé series. Again, the manner in which the initial approximations are extrapolated depends on the number of terms in the base matrix and the level of extrapolation. Guidelines for repeated extrapolation were developed, along with equations for the extrapolation parameters. The equations for the integration error of the extrapolated values were also given.

Numerical results were given to provide comparison for the various techniques developed. The numerical results also demonstrated the the error bounds developed do provide for an upper limit to the integration error. Results were given for both before and after the characteristic time. The system integrated was the same as for the Taylor series approach to allow further comparison.

Lastly, equations for the chain model approximation were also developed. This method allows for more flexibility in determining integration error. However, the error

equations are no longer guaranteed to bound the actual error experienced. Additionally, equations for the integration error in terms of relative error were developed using the chain model approximation. Again, the equations are not guaranteed to bound the actual error incurred.

At this point, the integration error propagation patterns for both Taylor and Padé series have been developed. In addition, the roundoff error patterns have been developed for the various integration options. The problem now is to develop a mechanism and guidelines by which a certain integration method and the integration patterns should be chosen to deliver the desired accuracy in the least amount of computer work. This problem is the subject of the next two chapters.

Chapter 7

Equivalent Work Comparisons

The previous chapters have described the roundoff and integration errors and their propagation patterns for various integration methods. Additionally, the error propagation patterns for several error reduction techniques have been discussed. For many of these techniques, the basis for comparison for integration error was the order of the approximation of the order of the error obtained, and the basis for comparison for roundoff error is simply the cumulative relative error. A method was considered superior if it increased the order of the approximation or reduced the overall cumulative error.

In this chapter a new basis for comparison of integration schemes is introduced. Often times, an integration algorithm will yield an improvement in accuracy at a great increase in computer work. For instance, increasing the number of terms in the base matrix and cutting the time step in half both decrease the error of the approximation. In this case, however, it is not clear which would be more advantageous. Thus, the amount of computer work required to obtain a certain accuracy will be used as the new basis for comparison of the potential benefit of the various integration schemes.

The data will be presented primarily in terms of equivalent work. That is, what accuracy can be obtained for a given amount of computer work. The data may also be presented as how much work is required to achieve a certain accuracy, as this is how the information is usually implemented in algorithms. Lastly, the data will be presented in the most convenient form for the situation at hand; most likely that is, relative error for roundoff errors and absolute error for integration errors.

This chapter will first discuss the idea of computer work and the amount of computer work necessary to approximate the matrix exponential using the assorted integration techniques. Equations for the amount of work required for the integration will be developed in terms of the system and the integration parameters. After the basic equations for computer work for each of the methods are given, numerical results will be generated comparing these algorithms. Once the means to determine the better algorithms (in terms of computer work) have been set, there needs to be a means to automatically select the integration parameters that meet the specifications. The automatic selection of system parameters is the topic of Chapter 8.

7.1 Computer Work

A unit of computer work described earlier was the floating point operation or the 'flop' (see Section 3.5). It was shown in that section that it requires m^3 flops for a matrix-multiply-matrix operation, and m^2 flops for a matrix-multiply-vector operation, where m is the dimension of the matrix. In this chapter, we introduce two simpler measures of computer work that remove the dependence on the dimension of the matrix. One matrix-multiply-matrix operation will be called a *mop* and one matrix-multiply-vector operation will be called a *vop*. These will be thought of as single units of computer work for comparison purposes.

In most cases, a mop or a vop will properly describe the amount of computer work expended in arriving at a solution. In the few cases where these are inadequate, the flop will be used with the dimension of the matrix stated specifically. Additionally, since most of the solution techniques require a multiplication of the final transition matrix and the state vector, this operation will not be included in the overall count of computer work. Computer work that is of lower order (that is, an m^2 operation when most other operations are m^3) will be neglected.

7.1.1 Work Associated with Taylor Series Approximations

The amount of computer work necessary to use compute the matrix exponential with Taylor series approximations depends on the two integration parameters of the time step and the number of terms in the truncated series determining the base matrix. Each term in the truncated series requires one additional mop to calculate. This value, of course, assumes that lower ordered matrices $(A\Delta t)^x$ are saved and used to calculate the higher ordered matrices $(A\Delta t)^{x+1}$. Thus, the amount of computer work CW required to determine the base matrix is

$$CW(base) = (k-1) \text{ mops}$$

where k is the highest power of the transition matrix in the base matrix. This equation can be looked at as the initial cost of the integration scheme. A higher initial cost, it is hoped, will yield lower secondary costs yielding a lower total cost.

The computer work associated with the time step is dependent on the scaling and squaring parameters for the integration routine. We know that $n\Delta t = t$ applies. If the matrix exponential for the total time is determined by stepping up to the final time, the

algorithm uses n matrix-multiply-vector operations. Thus, the computer work associated with the stepping method is

$$\text{CW(stepping)} = \text{n vops}$$

If, on the other hand, the matrix exponential for the total time is determined by squaring up to the final time with no stepping, the equation $2^{\alpha}\,\Delta t = t$ applies, where α is the number of squarings of the base matrix. Clearly then, $2^{\alpha} = n$, and the amount of computer work is

$$\text{CW(squaring)} = \alpha \text{ mops} = \frac{\ln n}{\ln 2} \text{ mops}$$

However, it was shown in section 3.5 that the computer work can be minimized for the Taylor series methods using a combination of the squaring and the stepping. We defined s as the point at which we switch from squaring to stepping. Thus, the computer work associated with a combined method is

$$\text{CW(combined)} = \frac{\ln s}{\ln 2} \text{mops} + (n\text{-}s) \text{ vops}$$

The minimum amount of work is achieved when the switching point is $s = \frac{n \ln 2}{m}$.

Unfortunately, the combined method presents a problem in presenting data. The computer work is dependent on both mops and vops. This implies that the actual amount of work is dependent on the dimension of the matrix. This dependence makes direct comparison more difficult. Therefore, where appropriate, the work for the squaring routine will be presented with the understanding that this work is slightly more than would be necessary. An example of the combined method will be compared to the stepping and squaring routines to round out the presentation.

7.1.2 Work Associated with Padé Series Approximations

Like the Taylor series approximations, the amount of computer work necessary to use compute the matrix exponential with diagonal Padé series approximations depend on the two integration parameters of the time step and the number of terms in the truncated series determining the base matrix However, the Padé series approximations require one to compute a numerator and a denominator. Each term in the numerator and the denominator are the same with the signs changed. Thus, only (p-1) mops are necessary to

calculate both of these matrices, if lower ordered matrices are saved and used to calculate the higher ordered matrices. The denominator must also be inverted, an m^3 operation equivalent to one mop. Lastly, the inverted denominator must pre-multiply the numerator for another mop. Thus, the total amount of computer work necessary to determine the Padé series base matrix is

$$CW(base) = (p+1) \text{ mops}$$

Once the base matrix has been determined, the Padé series approximation and the Taylor series approximation are operated on in the same manner. We could step through the integration using n matrix-multiply-vector operations, for a computer work of

$$CW(stepping) = n \text{ vops}$$

Additionally, we could use a squaring algorithm for a computer work of

$$CW(squaring) = \frac{\ln n}{\ln 2} \text{ mops}$$

However, as was discussed earlier, a combination of squaring and stepping results in the least amount of computer work, where s is the switching point

$$CW(combined) = \frac{\ln s}{\ln 2} \text{mops} + (n-s) \text{ vops}$$

Again, the combined method presents a problem for presenting data. The dependence on both mops and vops implies a dependence on the dimension of the matrix, hindering direct comparison between methods. To present data, the same guidelines will be used for the Padé series that are used for the Taylor series. The squaring routine will be used with the understanding that the computer work is slightly more than would be necessary. Also, a combined method will be given for comparison to the stepping and squaring routines.

Clearly, other than the base matrix, the computer work for the Padé series approximation is the same as that for the Taylor series. The Padé base matrix requires more computer work than Taylor for an equal number of terms. However, the Padé series yields a higher order approximation, 2p rather than k. Thus, the tradeoff is whether the higher order approximation is worth the additional work in determining the Padé base matrix. This study will be given in section 7.5.

7.1.3 Work Associated with Accumulator Methods

The computer work associated with accumulator methods varies with the base matrix chosen. This is expected since the algorithm itself changes with the different base matrices.

Accumulators essentially subtract everything they add to see what accuracy was lost. In this way they at least double the computer work when compared to the regular stepping routines. In addition, the base matrix must be broken down into it components to account for the discrepancies that may arise due to these inaccuracies, again increasing the amount of computer work required.

Counting the amount of computer work for the accumulator integration schemes given in Section 4.2.2, we come up with the following approximate values for computer work using M_1 and M_2.

$$\text{CW (Stepping with accumulator, } M_1) = 4n \text{ vops}$$

$$\text{CW (Stepping with accumulator, } M_2) = 10n \text{ vops}$$

We now look at these two cases in an effort to derive a general equation for the amount of computer work associated with an accumulator. We will start with M_1. Both the error vector and the state vector must multiply the base matrix. Since the base matrix only has $I+A\Delta t$, this requires only 2 vops (the state vector and the error vector do not need to be multiplied by the identity matrix). In order to accumulate the error, it is necessary to a number by a column n times, amounting to one vop. It is also necessary to subtract vectors n times, amounting to another vop. There are seven other operations that are of lower order and neglected in this count.

The only difference between M_1 and M_2 is the additional term in the base matrix. This additional term clearly adds another four vops by the same way that M_1 required four vops. The extra two vops is due to the addition of the $A\Delta t$ and the $\frac{(A\Delta t)^2}{2}$ terms. An additional 'accumulation' is required.

We can now extrapolate this reasoning to arrive at the amount of computer work for the general accumulator method. Clearly, an additional four vops is required for each increase in the order of the base matrix. Also, because the additional order of M will

require another addition to be 'accumulated', two more vops will be required. The final equation for the general accumulator method becomes

$$\text{CW (stepping with accumulator)} = (6k - 2)n \text{ vops}$$

It is clear that accumulator methods are very costly compared to the normal stepping routines. The increase in cost dictates that these routines should be used only as a last resort.

7.1.4 Work Associated with Richardson Extrapolations

Richardson extrapolations are based on a combination of other integration schemes. Logically then, the computer work associated with Richardson extrapolations will be a combination of the computer work necessary for the other integration schemes that make up the extrapolation. These other integration methods could be Taylor or Padé series approximations. Additionally, they could be either stepping routines, squaring routines, or a combination of the stepping and squaring routines.

In general, every Richardson extrapolation has a few essential elements. Most obvious of these elements is the two integration passes. The first integration pass is done with a full time step, requiring n steps. Depending on the method used, this results in three possible computer work equations.

$$\text{CW(stepping)} = n \text{ vops}$$

$$\text{CW (squaring)} = \frac{\ln n}{\ln 2} \text{ mops}$$

$$\text{CW (combined)} = \frac{\ln s}{\ln 2} \text{ mops} + (n\text{-}s) \text{ vops}$$

The second integration pass uses a half time step, requiring 2n steps. Clearly, if a stepping routine is used, the second pass will be twice the work of the first stepping pass. If, on the other hand, a squaring algorithm is used, the second pass only requires one additional squaring. Thus, the second pass using squaring requires only one more mop than the first pass.

The combined method is a little more complicated to compare. The switching point is discrete because of the nature of the integration. Due to this discreteness, there are two possible scenarios for the second integration pass using a combined method. In the first case, the same number of matrix steppings are done. One additional matrix squaring is

performed increasing the amount of work by one mop. This case is easily understood in that squaring a base matrix with one-half unit time step yields a new matrix with one unit time step. For the second case, the same number of matrix squarings are done. This means that an additional n steps, or n vops, is required. Clearly, when the number of steps is larger than the dimension of the matrix, an additional squaring is done. On the other hand, when the number of steps is less than the dimension of the matrix, additional steppings are done.

We can now give the computer work necessary to use Richardson extrapolation due to the two integration passes.

$$\text{CW(stepping)} = (j+1)n \text{ vops}$$

$$\text{CW (squaring)} = \left[(j+1)\left(\frac{\ln n}{\ln 2} \right) + j \right] \text{mops}$$

$$\text{CW (combined)} = \left[(j+1)\left(\frac{\ln s}{\ln 2} \right) + j \right] \text{mops} + (j+1)(n-s) \text{ vops} \qquad (n > m)$$

$$\text{CW (combined)} = \left[(j+1)\left(\frac{\ln s}{\ln 2} \right) \right] \text{mops} + (j+1)(2n-s) \text{ vops} \qquad (n < m)$$

where j is the level of extrapolation, j=0, 1, 2,.... Again, s is the switching point in the combined method.

The other element of computer work for Richardson extrapolation is the determination of the base matrices. Whichever base matrix is used, extrapolation requires two evaluations of the base matrix--one with the full time step and one with the half time step. Thus, Richardson extrapolation requires twice the computer work for computing the base matrix compared to either Taylor or Padé series. We then have the following two equations for the computer work for the base matrix of Richardson extrapolation using Taylor and Padé approximations.

$$\text{CW(base, Taylor)} = (j+1)(k-1) \text{ mop}$$

$$\text{CW(base, Padé)} = (j+1)(p+1) \text{ mop}$$

Clearly, here is an example of how an increase in computer work gains an increase in the accuracy of the approximation. Again, the issue is whether the increase in accuracy is worth the additional computer work, or if it would be faster to use an alternative method

of integration. The computer work of the extrapolation methods will be further detailed in Sections 7.4 and 7.5.

7.2 Equivalent Work Comparisons--Taylor Series

In this section we will compare the various Taylor series methods and quantify the amount of accuracy obtained for a specified amount of computer work. Examples will be given using the different possible base matrices as well as the different integration methods of stepping, squaring, and combined. The results will be presented in terms of absolute error since the major differences in the equivalent work comparisons are caused by the integration error. The integrations will be run in double precision to minimize the effects of roundoff error. Nevertheless, the roundoff error will be determined to insure the integrity of the results.

The first example is the four-state non-cyclical Markov model that has been used extensively so far. The system parameters are those used previously and are repeated here.

$$a = 10^{-3}/\text{hour}$$
$$b = 10^{-4}/\text{hour}$$
$$t_1 = 100 \text{ hours}$$
$$t_2 = 2\text{x}10^{4} \text{ hours}$$

Again, because of the nature of the matrix exponential, two cases will be examined. The first will have a final time less than the characteristic time, while the second one will have a final time greater than the characteristic time.

Figures 7-1 and 7-2 present the results for the stepping routine, while Figures 7-3 and 7-4 show the results for the squaring routine. In all four cases, it is evident that an increase in the order of the base matrix yields a significant increase in the accuracy for an equivalent amount of work. These results substantiate the earlier notion that decreasing the error for the base matrix decreases the amount of error that will be propagated in time. These results also show that it is usually cheaper to improve the error of the base matrix rather than changing the time step.

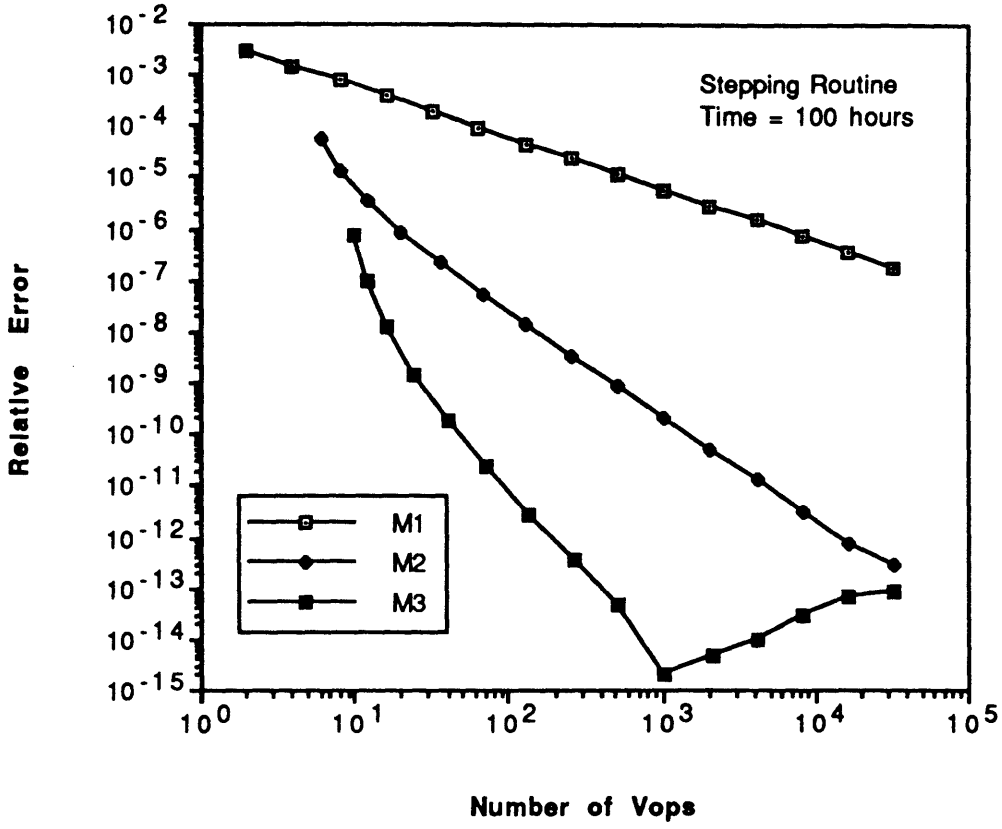


Figure 7-1: Relative Error versus Computer Work for the Taylor Series using the Stepping Routine before the Characteristic Time

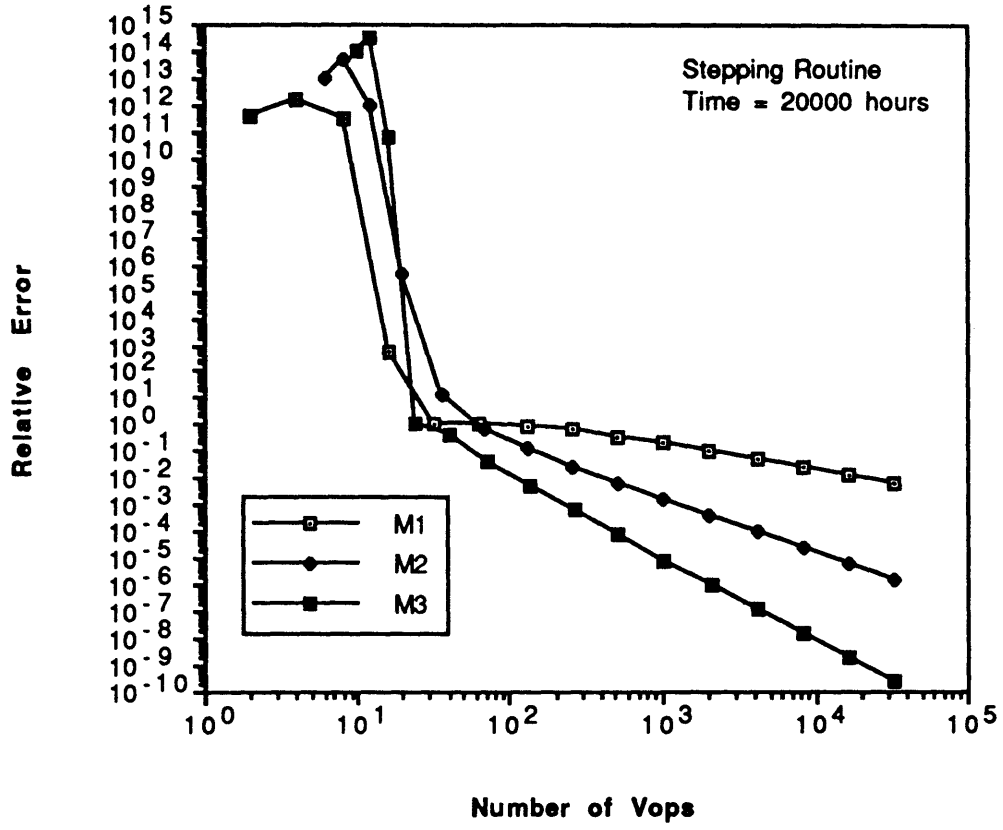


Figure 7-2: Relative Error versus Computer Work for the Taylor Series using the Stepping Routine after the Characteristic Time

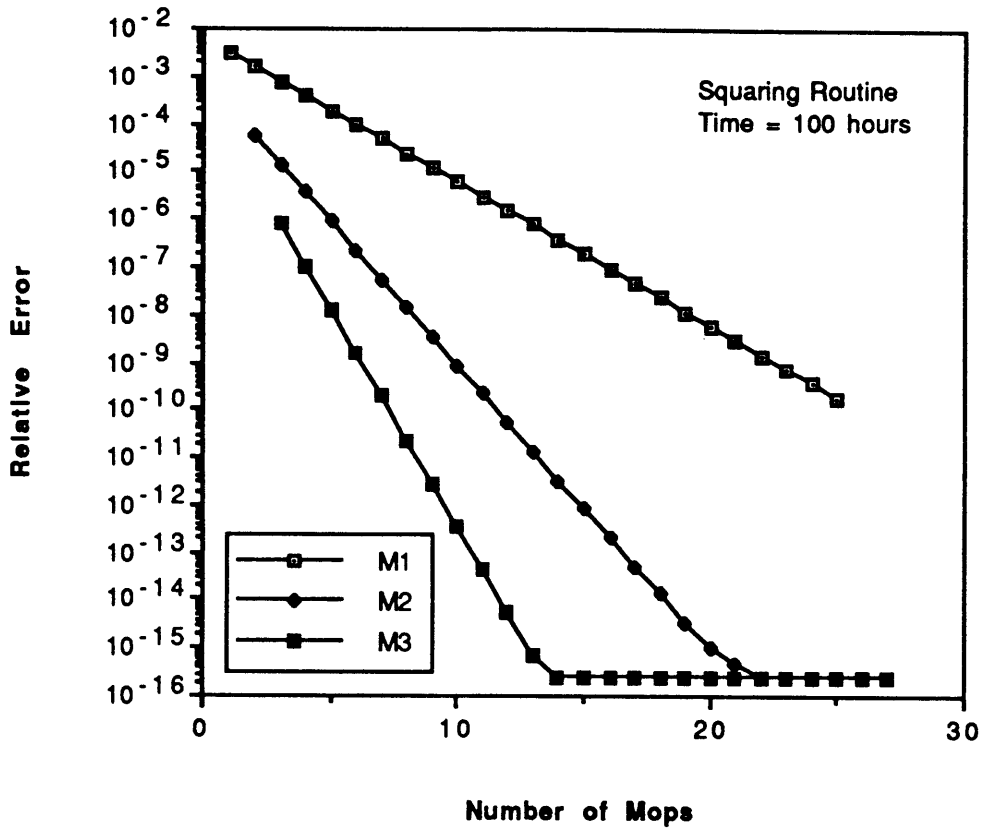


Figure 7-3: Relative Error versus Computer Work for the Taylor Series using the Squaring Routine before the Characteristic Time

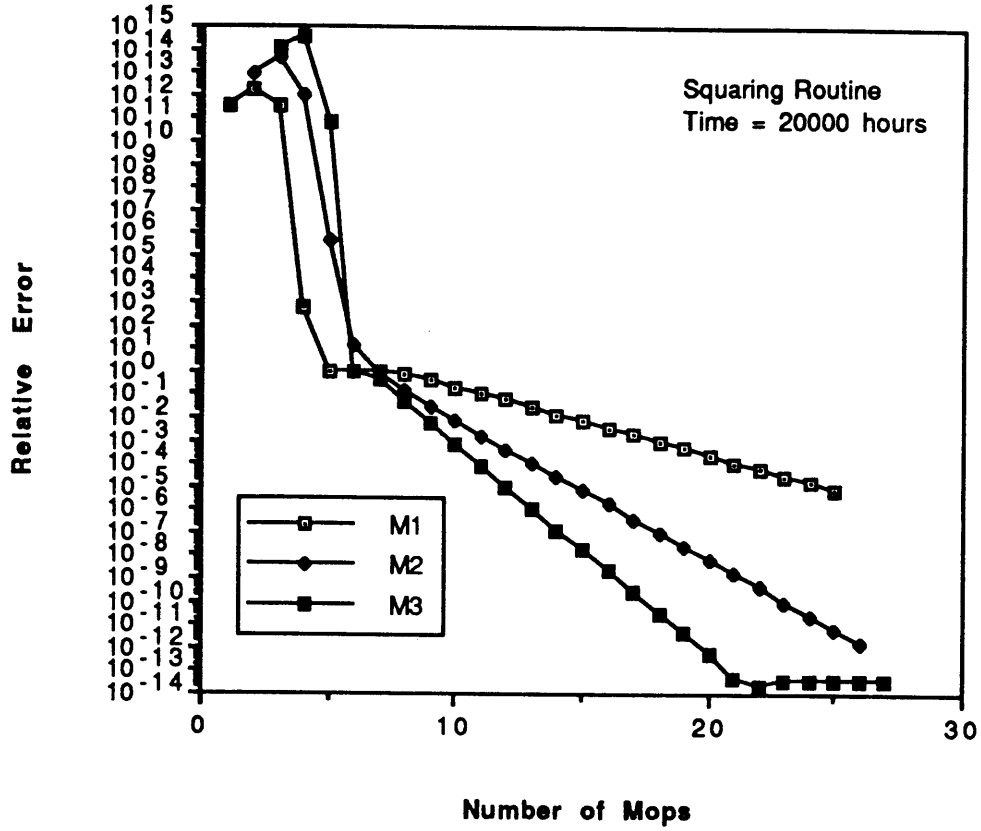


Figure 7-4: Relative Error versus Computer Work for the Taylor Series using the Squaring Routine after the Characteristic Time

To better facilitate the direct comparison of the two methods of stepping and squaring, the relative errors are plotted versus the computer work for the first two base matrices both before and after the characteristic time.

The graphs demonstrate that the squaring routine is generally more accurate than the squaring routine for an equivalent amount of work. However, as was shown in Chapter 3, there is a point before which the stepping is more accurate for an equivalent amount of work. This pattern is evident in all four of the graphs.

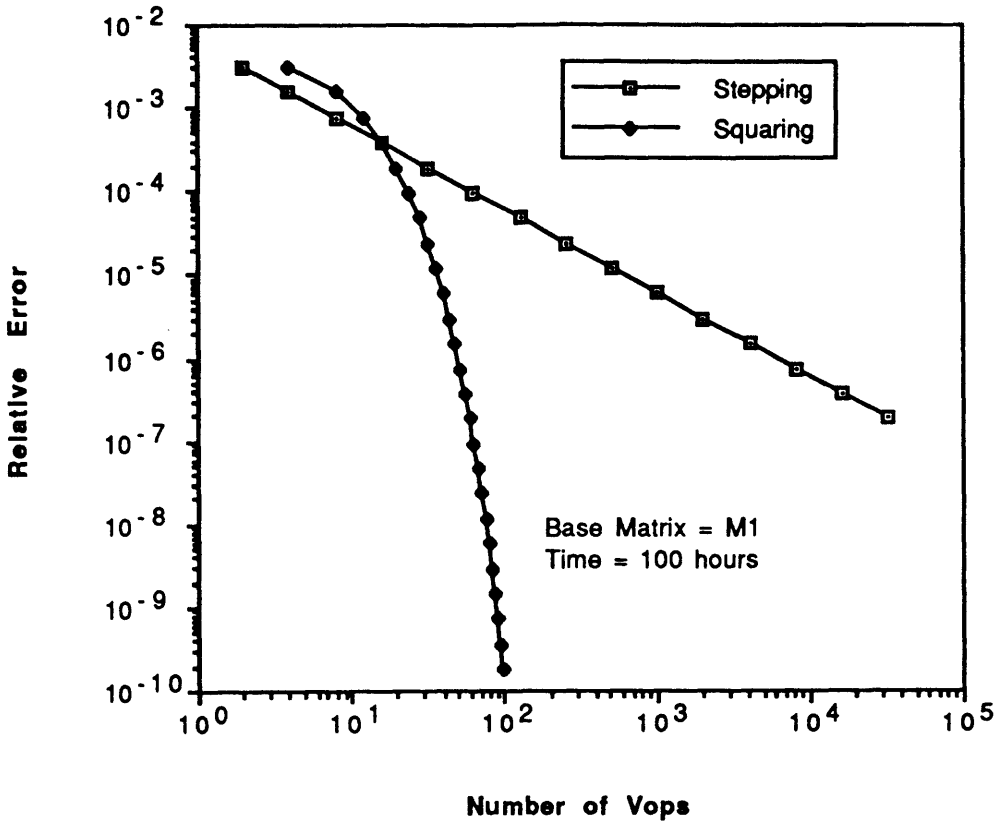


Figure 7-5: Stepping versus Squaring using the First Taylor Series Base Matrix before the Characteristic Time

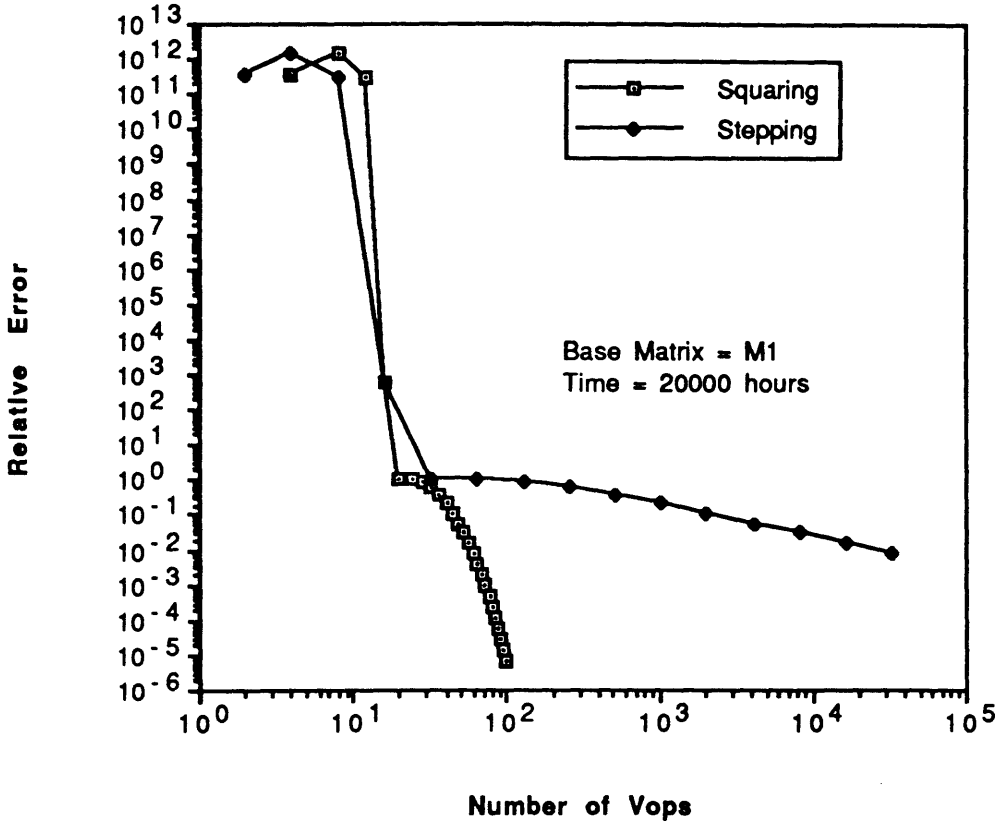


Figure 7-6: Stepping versus Squaring using the First Taylor Series Base Matrix after the Characteristic Time

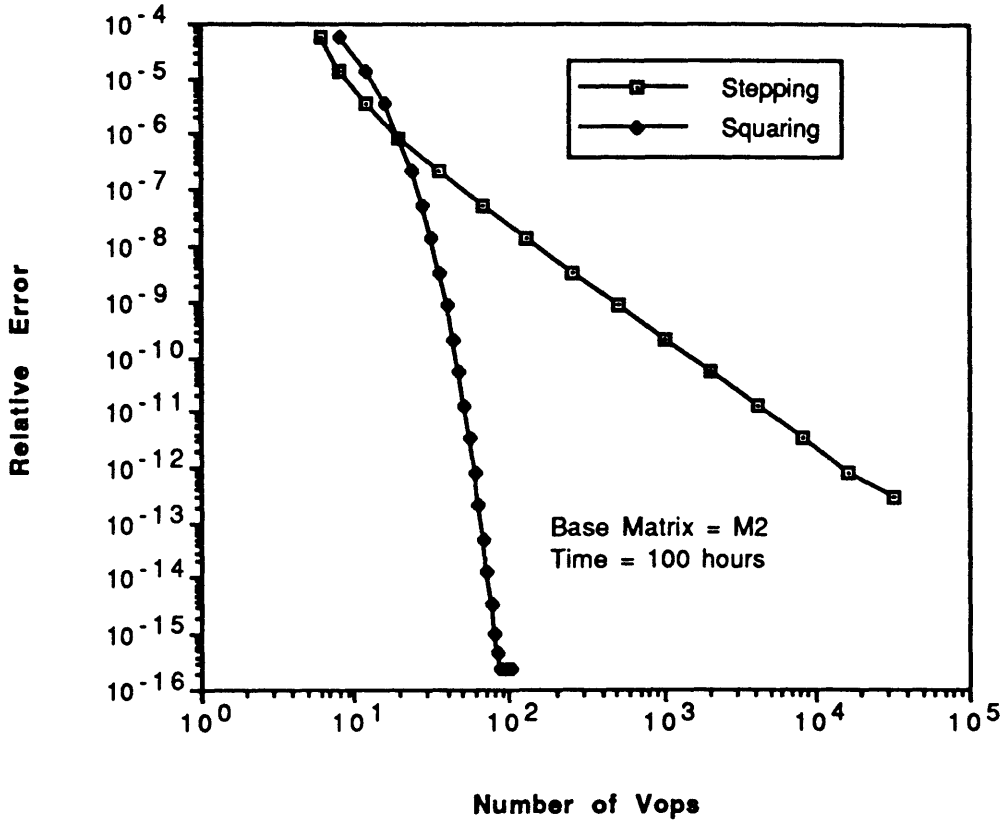


Figure 7-7: Stepping versus Squaring using the Second Taylor Series Base Matrix before the Characteristic Time

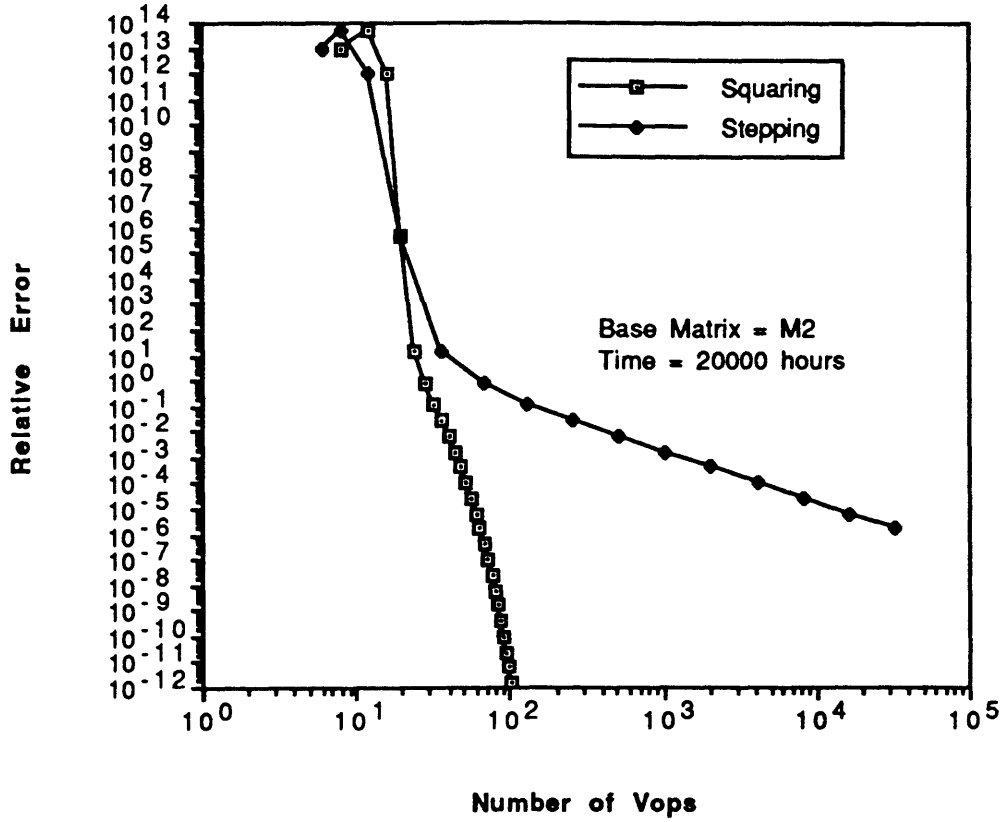


Figure 7-8: Stepping versus Squaring using the Second Taylor Series Base Matrix after the Characteristic Time

7.3 Equivalent Work Comparisons--Padé Series

In this section, results for the four-state non-cyclical Markov model will be given. The system parameters are again

$$a = 10^{-3}/\text{hour}$$
$$b = 10^{-4}/\text{hour}$$
$$t_1 = 100 \text{ hours}$$
$$t_2 = 2\text{x}10^4 \text{ hours}$$

This time, however, the results were generated using the Padé series approximations. Again, the two cases of before and after the characteristic time are exhibited. Additionally, the two integration methods of stepping and squaring are also indicated.

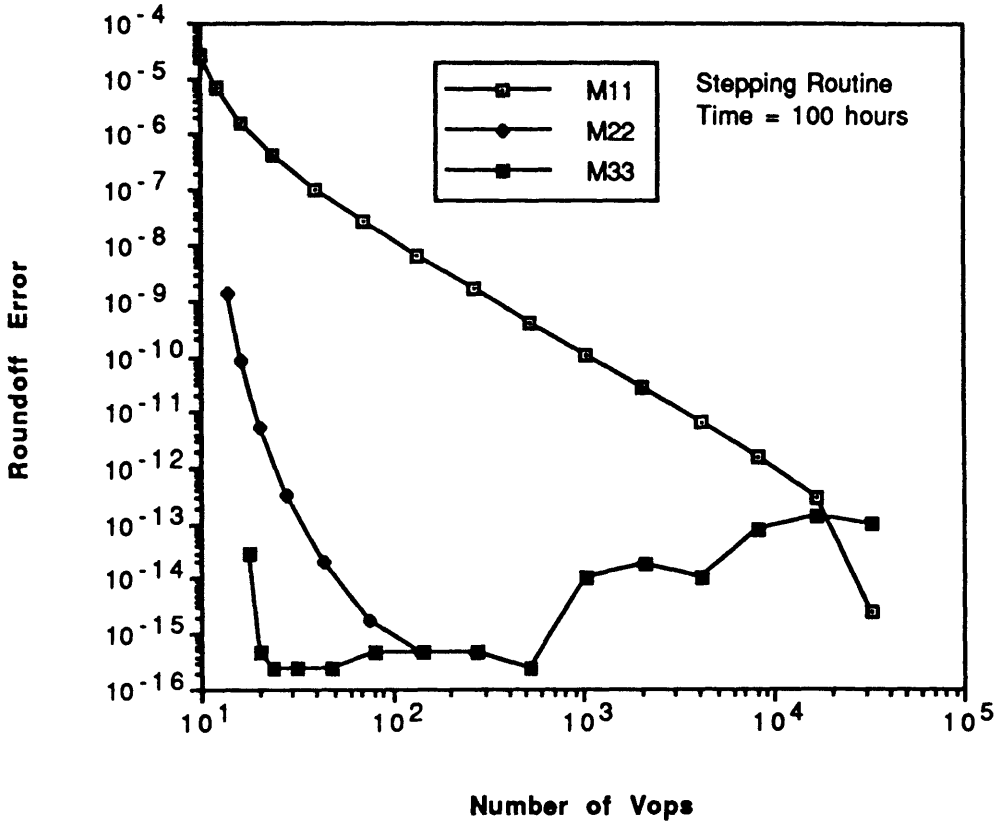


Figure 7-9: Relative Error versus Computer Work for the Padé Series using the Stepping routine before the Characteristic Time

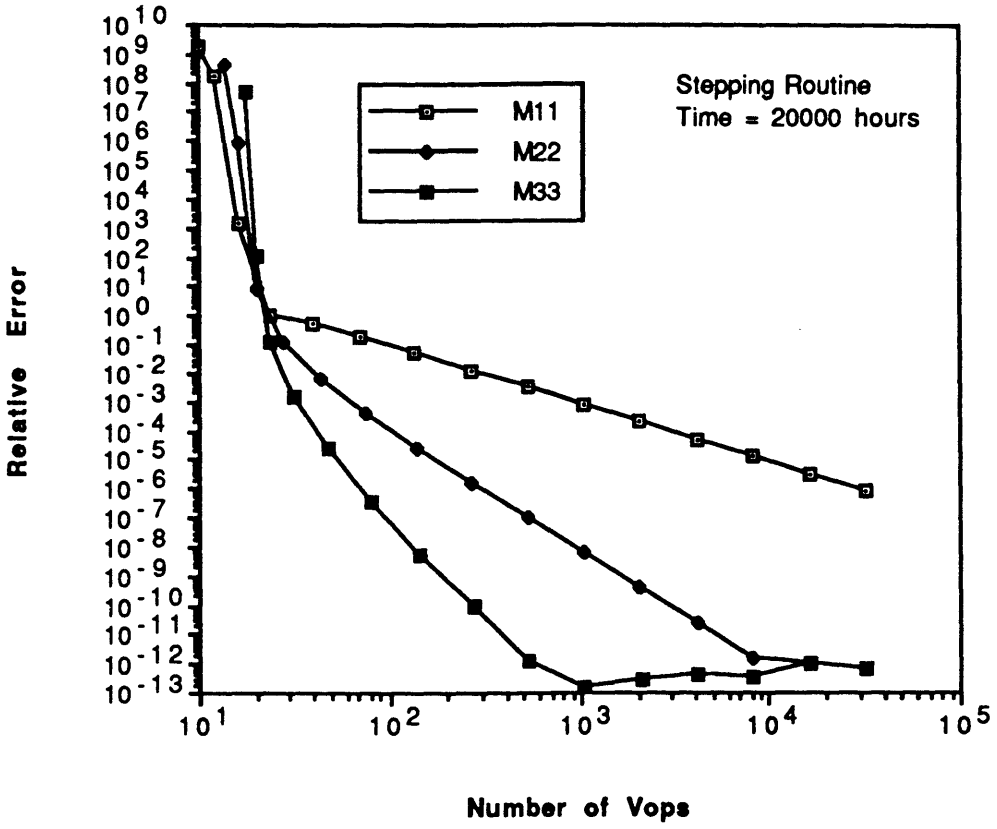


Figure 7-10: Relative Error versus Computer Work for the Padé Series using the Stepping Routine After the Characteristic Time

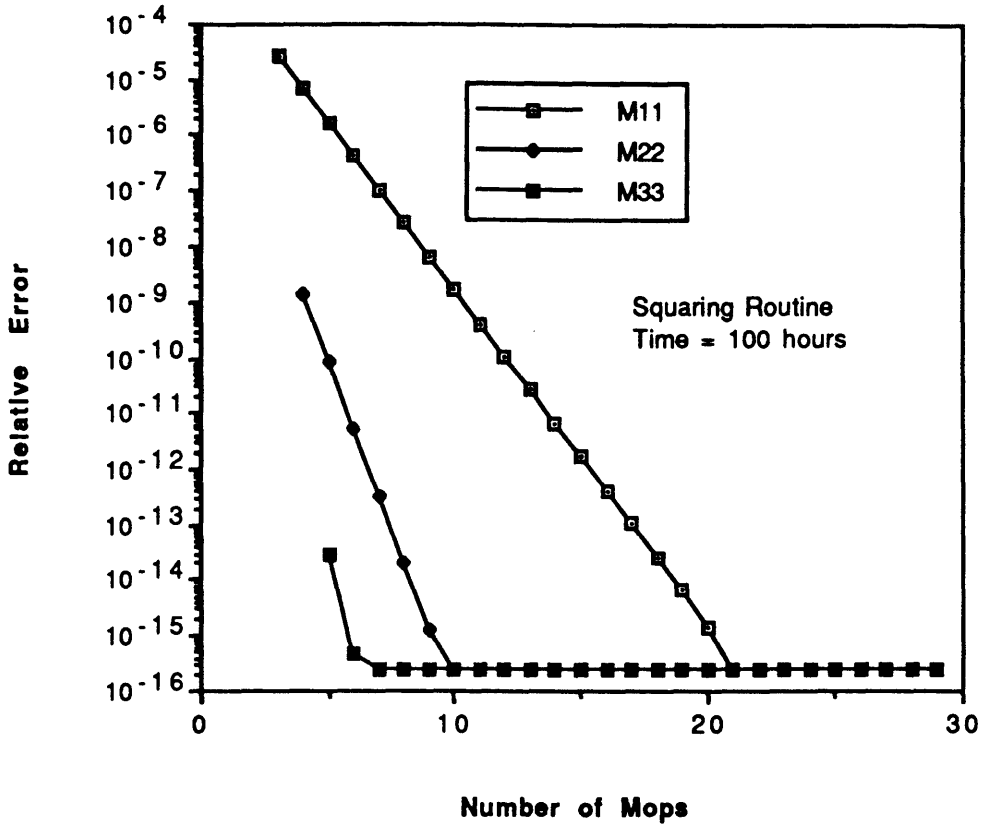


Figure 7-11: Relative Error versus Computer Work for the Padé Series using the Squaring Routine before the Characteristic Time

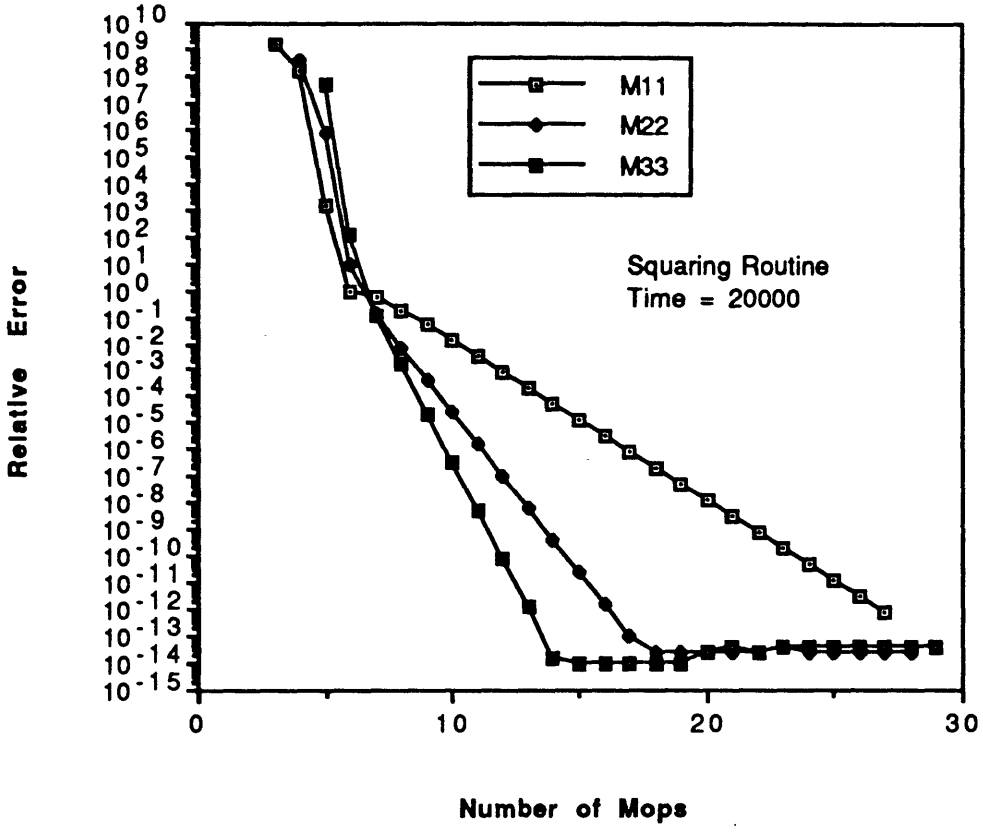


Figure 7-12: Relative Error versus Computer Work for the Taylor Series using the Squaring Routine after the Characteristic Time

In a manner similar to the Taylor Series, the Padé series shows a significant increase in the amount of accuracy for equivalent amounts of computer work by increasing the order of the base matrix. For the Padé series, the relative increase compared to the Taylor series is greater because the Padé series increases the order of the approximation by two for every increase in the base matrix.

Again to facilitate comparison between the two integration methods, the results for the Padé series approach for both stepping and squaring are presented on the same graph in the two examples below. The pattern the was evident for Taylor series is again evident here. The squaring algorithm yields more accuracy for the equivalent computer work in almost all cases.

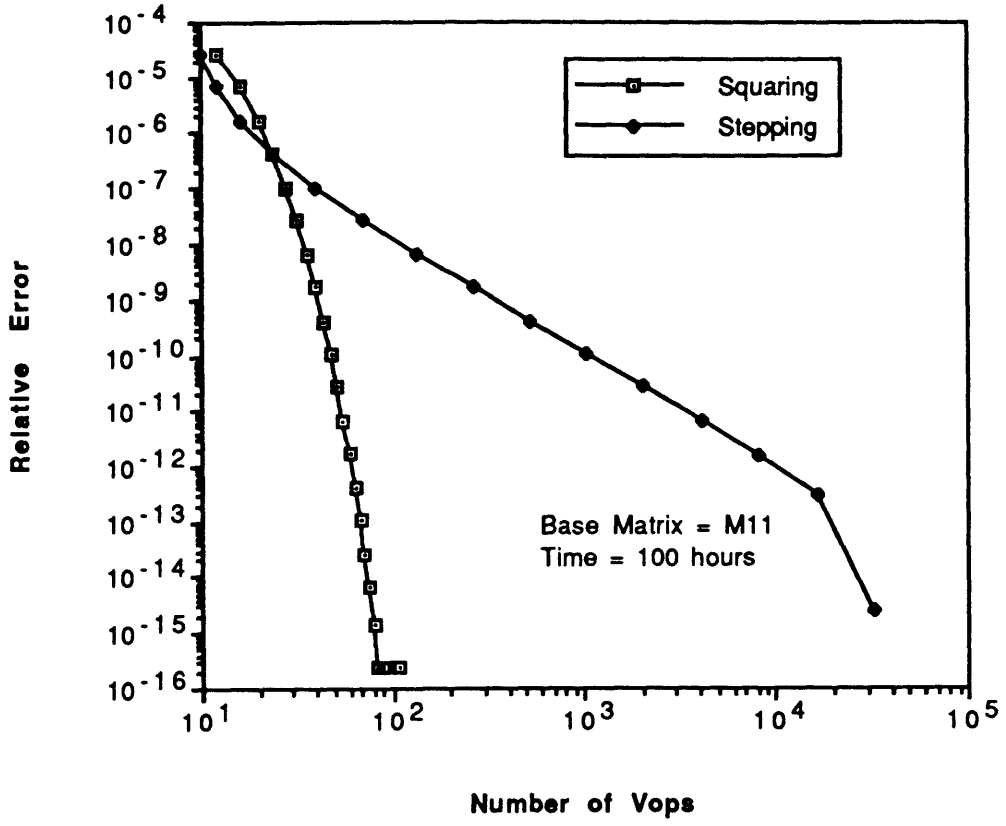


Figure 7-13: Stepping versus Squaring for the First Padé Base Matrix before the Characteristic Time

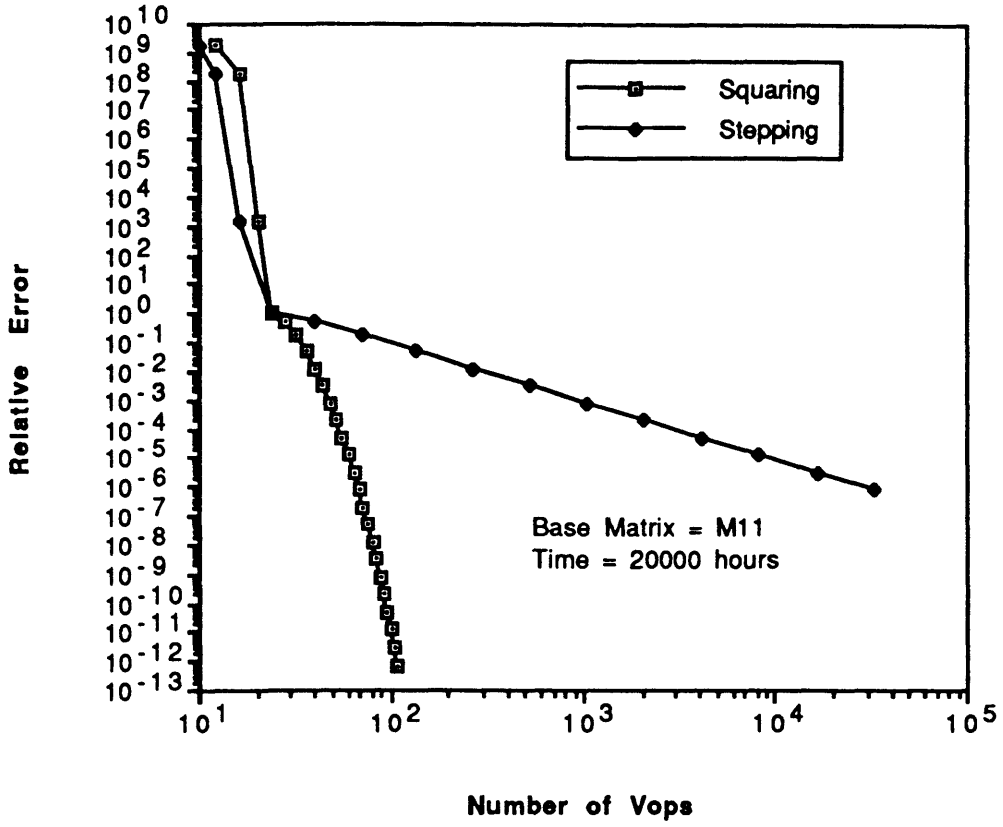


Figure 7-14: Stepping versus Squaring for the First Padé Base Matrix after the Characteristic Time

7.4 Equivalent Work Comparisons--Richardson Extrapolations

It was stated earlier that Richardson extrapolation uses two integration passes of another integration method. In this section, the results of the equivalent work study will be given for Richardson extrapolation using both a Taylor series approximation and a Padé series approximation. The consequences only of squaring will be given. Richardson is very costly using a stepping routine because of the doubling of the computer work. There are two important bases for tradeoffs in the extrapolation methods. The first is the level of extrapolation, and the second is the level of accuracy of the integration passes. The graphs given here will help to clarify these two issues.

The first four graphs (Figures 7-15 to 7-18) are the results using Taylor series approximations. The only time Richardson extrapolation gives an increase in accuracy for

an equivalent amount of work is when the first base matrix is used before the characteristic time (Figure 7-15). In the rest of the incidences, the extrapolation method cost more computer work than simply decreasing the time step. This pattern is especially true after the characteristic time.

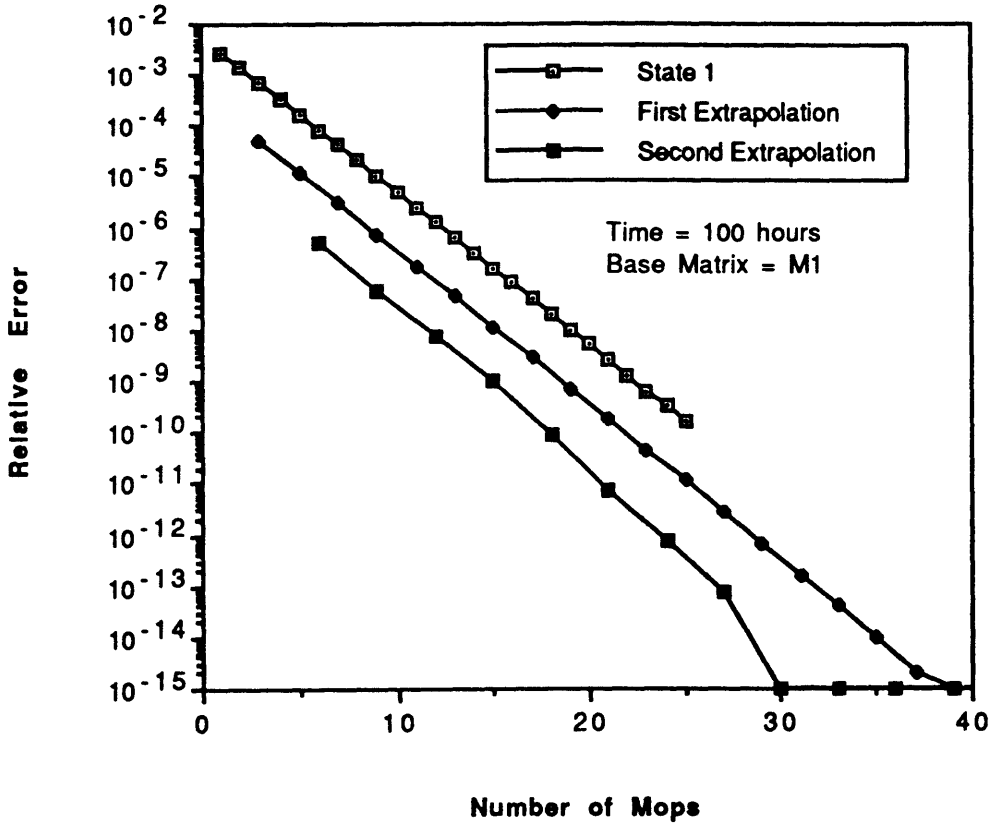


Figure 7-15: Relative Error versus Computer Work using Richardson Extrapolation with the First Taylor Base Matrix before the Characteristic Time

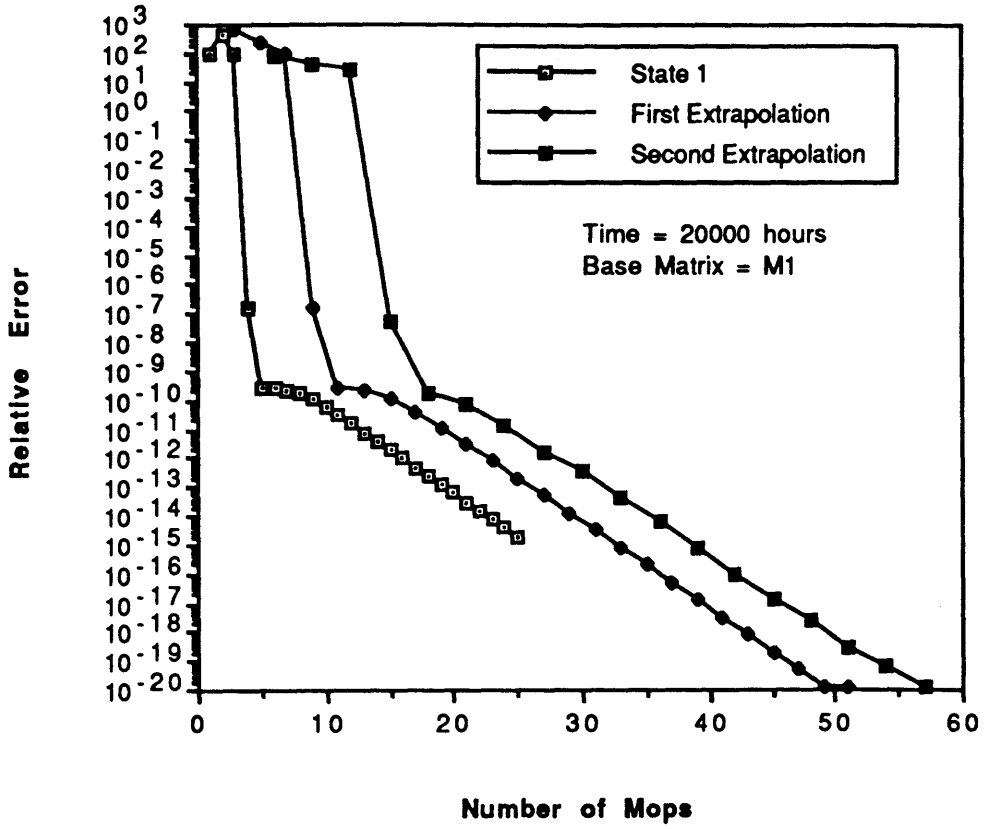


Figure 7-16: Relative Error versus Computer Work using Richardson Extrapolation with the First Taylor Base Matrix after the Characteristic Time

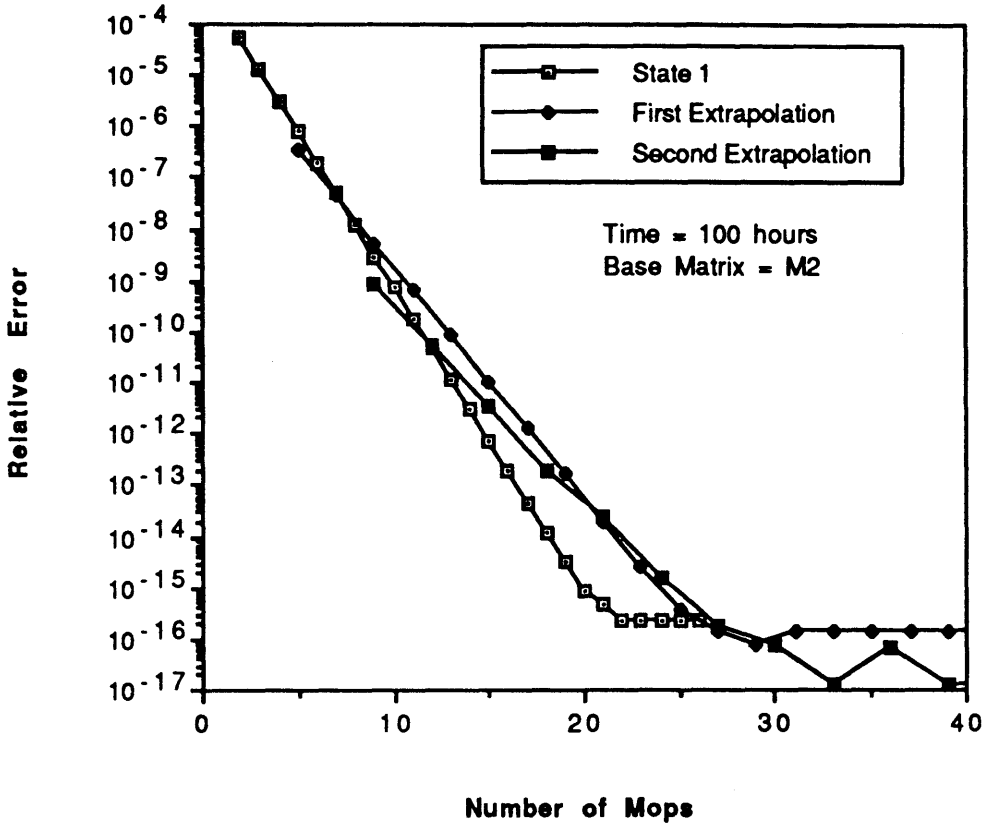


Figure 7-17: Relative Error versus Computer Work using Richardson Extrapolation with the Second Taylor Base Matrix before the Characteristic Time

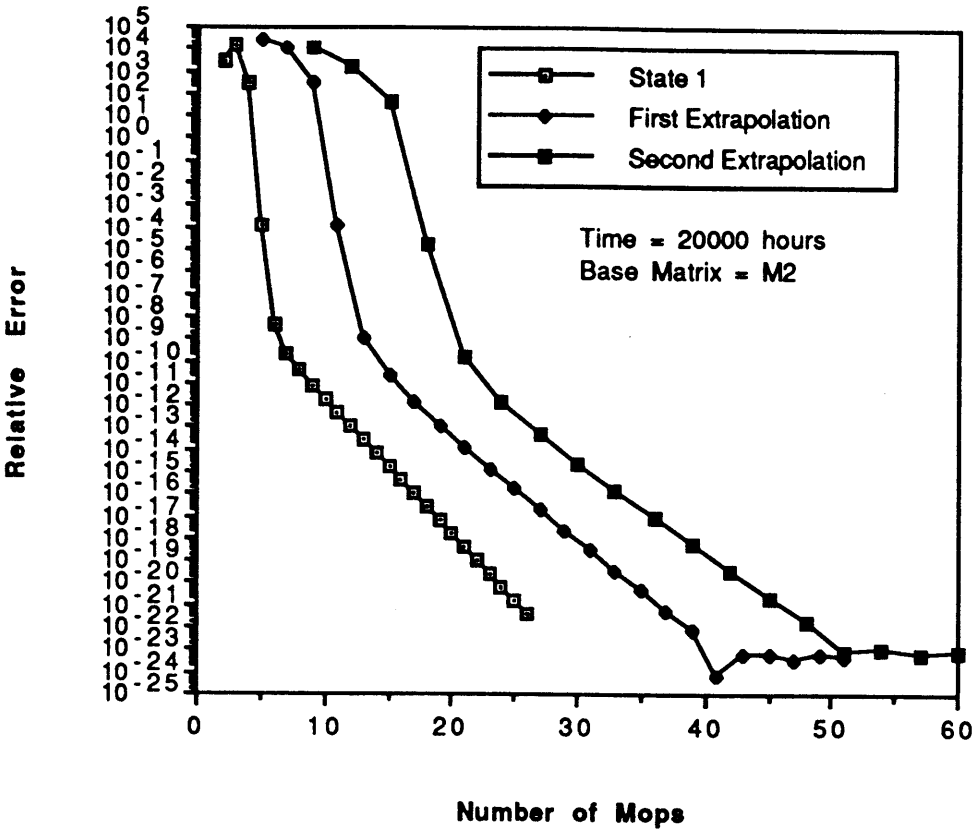


Figure 7-18: Relative Error versus Computer Work using Richardson Extrapolation with the Second Taylor Base Matrix after the Characteristic Time

To complement the results using Taylor series, the results of using Richardson Extrapolation with Padé series are given below. Again Richardson appears to be of advantage in terms of computer work only in the first case with the first base matrix before the characteristic time. In all other cases, equivalent accuracy with less computer work can be achieved by increasing the order of the base matrix or decreasing the time step.

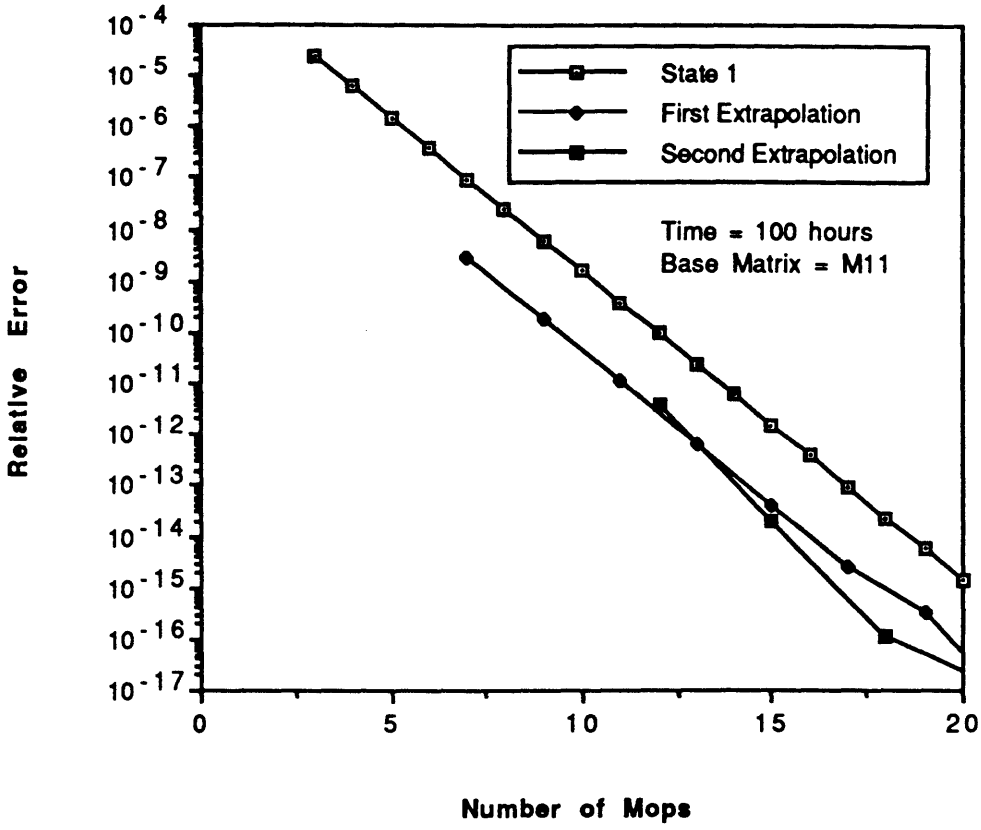


Figure 7-19: Relative Error versus Computer Work using Richardson Extrapolation with the First Padé Base Matrix before the Characteristic Time

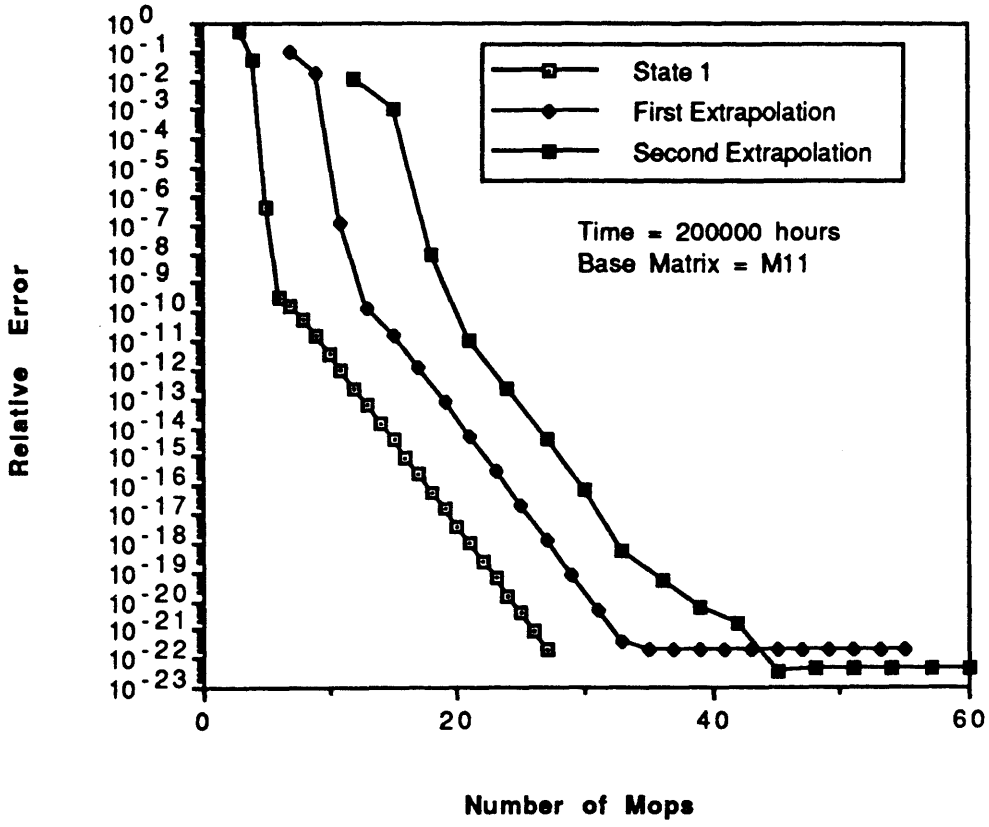


Figure 7-20: Relative Error versus Computer Work using Richardson Extrapolation with the First Padé Base Matrix after the Characteristic Time

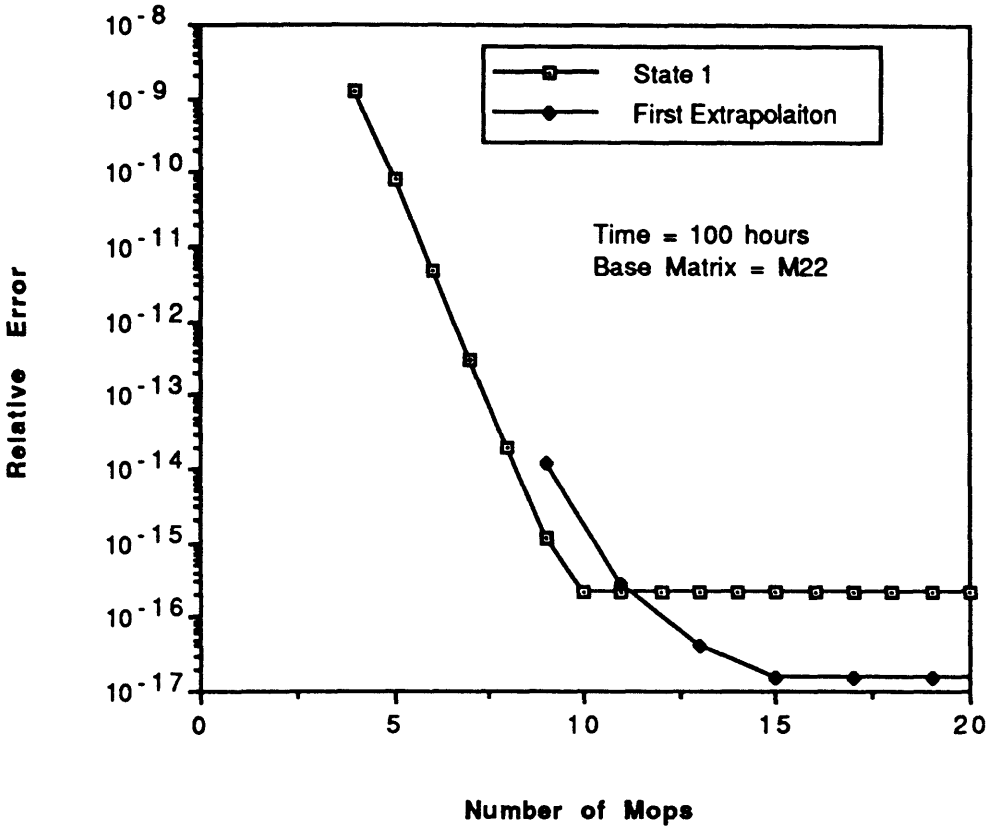


Figure 7-21: Relative Error versus Computer Work using Richardson Extrapolation with the Second Padé Base Matrix before the Characteristic Time

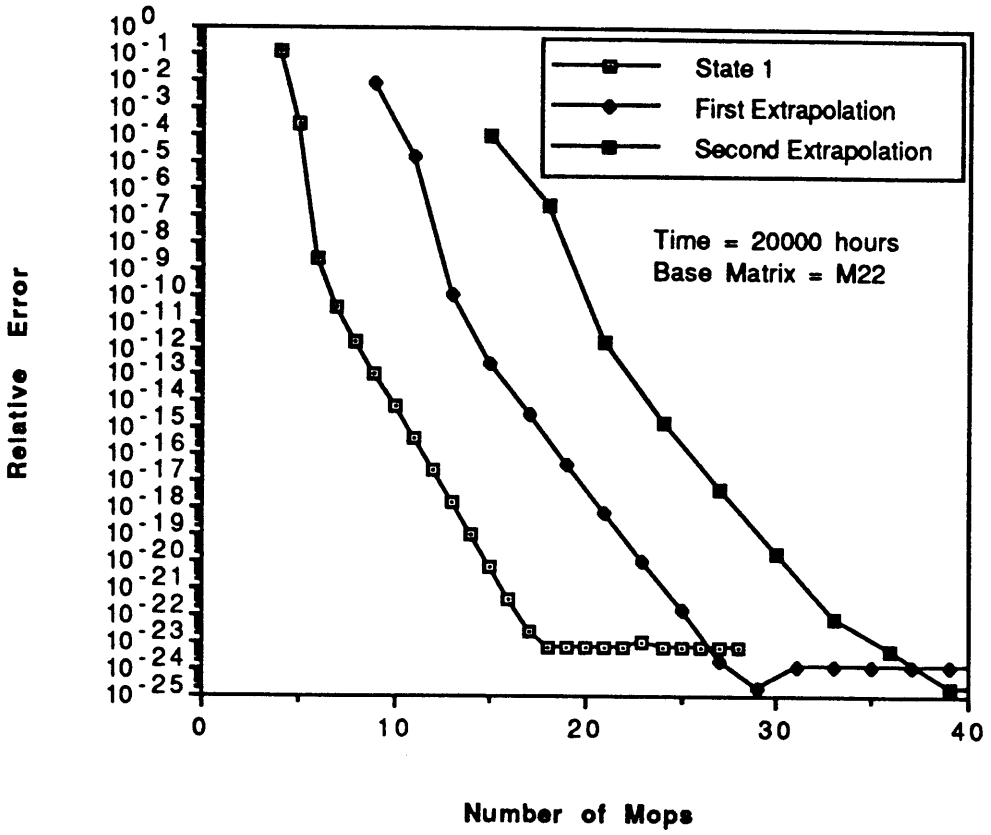


Figure 7-22: Relative Error versus Computer Work using Richardson Extrapolation with the Second Padé Base Matrix after the Characteristic Time

7.5 Equivalent Work Comparisons--Various Methods

So far we have discussed the computer work versus the accuracy obtained for different methods of integration. All the comparisons have been done within a certain integration method. In this section we will compare the accuracy versus the equivalent work for many of the methods described above. We will now compare Taylor directly with Padé. Richardson is not included because it was shown to be effective only for a limited number of cases. In this way we will better be able to choose determine a basis by which to choose the proper integration method.

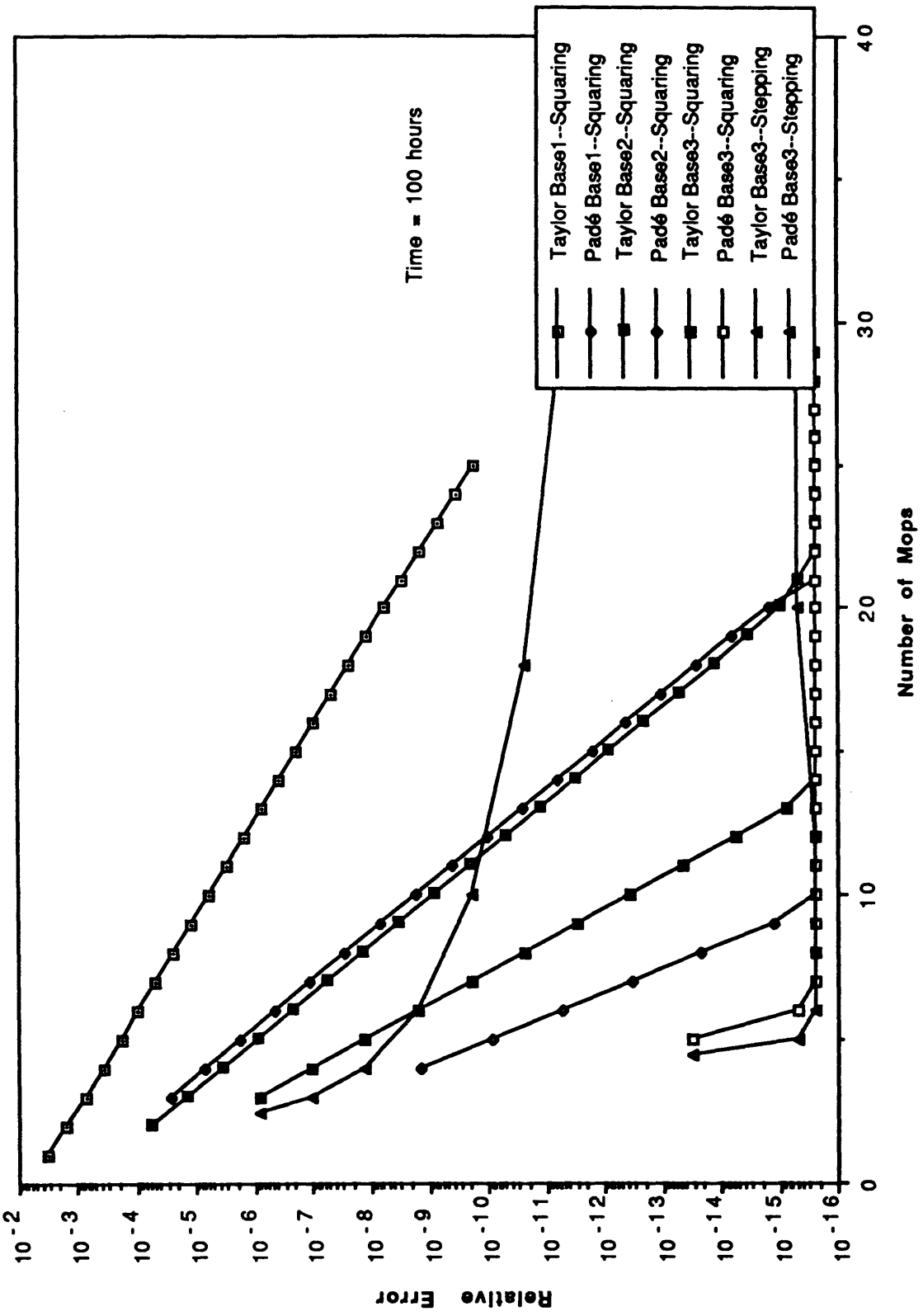


Figure 7-23: Relative Error versus computer Work--A Comparison of Techniques for Before the Characteristic Time

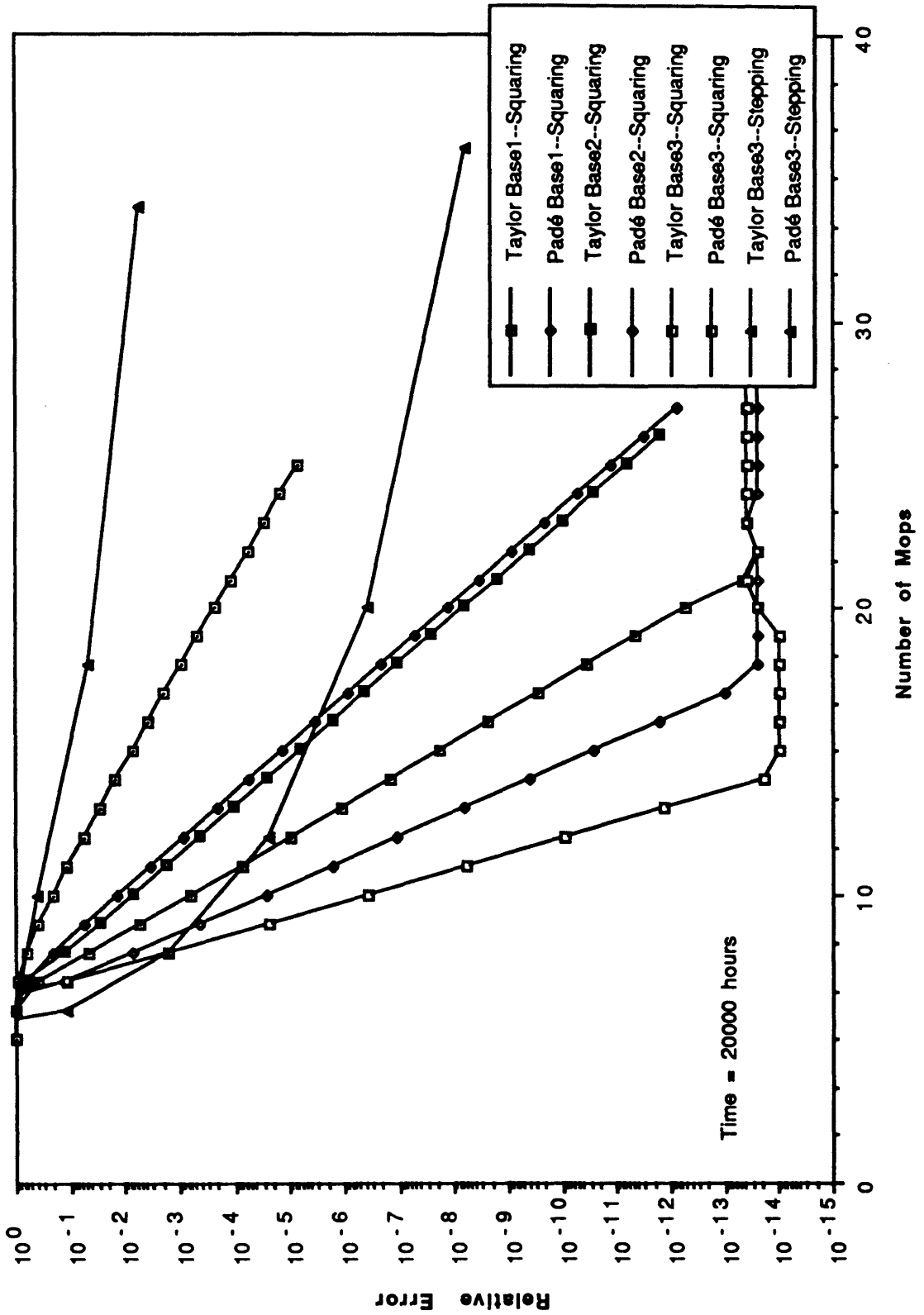


Figure 7-24: Relative Error versus computer Work--A Comparison of Techniques for After the Characteristic Time

Given in the two graphs above are the same results that were given previously, but in a form for direct comparison. Examples from models both before the characteristic time and after the characteristic time are shown.

It is clear from these two graphs that increasing the order of the approximation is generally the most effective means to reduce the numerical error while minimizing the computer work. Increasing the order of the approximation can be done with either Taylor or Padé series approximations. Note that the two stepping routines are quite efficient for the system whose final time is less than the characteristic time of the system (Figure 7-23). This is primarily due to the fact that the example only has four states. Comparing computer work for stepping and squaring algorithms is model specific because of the dependence on the dimension of the matrix. For this case, the stepping routines are quite useful for a final time less than the characteristic time.

7.6 Recapitulation

This chapter was designed to examine the numerical error in the determination of the matrix exponential in terms of the parameter of primary interest, computer work. This information is necessary before one can determine the solution technique that yields the requested accuracy in the least amount of time.

This chapter began with a discussion of computer work. The equations of computer work for the Taylor series approximations, for Padé series approximations, for accumulator methods, and for Richardson extrapolations were developed. This study also encompassed the work required for the techniques of stepping, squaring, and a combination of the two. Two new units of computer work were introduced, the mop and the vop. A mop is the amount of computer work required to multiply a matrix by another matrix. The vop is the amount of work required by multiply a matrix by a vector. Both are useful in presenting equivalent work data, since they remove the dependence on the matrix dimension.

Simply describing the computer work necessary for a certain technique, however, does not fully explain the benefits of the various methods. For this reason, equivalent work studies were done for a variety of examples. The examples included each of the major integration techniques discussed, as well as the two cases of before and after the characteristic time. The methods were first compared individually to see the effects of the number of terms in the base matrix and the effects of the time step. Additionally, the

outcomes of the integration using stepping was compared to one using squaring and one using a combination. After the individual comparisons, the equivalent work was compared across the integration routines. In this manner, Taylor approximations could be directly compared to Richardson extrapolation using Padé, and Padé approximations could be directly compared to a Richardson extrapolation using Taylor.

This chapter facilitated the direct comparison of numerical error based on the amount of computer work. Now, the problem of *a priori* choosing the proper integration method and integration parameters is left. The solution of this problem is the topic of Chapter 8.

Chapter 8

The Automatic Selection of the Integration Parameters

To this point, we have detailed the various error propagation patterns for both integration and roundoff errors. It was shown that, in general, the resulting error was dependent on the two integration parameters--the time step and the number of terms in the truncated power series. Equations that bound the numerical error were developed in terms of these integration parameters. Overall, these equations give a means to judge the various algorithms based on accuracy.

Chapter 7, in contrast to the other chapters, introduced a new basis for comparison of the integration schemes--the amount of computer work necessary to achieve a certain accuracy. Now we can determine which method is more desirable if both yield the same accuracy. Also, we can determine which improvements to a technique give acceptable results in the least expensive manner. Essentially, we can now find the most efficient algorithm (in terms of computer work) that still yields the proper accuracy.

This chapter establishes the methodology for determining the integration procedure and parameters that produce numerical results with the designated accuracy in the least amount of computer work. The technique developed is general enough for either power series solution technique. This chapter first describes the input that is necessary for the algorithm. Once the needed input is explained, the chapter discusses the work-error equations and the means to minimize them.

8.1 Necessary Input

The integration parameters to be determined automatically are the time step and the number of terms in the truncated power series. To supplement these integration parameters, it is requisite that certain system parameters be specified. These parameters are part of the work-error equations and must be supplied by the user.

The most obvious system parameter to be specified is the transition matrix, A. This matrix has the state transitions that describe the system. In order to obtain an error bound for the final solution, this matrix is operated on to determine its norm. Also, it is necessary to determine the largest and the smallest transition rates for use in chain model approximations. Another system parameter specified by the user is the final time. This

value is needed to determine the time step, and of course, the corresponding number of time steps.

The last necessary parameter is the desired accuracy. There are essentially two options in specifying the desired accuracy. One can specify the required number of significant digits, which translates into a relative error. Alternatively, one can specify that the error can be no greater than a certain value, which translates into an absolute error. Because some knowledge about expected probabilities is usually needed to specify an absolute error, the number of significant digits will typically be specified.

Frequently, a user is only interested in one of the states of the Markov model, say the full-up state for a reliability example. In this case, it is desired only to bound the error for the state of choice. Some of the examples in Chapter 5 showed a difference in accuracy depending on the failure level. Although bounding may be done for some cases using chain model approximations, the more general approach is taken here. Admittedly, the general approach may require more computer work. However, the general approach is simpler, covers a wider range of problems, and will still guarantee the accuracy of the state in question.

In addition to the system parameters, it may be desirable to have the user specify the power series method of approximation (Taylor or Padé). Some systems may require an absolutely conservative bound on the error of the solution. In this case, the Taylor series solution should be specified because of the uncertainty in the matrix inversion process of Padé. On the other hand, if the user is willing to accept the risk that the denominator matrix is well conditioned, the machine can select the power series as well as check the matrix inversion once it has been completed.

Thus, only three system parameters need be specified in order to have a computer automatically choose the integration parameters. The three system parameters are the transition matrix, the final time, and the desired accuracy in terms of significant digits. In addition to the necessary parameters, the user may want to specify the power series to use, or the user may want to specify a state of interest. Using these parameters, the time step and the number of terms in the truncated power series will be selected.

8.2 Work-Error Equations

Up to this point, equations for the amount of roundoff error, the amount of integration error and the amount of necessary computer work have been developed. We

will now combine these equations into a convenient form for determining the integration parameters.

There are different equations for the amount of roundoff error depending on the type of solution technique chosen. The two most common routines are stepping algorithms and squaring algorithms. The method of variable renormalization was given as a means to reduce the roundoff error when using a squaring algorithm. Accumulator methods were also described to reduce the roundoff error for stepping routines. Accumulator methods, however, were shown to be computationally intensive. The results of Chapter 4 are summarized here.

$$\text{RRE (stepping or squaring)} = n\,\varepsilon$$

$$\text{RRE (squaring with variable renormalization)} = \frac{\ln n}{\ln 2}\,\varepsilon$$

$$\text{RRE (stepping with accumulator)} = \begin{cases} \varepsilon & \text{for } n < \dfrac{1}{\varepsilon} \\ \left(n - \dfrac{1}{\varepsilon}\right)\varepsilon & \text{for } n > \dfrac{1}{\varepsilon} \end{cases}$$

The amount of integration error for various integration algorithms was also developed. All of the solution methods were based on the two power series techniques of Taylor and Padé. The integration error was due to the error in the base matrix and the error in the propagating of this base matrix. Additionally, an extrapolating procedure was described as a means to reduce integration error. The results of Chapters 5 and 6 are summarized here.

$$\text{AIE (Taylor series)} \le \frac{t\,(\Delta t)^k}{(k+1)!}\,\|A\|^{k+1}$$

$$\text{AIE (Padé series)} \le \frac{t\,(\Delta t)^{2p}}{(2p+1)!}\|A\|^{2p+1} + n\|E_i\|$$

$$\text{AIE}_{j+1}\text{ (Richardson with Taylor)} \le \frac{(EP_j^T) \bullet AIE'_j) - AIE_j}{EP_j^T - 1}$$

$$\text{AIE}_{j+1}\text{ (Richardson with Padé)} \le \frac{[EP_j^T \bullet AIE'_j] - AIE_j}{EP_j^T - 1}$$

where the superscript T and P in the final two equations represents the extrapolation parameters for the Taylor and Padé series, respectively. The additional roundoff error

incurred in using the Richardson extrapolation is usually negligible compared to the other errors involved. Nonetheless, the roundoff error is easily computed using the equations for addition and multiplication derived in Chapter 4.

Lastly, the third set of equations developed so far has been the computer work equations. These equations are based on the amount of computer work required using the various techniques in terms of flops, vops and mops. They are dependent on the time step and the number of terms in the truncated power series, as well as the method of solution. The results of Chapters 3 and 7 are summarized below.

$$\text{CW (stepping with Taylor)} = (k\text{-}1)\ \text{mops} + n\ \text{vops}$$

$$\text{CW (stepping with Padé)} = (p+1)\ \text{mops} + n\ \text{vops}$$

$$\text{CW (squaring with Taylor)} = \left((k\text{-}1) + \frac{\ln n}{\ln 2}\right)\ \text{mops}$$

$$\text{CW (squaring with Taylor)} = \left((p+1) + \frac{\ln n}{\ln 2}\right)\ \text{mops}$$

$$\text{CW (stepping with Taylor, Richardson)} = (j+1)n\ \text{vops} + (j+1)(k\text{-}1)\ \text{mops}$$

$$\text{CW (stepping with Padé, Richardson)} = (j+1)n\ \text{vops} + (j+1)(p+1)\ \text{mops}$$

$$\text{CW (squaring with Taylor, Richardson)} = \left[(j+1)\left(\frac{\ln n}{\ln 2}\right) + j + (j+1)(k\text{-}1)\right]\ \text{mops}$$

$$\text{CW (squaring with Padé, Richardson)} = \left[(j+1)\left(\frac{\ln n}{\ln 2}\right) + j + (j+1)(p+1)\right]\ \text{mops}$$

$$\text{CW (stepping with Taylor, Accumulator)} = (6k\text{-}2)n\ \text{vops}$$

Note that the additional computer work required with the variable renormalization method is one additional vop for each squaring of the matrix (one mop). Therefore, this additional work is considered negligible, and the computer work for a variable renormalization is taken to be equivalent to the computer work given for the squaring methods above.

We now have a comprehensive list of the three basic equations necessary to properly judge a technique for the solution of the matrix exponential subject to the constraints of the Markov model. The equations can be combined in the proper fashion to yield the work-error equations for the solution technique used. These equations can then be solved for the time step, in terms of the system parameters and the number of terms in

the truncated power series. The solution of these equations to meet the desired accuracy requirement is the subject of the following section.

8.3 Minimizing the Work Equations Subject to the Error Constraint

The goal of this research has been to find an equation or a set of equations by which the computer would automatically select the integration parameters and integration technique to solve the matrix exponential for a Markov model. The integration parameters selected should guarantee that the numerical error incurred is less than a given, desired amount. The equations in the lists above represent the means to select the integration parameters. It is required now to solve these equations to insure the desired accuracy, but it is also preferred to minimize the computer work associated with the integration technique. In this section an approach to select the integration parameters that yield the minimum computer work will be offered.

To demonstrate the approach we will use the most generally applicable and beneficial technique--the squaring algorithm using Taylor series with variable renormalization. This approach can be easily applied to the other techniques as well. We initially assume that the roundoff error is negligible. If desired, the roundoff error can be adjoined to the integration error. However, this is not usually necessary. Thus, we start with the integration error equation written in terms of relative error with the specified relative error (SPE) supplied by the user.

$$\text{SRE} \leq \left(\frac{t\,(\Delta t)^k \|A\|^{k+1}}{(k+1)!\ \text{factor}} \right) .$$

We then solve this equation for the time step, Δt.

$$\Delta t = \left\{ \frac{(\text{SRE})\,(k+1)!\ \text{factor}}{t\,\|A\|^{k+1}} \right\}^{\frac{1}{k}}$$

The associated computer work equation for scaling and squaring is

$$\text{CW} = \left((k-1) + \frac{\ln n}{\ln 2} \right) \text{mops}$$

For completeness, the roundoff error for the squaring with variable renormalization algorithm is, with n' indicating the point at which the characteristic time has been reached

$$\text{RRE (squaring with variable renormalization)} = \begin{cases} \dfrac{\ln n}{\ln 2}\,\varepsilon & \text{for } n\Delta t < \text{char time} \\ (n - n')\left(\dfrac{\ln n'}{\ln 2}\right)\varepsilon & \text{for } n\Delta t > \text{char time} \end{cases}$$

The goal is to minimize the computer work equation while still satisfying the relative error constraint. Such a problem would seem to imply Lagrange multipliers or other similar method. Unfortunately, the parameter k is integer valued and in a factorial, thus preventing any differentiation. Therefore, we are left with direct calculation methods.

The solution that has proved to be simple and effective is to iterate over k. First, select a value for k and determine the time step Δt that satisfies the error constraint. Next, increase the value for k by one and again determine the appropriate time step. Once the $(k, \Delta t)$ pairs have been determined, select the one that requires the least amount of computer work. Note that there may be more than one pair that satisfies the error constraint in the least amount of computer work. Because the error constraint is a one-sided constraint, some parameter choices may yield a much better accuracy than desired in the same amount of computer work.

Now that the $(k, \Delta t)$ pair of least work has been chosen, the assumption of neglecting the roundoff error should be checked. If the error constraint is still satisfied, the chosen integration parameters should be used. If the error constraint is not satisfied, the relative error requested should be adjusted to include the roundoff error of the first iteration.

A typical minimization search method for the example considered many times so far in this report would yield the following pairs of $(k, \Delta t)$ and computer work. The computer work is plotted against the number of terms in the base matrix. The equivalent results $(p, \Delta t)$ are given for the Padé series approach, also using the squaring algorithm with variable renormalization. Here one can see that two choices of integration parameters may satisfy the error constraint and still require the least amount of computer work.

The flow chart following the graphs is meant to summarize the minimization process for one of the integration algorithms given here.

$a = 10^{-3}$ /hour

$b = 10^{-4}$ /hour

$t_1 = 100$ hours

$t_2 = 20000$ hours

$SRE = 10^{-5}$

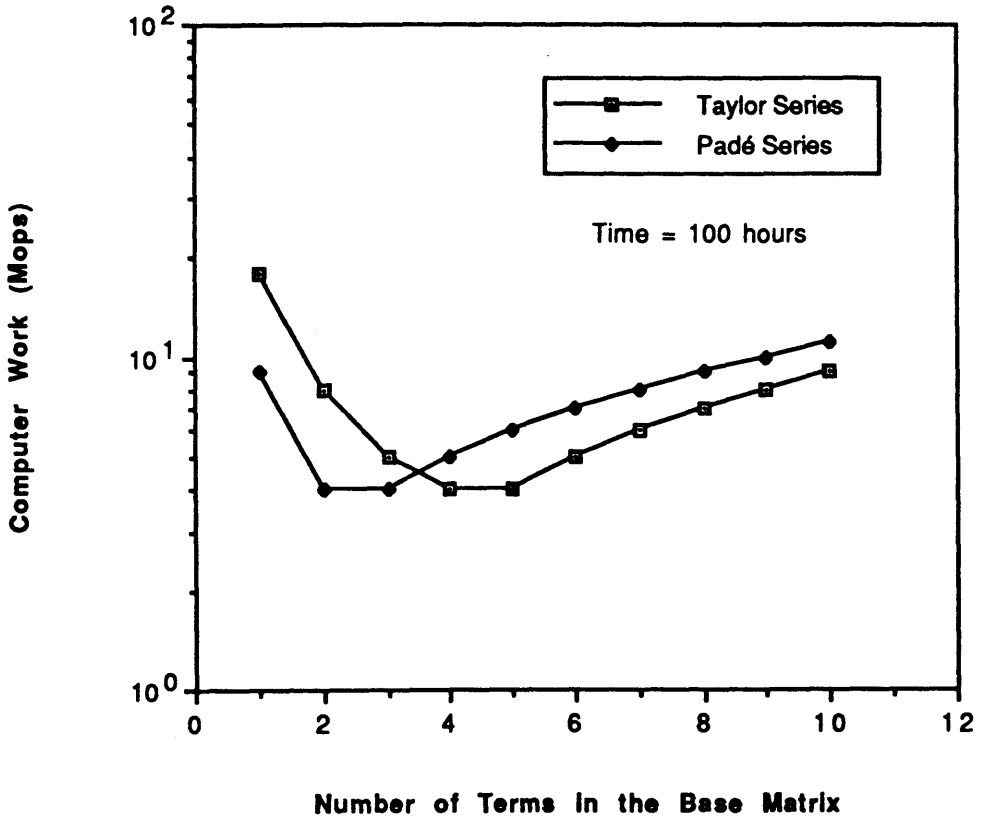


Figure 8-1: Computer Work versus the Number of Terms in the Base Matrix for a Specified Relative Error Before the Characteristic Time

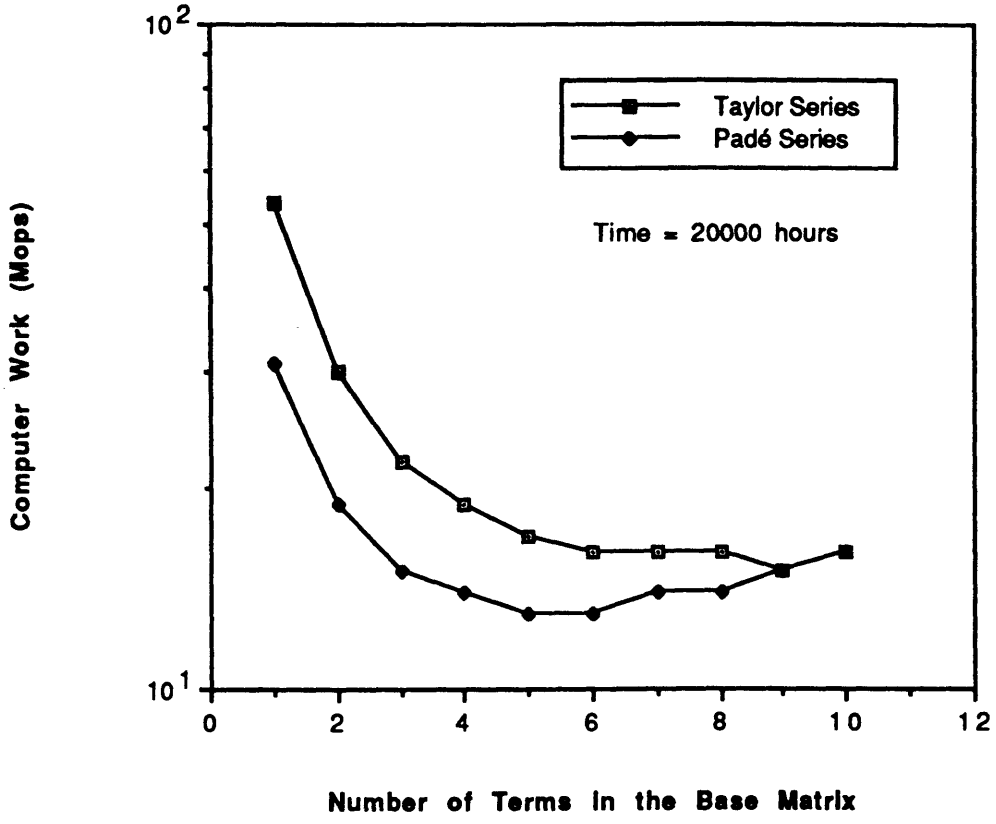


Figure 8-2: Computer Work versus the Number of Terms in the Base Matrix for a Specified Relative Error After the Characteristic Time

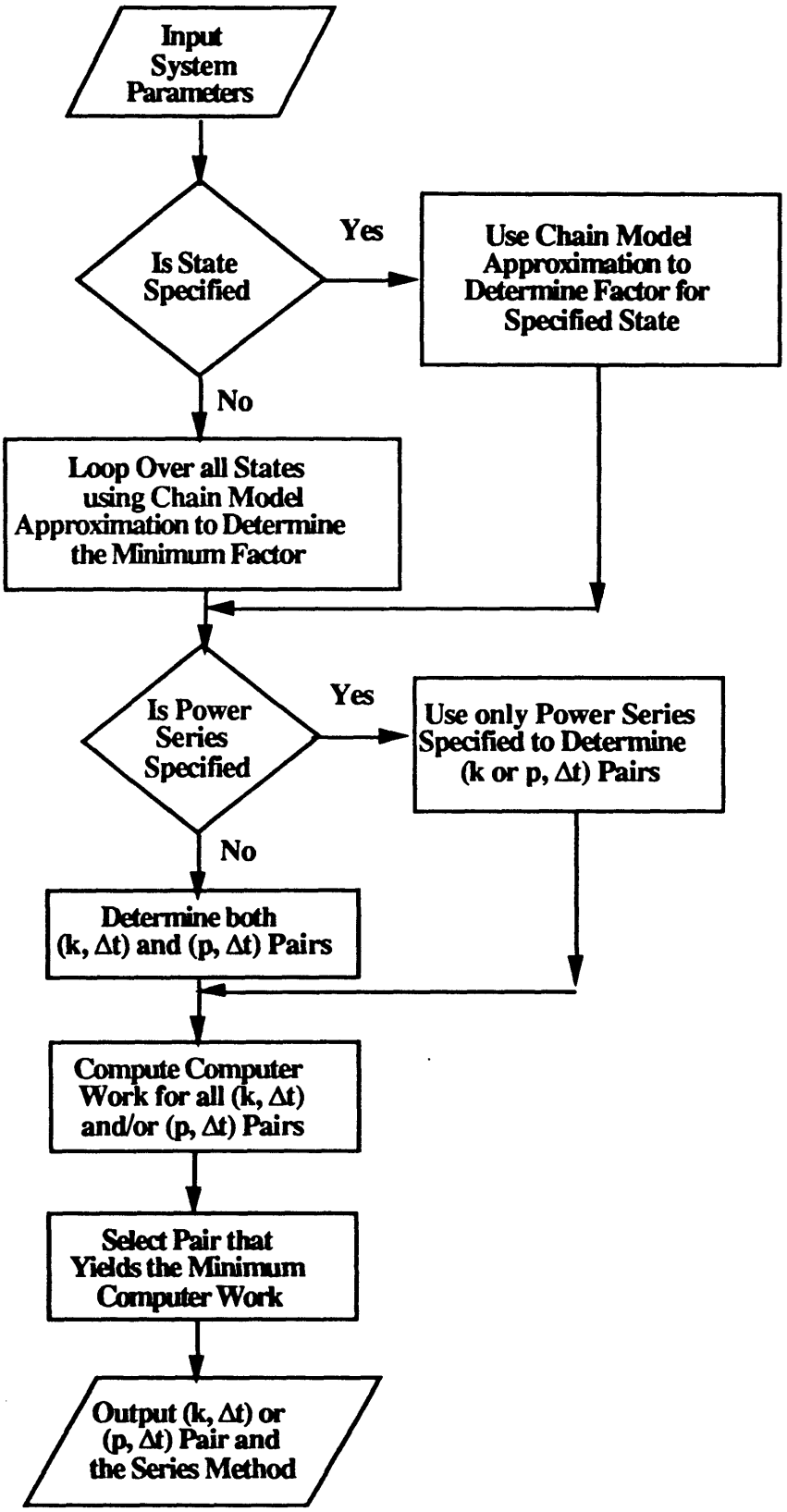


Figure 8-3: Flow Chart for Integration Parameter Selection

The minimization algorithm can be applied to the other integration methods as well. The results of these minimizations can then be compared to determine the integration method and the integration parameters. However, it may be more effective to chose a solution technique that fits the needs of the majority of the problems faced. In the event that the solution chosen is incapable of meeting the requirements, an alternate solution would be used.

Once the integration technique and the integration parameters have been selected, it is often desirable to verify that the solution met the accuracy specifications. Checking validity may be especially important because of the estimation errors in determining the relative error via the chain model approximation. The validity of the results can be checked by using the actual calculated base matrix for the integration error equations. Also the roundoff error can be bounded by propagating the errors as the problem is being solved using the multiplication and addition equations given in Chapter 4.

8.4 Recapitulation

The primary purpose of this chapter was to determine an efficient algorithm that would automatically select a set of integration parameters which satisfies the error constraint and still requires the minimum amount of computer work. To accomplish this goal the error equations for both roundoff error and integration error were given for the integration methods considered in this report. Accompanying the error equations were the computer work equations in terms of the time step and the number of terms in the base matrix.

Using these work-error equations, a minimizing algorithm was developed. Because of the discrete nature of the problem, an iterative approach proved to be both a simple and an effective solution technique. The algorithm given was applicable to all of the integration methods discussed, although an example was given only for the case of squaring using Taylor series with variable renormalization.

In addition to the work-error equations, the necessary input to determine the integration parameters was described. These parameters are the transition matrix, the final time, and the desired relative error. Options were also left open for the user to specify the desired accuracy for the a specific state, or for the user to specify the power series of choice.

Finally, this chapter brought together the work of the previous chapters to culminate and summarize the important results. One of the goals set forth at the beginning of this document was to develop the ability to determine a solution method that resulted in satisfying the error constraint with the minimum amount of computer work. Clearly then, this goal has been met in this chapter.

Chapter 9

Summary and Conclusions

9.1 Summary of Thesis

This research has resulted in a methodology for bounding the roundoff and integration errors associated with power series solutions for computing the matrix exponential applied to Markov models. This methodology was developed so as to remove some of the user-made decisions that would be necessary to properly compute the matrix exponential. The methodology developed lends itself to a computer algorithm that automatically determines the integration technique and the integration parameters that will result in a user-specified accuracy in the minimum amount of computer work.

Chapter 2 was background material on the Markov process and Markov models. It discussed the characteristics of the Markov model that are important in determining the numerical error incurred in the solution process. The characteristic time of the system was used as a switching point for some of the error propagation patterns. The conservation of system probability was the basis for the roundoff error reduction technique, variable renormalization. It was also used to determine the probability and error of trapping states of the Markov model. A specialization of the Markov model, the chain model, was also presented. Primarily, this chapter was given as background material.

The power series solution techniques of Taylor and Padé for the computation of the matrix exponential were discussed in Chapter 3. It was shown that the Taylor and Padé series were not acceptable solution methods by themselves. Instead, a method known as scaling and propagating was shown to be a beneficial alternative. Two methods of propagating, stepping and squaring, were discussed. Squaring was shown to be more efficient than stepping, in general, with a combination algorithm being the most efficient. The computer work associated with each of the methods was also given. For certain cases, it was shown that the Padé series could have a much higher order of approximation than the Taylor series for equivalent computer work.

Equations that bound the roundoff error were developed in Chapter 4. It was shown that the roundoff error was independent of the series solution method employed. Instead, the roundoff error was found to depend on the number of computer operations required to obtain a solution. The number of operations is directly associated with the time

step, one of the integration parameters that must be selected. A method of reducing roundoff error, variable renormalization, was introduced. This method was shown to significantly reduce roundoff error for most cases. Unfortunately, variable renormalization is only effective with the squaring algorithm. Another roundoff error reduction method, known as the accumulator method, was developed to be used with the stepping routine. While this method results in a quasi-double precision algorithm, it is very computationally intensive and thus, not generally applicable.

The integration error propagation patterns for the power series methods of Taylor and Padé were given in Chapters 5 and 6, respectively. These equations are dependent on the system transition rates and the final time. The error bounding equations are also dependent on the integration parameters of time step and the number of terms in the truncated power series. Because of the many interdependencies, there are various levels of error bounding equations, with the tighter bounds requiring more computer work. The integration error reduction method of Richardson Extrapolation was also explored. It was shown that the extrapolation method resulted in an increase of one order of the approximation for each extrapolation conducted. The integration error for extrapolation was also bounded. An example was given that demonstrated that the bounds for the various methods were conservative.

Once the error propagation patterns had been bounded, the methods were compared in Chapter 7 on the basis of the amount of computer work required to obtain a certain, desired accuracy. This new basis of comparison showed that, in general, increasing the number of terms in the truncated power series reduces the amount of computer work required to achieve the desired accuracy. This study also showed that Padé approximations are more efficient for cases where an order of the approximation greater than four is needed, while Taylor approximations are more efficient for orders of the approximation less than four. The two methods are approximately equal at the fourth order. Richardson Extrapolation was shown only to be beneficial when used with the first order Taylor approximation. For all other cases the increase in accuracy came with too large a computer work penalty.

Finally, Chapter 8 presented a means to determine the integration method and integration parameters which result in the desired accuracy in the least amount of computer work was presented in Chapter 8. Due to the discrete nature of the number of terms in the base matrix, the constrained minimization problem was most easily solved by iteration. The results showed that the optimal solution is not necessarily unique. It may be a Taylor

series or a Padé series solution technique. The number of terms in the base matrix, the size of the time step, and the computer work required all vary with the system parameters and the desired accuracy. The solution of the constrained minimization lends itself easily to implementation on a digital computer.

Chapter 8 culminates in a methodology to automatically determine the integration scheme and parameters. This approach uses the integration methods that were applicable to the general case Markov model. These methods can be tailored to improve efficiency if only a certain class of Markov models are expected. Additionally, the results generated here can be applied to a general solution of the matrix exponential if the procedures that are specific to Markov models are removed. These procedures would include variable renormalization and others that are dependent on the characteristic time, the conservation of system probability, or the chain model approximations.

9.2 Suggestions for Further Work

There are two main areas in which further research is necessary. The first is an implementation of this methodology in a Markov model solver so a wide variety of cases may be tested. The work presented has been tested using a Markov model solver but only for a limited number of cases.

The second area that needs further work is the development of an *a priori* error bound for the matrix inversion process. Efforts to find a compact error bound for matrix inversion in the literature were fruitless; only posteriori bounds were found. Attempts to *a priori* bound this error proved unsuccessful. Currently, the methodology does not bound this error in the Padé series solution method. The error of the inversion process is initially ignored and then calculated once the inversion process is complete. This error is then added in to the final error to insure the desired accuracy has still been achieved. Clearly, an *a priori* bound for the inversion process would eradicate this uncertainty in the error bounding methodology.

References

Bender, Carl M. and Steven Orszag, Advanced Mathematical Methods for Scientists and Engineers, McGraw-Hill Book Company, New York, NY, 1978.

Bickart, T.A., *Matrix Exponential: Approximation by Truncated Power Series*, Proc. IEEE, 56, 1968, pp.872-873.

Butcher, J.C., The Numerical Analysis of Ordinary Differential Equations, John Wiley and Sons, Chichester, U.K., 1987.

Choudhury, A.K., et al., *On the Evaluation of* e^{At}, Proc. IEEE, 56, 1968, pp. 1110-1111.

Dahlqist, Germund and Ake Bjorck, Numerical Methods, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1974.

Dongarra, J.J., C.B. Moler, J.R. Bunch, and G.W. Stewart, Linpack User's Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.

Faddeev, D.K., and V.N. Faddeeva, Computational Methods of Linear Algebra, W.H. Freeman and Company, San Francisco, CA., 1963.

Fair, Wyman and Yudell L. Luke, *Padé Approximations to the Operator Exponential*, Numer. Math., 14, 1970, pp 379-382.

Gear, C. William, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

Healey, M. *Study of Methods of Computing Transition Matrices*, Proc. IEE, 120, 1973, pp 905-912.

Kirchner, R. B., *An Explicit Formula for* e^{At}, Amer Math Monthly, 74, 1967, pp 1200-1204.

Moler, Cleve and Charles van Loan, *Nineteen Dubious Ways to Compute the Exponential of a Matrix*, SIAM Review, Volume 20, Number 4, October 1978.

Møller, Ole, *Note on Quasi Double-Precision*, BIT, 5, 1965, pp. 251-255.

Møller, Ole, *Quasi Double-Precision in Floating Point Addition*, BIT, 5, 1965, pp. 37-50.

Shah, M. M., *Analysis of orundoff and truncation errors in the Computation of Transition Matrices*, Cambridge Report CUED/B-Control TR12, Cambridge, England, 1971.

Stoer, J. and R. Bulirsch, Introduction to Numerical Analysis, Springer-Verlag, New York, NY, 1980.

Strang, Gilbert, Linear Algebra and Its Applications, Second Edition, Academic Press, New York, NY, 1980.

Ward, Robert C., *Numerical Computation of the Matrix Exponential with Accuracy Estimate*, SIAM J. Numer Anal., Vol 14, No. 4, September 1977, pp. 600-615.

Wilkinson, J. H., *Error Analysis of Direct Methods of Matrix Inversion*, J. Assoc. Comp. Mach 8, 1961, pp. 281-330.

Wilkinson, J. H., Rounding Errors in Algebraic Processes, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.

Wragg, A. and C. Davies, *Computation of the Exponential of a Matrix, I: Theoretical Considerations*, J. Inst. Maths Applics, 11, 1973, pp. 369-375.

Wragg, A. and C. Davies, *Computation of the Exponential of a Matrix, II: Practical Considerations*, J. Inst. Maths Applics, 15, 1975, pp. 273-278.